Process algebras for hybrid systems : comparison and development

DOI:
10.6100/IR637804

Document status and date:
Published: 01/01/2008

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Process Algebras for Hybrid Systems: Comparison and Development

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 29 september 2008 om 16.00 uur

door

Uzma Khadim

geboren te Gujranwala, Pakistan

Dit proefschrift is goedgekeurd door de promotor:

prof.dr. J.C.M. Baeten

Copromotor:
dr.ir. P.J.L. Cuijpers

IPA Dissertation Series 2008-25
The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Printing: Eindhoven University Press

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-1398-7

To my father and mother

Contents

Acknowledgements

On the completion of my thesis, I feel grateful for this opportunity I was given to learn and grow. I am grateful to God who gave me the strength to pursue it till the end. I would also like to use this chance to thank the following: To UNUIIST (International Institute for Software Technology), in particular Dr. Tomasz Janowski for giving me a (partial) scholarship for PhD at Eindhoven University of Technology; To Prof. Kees Middelburg who was my first supervisor and who guided me through the basics of the subject; to Bas Luttik for helping me when I needed it. In fact, I am indebted to the kindness of quite a number of people at the Formal Methods Group. I enjoyed and learnt a lot working under the supervision of Prof Jos Baeten. Through Pieter Cuijpers, who supervised me for the last one and a half year, I was able to strengthen my link with the Hybrid Systems world. We had many lively discussions and I feel good about the work we produced together. If you (the reader) find this thesis readable, the credit goes to Pieter, who was very patient (and persistent) when my writing style fell short of his standards. Also to the other members of my defence committee, Jan Bergstra, Jan Broenink and Koos Rooda, for suggesting improvements in the thesis. Other people from whom I benefitted during my work include Bert van Beek, Michel Reniers, MohammadReza Mousavi, Francien Dechesne and my friends at the formal methods group. I am grateful to all my teachers back at home who taught me with compassion and encouraged me. I also want to thank my husband, Faisal, for always showing a very keen interest in my work and encouraging me.

Next, I would like to thank people with whom I shared my first encounter at TU/e. Thank you Ana, for interpreting all the maps and documents for me, for giving directions where ever I needed to go and for being a very considerate and encouraging office mate. Of course I am indebted to the people at the personnel department who helped me with the visa difficulties I faced for coming here and during my stay. Thanks also to Sonja Joosten and Astrid Volkers for helping me with the translations and administrative tasks.

I would like to thank all my colleagues at the Formal Methods Group and also to Jing who has moved to a higher floor. I enjoyed the vlaais, the Tuesdays lunch breaks and the female lunches.

Outside the university, I am grateful for the friendship of Fareeha, Munazza and Uzma Khalique.

Finally, I thank my parents, my parents-in-law, my sisters and my husband for their constant support, encouragement and prayers. Thank you Sumaira, for being here and for the proof reading.

Chapter 1

Introduction

1.1 The Domain of Hybrid Systems

Embedded systems are embedded in our daily routine. Every day from utilizing consumer electronics in our houses, while commuting to our businesses and within our work places, we come across various automatic and semi-automatic control systems. These systems are products of at least two separate domains–one is the software industry and the other is the electrical, mechanical or chemical industry or any combination of them. A simple description of an embedded system, is a digital controller controlling a physical device (as in a thermostat, fire-alarm or a microwave oven). Being a product of different domains, embedded systems exhibit both discrete behaviour of the digital controllers and the non-discrete or real time (continuous) behaviour of its physical components. In the computer science and control theory communities, this dual behaviour is commonly called *hybrid* behaviour. It has given rise to a branch of engineering and research called *Hybrid Systems*.

Together with the increasing use of embedded systems in our daily lives, a branch of formal methods has emerged that aims to verify the correctness of hybrid systems. Formal Methods are concerned with the application of mathematical reasoning to software development. Integrating rigorous mathematical reasoning within the software development cycle, we can identify ambiguities, under-specifications and errors at an early stage of a software system development and hence reduce its development cost [MH04, Pel01].

With the widespread use of embedded software in safety critical systems, it has become extremely important to verify that these systems behave as specified. An error can result in loss of huge amount of investments or worse, loss of lives. Using formal methods, we can to a large extent certify the correctness of software systems and confirm absence of undesired behaviour. Some interesting examples of embedded control systems and the use of formal methods in their development are *verification of the design of a storm surge barrier control system* [Kar98], *verification of the software for embedded controllers for NASA space missions* [NP02, HLP$^+$00, PSE06] and *verification of safety critical embedded software in aeroplanes* [BBF$^+$00].

Realizing the potential role of formal methods in the development of embedded software, many computer science formalisms have been extended with features to model hybrid behaviour of control systems. These include Petri- nets, automata theory, input/output automata, process algebras and action systems. Hybrid extensions of formalisms mentioned include Hybrid Petri-nets [DA01], Hybrid Automata [Hen96], Hybrid Input Output Automata

[LSV03], several process algebras for hybrid systems [RS03, CR05, BMR$^+$06, BM05, Kri06, GBH08] and Hybrid Action Systems [RRS03].

Among the formalisms for modelling hybrid systems, the most popular one is hybrid automata. The popularity of hybrid automata is (in our opinion) for one a consequence of the general popularity of automata. The other factor in acceptance of hybrid automata has been the tool HyTech. HyTech [HHWT97] is a tool for automatic verification for a class of hybrid systems (with restricted flow behaviour) modelled in hybrid automata. The tool HyTech was available almost from the outset of the theory. Tools play a crucial role in the use of any methodology by software industry.

Another formalism for specification and verification of hybrid systems, which is perhaps less popular from an industrial point of view, but which enjoys some popularity among theoretical computer scientists working in the field of hybrid systems, is process algebra. Process algebras are found in several flavors. The three major schools of process algebra are: Calculus of Communicating Systems, [Mil99], Communicating Sequential Processes [Hoa85] and Algebra of Communicating Processes [BBR09, BW90, Fok98]. A nice introduction to process algebra can be found in [BBR09], where a process algebra is compared to a group structure in mathematics. A process algebra has its own signature and set of rules called *axioms*. A process is an element of a process algebra which must satisfy the axioms of the algebra. Process algebra allows compositional modelling of processes–i.e. simple processes are composed together (using operators) to form complex processes. Using axioms, two different representations of a process can be proved to be equivalent. A major contribution of process algebra is the satisfactory description of parallel interactive processes [BBR09].

An advantage of process algebra over other formalisms is its compact symbolic representation of processes, whereas automata theory and petri-nets have more elaborate representations of process behaviour. Model checking is the usual verification method for systems modelled in automata. Model checking of a real life system generates huge intermediate statespaces which in case of model checking by software tools often overflows the memory of a computer. This is commonly known as the *state explosion* problem. An advantage of axiomatic reasoning of process algebra over model checking, is that axiomatic reasoning can be applied compositionally on a complex process. First all components of a complex process can be simplified, and then axioms are applied to the composition of the simplified processes. This is one way in which process algebra tries to attack the state space problem.

Where we count compact representation of processes as an advantage of process algebra, it is also a source of disadvantage. It is a cause for its comparatively less popularity in the industry, as people find process specifications in process algebra obscure and un-tangible. There is a need to develop tools that have front ends with more tangible representation of processes and their back ends implement the mathematical rigor of process algebra. Among process algebras, CSP has been more successfully used in the industry. Examples of industrial application of CSP are the programming language occam and tools FDR, Casper and ProBe [CSP].

With the increase in complexity of computer systems, many extensions of process algebras have been built. Now besides discrete event systems, process algebras can model timed systems [RR88, NS94, BM02a], stochastic systems [And02] and hybrid systems [RS03, CR05, BMR$^+$06, BM05]. An account of the developments in process algebra and its history can be found in [Bae05]. Preliminary work on combining hybrid and stochastic systems in one process algebra can be found in [Hil05, BLB08].

1.2 Research Problem

This thesis is concerned with the formal specification and analysis of hybrid systems through process algebras. Several process algebras for hybrid systems exist in literature. We do not intend to develop a new process algebra. Rather we take the task of building some useful extensions for the existing theories.

We begin with a survey of existing process algebras for hybrid systems, including HyPA[CR05], ϕ-calculus [RS03], Hybrid χ [BMR$^+$06] and ACP^{srt}_{hs} [BM05] in our study. Our aim is to develop an insight into the essential features of a process algebra for hybrid systems, to understand the contemporary issues in the field and to present the state of art in process algebras for hybrid systems. Our study points out a number of possibilities for future work. From them we chose two for further development in this thesis, namely, extending ACP^{srt}_{hs} with variable abstraction; and developing a more efficient linearization algorithm for Hybrid χ, in terms of the size of the linear form, than is currently available.

Linearizing hybrid χ process terms is described in Chapter 4 of our thesis.

An initial attempt at adding variable abstraction to ACP^{srt}_{hs} is given in [Kha05]. During our work on adding variable abstraction to ACP^{srt}_{hs}, we discovered that the semantics given in [BM05] is such that the choice operator is not associative and a number of axioms, including the axiom of time determinism (SRT3), do not hold. Hence, we had to change our course from extending ACP^{srt}_{hs} to correcting it. The errors appearing in [BM05] are also present in [Kha05]. Chapter 3 of our thesis presents a correction of $BPA^{srt}_{\perp}$, an essential sub-algebra of ACP^{srt}_{hs}.

1.3 Positioning of our work

The research included in this thesis comprises of three main projects, i.e. a comparative study of process algebras for hybrid systems; a basic timed process algebra with non-existence and linearizing Hybrid χ. These are presented in three different chapters. In this section, we give the research problem addressed in each chapter, prior work related to the problem and our approach towards solving it. We describe the limitations of the proposed solution and discuss methods to evaluate our results.

The positioning of work undertaken in our thesis is discussed below:

1. **A comparative study of process algebras for hybrid systems**

   Today, a number of process algebraic theories for specification and analysis of hybrid systems exist. Examples are Hybrid Process Algebra (HyPA), ϕ-calculus, Hybrid χ, ACP^{srt}_{hs} and $BHPC$. These theories have been developed independently of each other; they target slightly different application areas and use different notations. Although each of the works mentioned above refers to the others briefly, a thorough comparison among these process algebras is missing. A comparison can reveal omissions in a theory and suggest improvements. It can help a user interested in applying these theories make a suitable selection according to the problem at hand and it can also guide the development of new process algebras for hybrid systems, for example see [GBH08].

   In chapter 2, we compare four process algebras for hybrid systems namely HyPA, ϕ-calculus, Hybrid χ and ACP^{srt}_{hs}. We study the operators of each process algebra both syntactically and semantically. We distinguish their common features and point out their differences. We compare how each process algebra deals with issues such as discontinuities in variables, choice resolution and parallelism. We compare their available tools. We model a case study in each process algebra, apply the algebraic reasoning (using axioms) offered by the algebra to simplify the specification of case study and try to reason about a safety condition.

Our comparison is not complete in the sense that,at the moment of this writing, we are aware of three other existing hybrid process algebras, namely, BHPC [Kri06], HYPE [GBH08] and Hybrid CSP [Jif94], that have not been included in our study. The process algebra BHPC was developed almost parallel to this comparative study which is why we could not take it into account at that time. Similarly, the process algebra HYPE has just been presented in 2008. Currently, we are renewing our study in [KC], where BHPC is included. The paper [Jif94] introducing Hybrid CSP gives a predicate semantics to Hybrid CSP constructs. The predicate semantics is very different from the nowadays usual structural operational semantics. We do not understand the semantics of some basic operators in [Jif94] which is why we could not include it in the comparison. Finally, some tools for analyzing process algebraic models, namely, the SPHIN model checker for ϕ-calculus and the linearization tool for HyPA, could not be used successfully for the analysis of the case study included in our comparative study. We could not get guidance for the problems encountered by us in using the tools for the analysis of our case study because research on these tools has been discontinued and there is not much user support available.

The comparative study identifies common features of process algebras studied and highlights their differences. It suggests a number of improvements in the theories studied. In case of ϕ-calculus, our suggestions have been taken into account by the authors, see [RS].

2. **Basic timed Process Algebra with Non-existence**

ACP_{hs}^{srt} [BM05] is a well-known process algebra for hybrid systems. It is an extension of a timed process algebra ACP^{srt} [BM02a] and Process Algebra with Propositional Signals ACP_{ps} [BB97].

During our work on adding variable abstraction to ACP_{hs}^{srt}, we found a number of errors in it. Namely, the semantics given in [BM05] is such that the choice operator is not associative, the axiom of time determinism (denoted by SRT3) is not preserved by the semantics and a number of other less important axioms given in [BM05] do not hold.

The errors in ACP_{hs}^{srt} mainly stem from the fact that the property of weak time determinism (taken from ACP^{srt}) was not correctly extended to a hybrid setting. The presence of signals in ACP_{hs}^{srt} (inherited from ACP_{ps}) complicates the task.

ACP_{hs}^{srt} is a rich process algebra with a large number of operators and an elaborate semantics. Rectifying the whole of ACP_{hs}^{srt} turns out to be a non-trivial task. We adopt a modular approach towards correcting ACP_{hs}^{srt}. The process algebra ACP_{hs}^{srt} is built hierarchically from a number of simpler process algebraic theories. The error in the axiom of time determinism, (SRT3), can be traced down to one of its basic theories $BPA_{\perp}^{srt}$. Therefore, fixing $BPA_{\perp}^{srt}$ is the first step in rectifying the errors in ACP_{hs}^{srt}. Accordingly, in Chapter 3, we present two proposals for a sound process algebra $BPA_{\perp}^{\mathrm{srt}}$.

We accompany our proposals with proofs. Finally, we discuss extending $BPA_{\perp}^{\mathrm{srt}}$ to a full hybrid process algebra.

Our work in chapter 3 identifies the problem in ACP_{hs}^{srt}, provides a solution to part of the problem and paves the way for further development in finding a solution.

We have accompanied our proposals for $BPA_{\perp}^{\mathrm{srt}}$ with complete proofs. Experience shows that proofs can be wrong. We do our best to ensure correctness of the proofs by emphasizing clarity and completeness. One way to check the soundness of the axioms is to use existing tools such as theorem provers (for example, Coq [Coq] and PVS [PVS]) or soundness provers (such as [Wee]). Using existing theorem provers to prove soundness of axioms requires some effort from the user. Using the tool implementing the algorithm in [Wee] is simpler. But the tool is yet is an experimental tool and does not guarantee results. Another method to verify properties of operators in a process algebra is to take help from structural operational semantics theory. In literature, meta theorems giving formats for structural operational rules that guarantee certain properties for operators are available [CMR08] and [Mou05]. But such meta theorems are only available for very limited number of properties (i.e., commutativity and associativity) and are applicable on a restrictive rule format. While we use the commutativity format of [Mou05] to prove the commutativity of the choice operator, the format for associativity given in [CMR08] cannot be used. As of right now the associativity format does not allow negative premises, which we require in the rules used in our proposals for $BPA_{\perp}^{srt}$.

3. **Linearizing Hybrid χ**

   A contribution of process algebras in general is the satisfactory description of parallel interactive processes [BBR09]. A parallel operator is a compact representation of the simultaneous execution of two processes. Its semantics includes details such as synchronization, communication and interleaving of actions of the process terms executing simultaneously. A parallel operator is very useful for specifying a multi-component system. On the other hand, the analysis of specifications with parallelism is difficult. In different process algebras, see [Use02, CR05, BM05], tools and techniques exist that can only be applied to process terms without a parallel operator. Hence, methods to rewrite specification into an equivalent term without parallel operator are important for analyzing a specification in a process algebra.

   Linearization is a process of rewriting a process term into a simpler form called a *linear* term. Linearization is similar to *elimination* found in many ACP style process algebras such as [BW90, BM02a, BM05, CR05, BMR$^+$06]. Where elimination theorems are applicable only to process terms without recursion, linearization also applies to recursive processes. A linear term (also called a *basic* term) consists of only basic operators of an algebra such as actions, choice and sequential composition. In particular, a linear form does not contain a parallel operator. Historically, the authors of μCRL [GPU01], a process algebra with data, first used the term "linear process equation" (LPE) for its basic terms, and referred to the procedure of rewriting a process specification into a basic term as *linearization.*

   A linear form models all process behaviour using only basic operators. A consequence is that the size of a linear form of a process term with parallel composition can be exponential to that of the input term. A challenge in linearization is to contain the increase in size of the linear form during the elimination of a parallel operator.

Linearization algorithms and tools for μCRL [Use02], HyPA [BRC06] and Hybrid χ [The06] are available. The linearization algorithms of μCRL and HyPA use stack-like data structures to reduce the increase in size of a linear form of a parallel composition. Hence, the size of the resulting linear term is believed to be of the order of the input process term, although this claim has never been formalized.

The linearization algorithm of Hybrid χ [The06] does not use any special data structures or techniques during linearization. The size of the linear form resulting from that linearization algorithm can be exponential to the size of the input process term.

In [Use02], it is mentioned that in place of stacks, discrete counters can be used for linearization of a restricted set of processes. This should result in a simpler algorithm. In chapter 4, we develop a linearization algorithm for a subset of Hybrid χ process terms using discrete counters. We use the counters to represent interleaving of actions when a parallel operator is removed from a specification. Using these counters reduces the increase in the size of a specification during linearization. We claim without proof that the resulting linear term from our linearization algorithm is of the order of the square of the size of the input term. Hence, our linearization algorithm is an improvement over the linearization algorithm given in [The06].

As mentioned, the algorithm applies only to a subset of Hybrid χ process terms.

We do not give a proof of correctness of the linearization algorithm showing that the linearized process term behaves bisimilar to the input process term. It is intended that the work done is evaluated by using a tool implementing this linearization algorithm. The development of a linearization tool by the developers of Hybrid χ is currently delayed in favor of the ongoing work on the semantics of process algebra Hybrid χ.

1.4 Other Work on Hybrid Systems

Research in Hybrid Systems is pursued actively in both computer science and control theory communities.

In computer science, several formalisms besides process algebras, (e.g. Petri-nets, automata theory, I/O automata theory), have been extended with features to model the continuous dynamics of hybrid systems. See [DA01, Hen96, LSV03] for hybrid extensions of the afore mentioned formalisms.

The formalisms in computer science are mainly focussed on specification and verification of hybrid systems. Among them, model checking is the most commonly used technique for verifying hybrid systems. Model checking requires building a finite state model of the system to be verified. The state space generated by the physical component of a non-trivial hybrid system is infinite. This poses a challenge for model checking hybrid systems. In [LPY99, LPY01, Hen96], classes of hybrid systems are identified that have equivalent finite state models. For an overview of recent techniques applied in model checking of hybrid systems, see [ADF$^+$06]. Examples of model checkers for hybrid systems are HYTECH, d/dt, CHECKMATE, Verishift, PHAVER etc. For a summary of tools for model checking hybrid systems, see [SSKE01]. Often model checkers for *timed* systems are also used to verify models of hybrid systems after calculating time bounds manually. Examples are KRONOS and UPPAAL. Besides model checkers, many other tools are available for modelling and analysis of hybrid systems. Examples are CHARON [AGH+00], HYSDEL [TB04], Omola/OmSim [BM97], HSML [TK95] and STeP [BBC+96].

Verified or not, hybrid systems have been in use for a long time in control theory as multi-mode dynamic systems. Examples of systems that switch between multiple dynamics can be found in mechanics (examples are friction models, robotics, aircraft landing, etc.) and electronics (diodes, switched capacitor filters, analog to digital converters, etc.). Sometime ago, these systems were studied using methods that concentrated only on one aspect (discrete or continuous) of the hybrid systems behaviour. Methods also existed that studied and analyzed discrete and continuous parts of hybrid systems separately only to combine them in the final stages [Hee99]. Now several control theory methods exist that study both discrete and hybrid behaviour together. See [Son98],[SS98],[BM99], [HSB01], [ALQ+02] for some control theory formalisms for hybrid systems.

Chapter 2

A Comparative Study of Process Algebras for Hybrid Systems

In this chapter, we present a comparative study of process algebras for representing hybrid systems. We take four recent process algebras for describing hybrid systems, namely, Hybrid Process Algebra (HyPA) [CR05] , ϕ-Calculus [RS03], Hybrid χ [BMR$^+$06] and Process Algebra for Hybrid Systems (ACP_{hs}^{srt}) [BM05]. We make a comparison based on their history, their operators and semantics, and the available tooling. Our aim is to develop an insight into the essential features of a process algebra for hybrid systems; to survey recent literature for available theories; compare available process algebras syntactically and semantically; to draw conclusions regarding usability, expressiveness and ease of modelling for each process algebra. To achieve this, we model a simple case study in all four process algebras. The case study comprises of a common example of train gate controller system as given in [BM05].

We have included four process algebras HyPA, ϕ-calculus, ACP_{hs}^{srt} and Hybrid χ in our study. All these process algebras appeared between 2003 and 2006. The algebra ϕ-calculus is different from others in that it is an extension of π-calculus, a CCS style process algebra; HyPA and ACP_{hs}^{srt} are ACP style process algebras and Hybrid χ takes the communication constructs of CSP. The latter three originated from Eindhoven University of Technology (Netherlands) and ϕ-calculus was developed in University of Michigan (USA).

Two prominent process algebras for hybrid systems that are missing from this list are the most recent algebra BHPC [Kri06] and perhaps the oldest process algebra for description of hybrid processes, Hybrid CSP [Jif94].The process algebra BHPC was developed almost parallel to this comparative study which is why we could not take into account at that time. Currently, we are renewing our study in [KC], where BHPC is included. The paper [Jif94] introducing Hybrid CSP has a predicate semantics. We find the semantics given in [Jif94] unclear about the meaning of some basic operators (like chop operator) which is why we do not include it in the comparison.

The rest of the Chapter is outlined as follows: First of all we mention the changes that took place in each process algebra since this comparative study was first presented in [Kha06]. In Section 2.1, we begin with a list of common features of process algebras for describing hybrid systems, operators to model discrete changes and continuous flows, conditionals, types of process equivalences used, variable abstraction etc.. In the light of these common features, the next section, Section 2.2, gives a brief introduction of all four process algebras. In Sections 2.3 and 2.4, we address the issues of “flow determinism” and discontinuities taking place in a hybrid process. Section 2.5 describes concurrent composition of hybrid processes in all process algebras. Section 2.7 compares available tools and finally we specify the train gate controller system in each process algebra in Section 2.6 and draw conclusions from the specifications.

Current State of the Art

The field of hybrid systems is an active area of research. A lot of research is going on in the process algebras mentioned here that is not yet finished nor published. Below, we mention the current state of art in each process algebra.

The process algebra HyPA is an extension of ACP. It first appeared in 2004 (see [CR05]) and later a variable abstraction operator was added to it in [BRC06]. Tools for linearization and simulation were added later [BRC06, Sch05]. The version of HyPA given in the comparative study here discusses the variable abstraction operator and additions in tools.

The process algebra ϕ-calculus first appeared in 2003 (see [RS03]). Later a thesis (see [Son05]) by Hosung Song presented an extension of model checker SPIN for ϕ-calculus processes. (Also see [RSC06] for a discussion on model checker SPHIN). A (preliminary) version of spatial logic has also been proposed as a general assertion language for ϕ-calculus in [Rou04].

A discussion in [CR05], prompted changes in the theory of ϕ-calculus. It was decided to allow a more wider class of continuous behaviour than can be modelled by a valuation of derivatives to the set of continuously differentiable functions. Also we found an error in the weak bisimulation theorem of [RS03] as urgent communication was not reflected properly in the delay rule for replication operator (see section 2.2.2). Both these changes have been incorporated in a later (unpublished) version of ϕ-calculus [RS]. In our comparative study we refer to both [RS03] and the later version of ϕ-calculus in [RS].

Hybrid χ is an extension of the language χ which is a modelling and simulation language for industrial systems [MFR95, MFRvdN95]. It attained its current form (see [BMR$^+$06]) in a number of iterations in which the features to model dynamic behaviour of systems were added to it and it was given a formal semantics. Translations from Hybrid χ to a number of formalisms for describing hybrid systems have been defined (see [BRS$^+$07]). A lot of research is going on in the theory of Hybrid χ as well as for its tools. Recently, the syntax of Hybrid χ has been improved further and some complexities from its semantics have been removed. See [BHR$^+$08] for Hybrid χ 2.0. In our comparative study, we include the version of Hybrid χ given in [BMR$^+$06].

ACP^{srt}_{hs} is an extension of timed process algebra ACP^{srt} [BM02a] and Process Algebra with Propositional Signals ACP_{ps} [BB97]. It first appeared in 2003 (see [BM03]). It was then found that the definition of variable abstraction operator (called signal hiding in [BM03]) was not correct. ACP^{srt}_{hs} appeared without a signal hiding operator in [BM05]. In [Kha05], a graph model was presented for the process algebra and a signal hiding operator was introduced in the graph model of ACP^{srt}_{hs}. In the comparative study of process algebras [Kha06], it was pointed out that the semantics of ACP^{srt}_{hs} does not preserve time interpolation. Later it was found out that the Choice operator of ACP^{srt}_{hs} is not associative and a number of axioms are unsound. The errors found in ACP^{srt}_{hs} and the proposed corrections are discussed in detail in Chapter 3. As discussed in Chapter 3, by changing the semantics of the Choice operator the error in associativity can be corrected. ACP^{srt}_{hs} contains many constructs for modelling hybrid systems. In this comparative study, we discuss different features of ACP^{srt}_{hs} and also mention the recently discovered errors. The comparative study given here includes the version of ACP^{srt}_{hs} given in [BM05].

2.1 General Characteristics of a Hybrid Process Algebra

Although the development of hybrid process algebras has not come to a stand still yet, a consensus has been reached on the main ingredients and the main development choices. In this section we give a general list of features considered essential for modelling hybrid processes in a process algebra.

- **Data:** A process algebra for hybrid systems consists of a data part in addition to process terms. This data part is composed of valuations of environment variables that represent physical entities in the environment. Examples of environment variables are temperature, water-level, pH of a solution and speed of a vehicle.

- **Modifications:** In the physical world, entities that are modelled by environment variables vary in two ways:

i. **Instantaneous changes:** Modifying a variable instantaneously usually requires a system action. For example, dropping an electric current to zero in a circuit, requires opening a switch. Instantaneous modifications to environment variables are represented by predicates in terms of previous and new values of variables.

ii. **Gradual changes:** Gradual changes over time are governed by predicates over variables that may contain algebraic equations, inequalities, differential equations and differential inclusions. For example, $20 \leq x \leq 40$ can be a predicate that restricts the value of the variable x between 20 and 40. The gradual changes in an environment variable during a delay are governed by a predicate and are represented by a trajectory over time. The treatment of time is of interest when representing gradual changes in an environment.

- **Passage of Time**: Passage of time is represented differently by different process algebras. One approach is to have a special variable *time* representing time, as in Hybrid χ. Another is to have a delay operator that introduces a delay between subsequent discrete actions, as in ACP^{hs}_{srt}. Yet another approach is not to have any special constructs but simply define a variable with derivative equal to one. HyPA and ϕ-calculus follow this approach. This variable can then act as a clock or timer.

- **Absolute versus Relative Time**: In absolute timing, passage of time is measured from the start of the process. For example, the variable *time* in Hybrid χ refers to time elapsed since the start of a process.

In relative timing, time is measured since the last action executed. This is the case in ACP^{srt}_{hs}.

- **Continuity requirements of variables:** Talking about gradual flows and trajectories brings us to the question of whether an environment variable is allowed to jump during a flow or not. Depending on the variables' requirements, we can categorize functions representing trajectories of variables into the following commonly used categories:

i. *Continuous*: The value of a variable cannot jump during a flow. Roughly speaking, it means that we can draw the trajectory of the variable without lifting the pencil off the paper.

ii. *Piece-wise Continuous*: The value of a variable can jump a finite number of times during a delay.

iii. *Continuously differentiable* Both the variable as well as its derivative cannot jump during a delay.

iv. *Piece-wise Continuously differentiable*: Both the variable as well as its derivative can jump a finite number of times during a delay.

- **Guards:** A common requirement of hybrid process algebras is the ability to express conditional actions or conditional delays. A *guard* is a predicate on the environment variables. A guarded action or guarded delay can only be executed if the guard evaluates to true. For example, we may need to express the following statement in a specification.

if speed of the vehicle becomes greater than 80 km/hr, release the pressure from the accelerator

- **Specifying assumptions about the environment:** We may want to study the behaviour of a process under some assumptions about its environment. For example we may want to study the acceleration of a car on a metalled road, disregarding air-resistance. It is desirable to state these assumptions formally in a system specification as they are critical in determining the behaviour of a process.

In ACP^{srt}_{hs} and Hybrid χ, initial conditions and system invariants can be specified by means of different operators and constructs. (Refer to sections 2.2.4 and 2.2.3).

- **Variable Abstraction:** The description of the physical component of a hybrid system requires some variables. The presence of a large number of variables clutters the state space of a large hybrid system and thus making it more complex to calculate its behaviour. *Variable abstraction* offers a way to reduce this complexity and provides a modular (step-wise) approach to the analysis of large systems. In practice, multi-component systems can be designed in such a way that most variables only influence the behaviour of a single component. In other words these variables are local or private to a component of the system. In variable abstraction we calculate the behaviour of a system component and then hide the details of changes taking place in its local variables. The interaction of this component with another component is then calculated while ignoring its local variables. The local variables are no more observable although their effect on the behaviour of a component is visible. Variable abstraction is also called *signal hiding*.

All process algebras except ACP^{srt}_{hs} describe a way of defining local variables.

- **Semantics:** The process algebras for hybrid systems that we study here give an operational semantics to processes and associate a hybrid transition system with a process. A process term together with a data part, usually a variable valuation, constitute a hybrid process. A hybrid transition system consists of transitions for both discrete actions as well as continuous evolutions of a hybrid system.

The definition of a hybrid transition system (taken from [CR05]) is given below: A hybrid transition system is a tuple $(\mathbf{X}, \Sigma, \mathbf{T}, \varphi)$, where,

1. $\mathbf{X}$ is the state space;
2. Σ is the alphabet of the transition system. Elements of Σ appear as labels to transitions between states;
3. $\mathtt{T}$ denotes the time axis; and
4. φ defines the transition relation of the hybrid system.

The state space $\mathbf{X}$ consists of pairs of process terms and data (usually a variable valuation). In a hybrid system, a state can evolve into another through discrete actions or time delays during which the variable valuation changes. Therefore Σ contains both a set of discrete actions (denoted by Σ_d) and a set of continuous trajectories (denoted by Σ_c). Elements of the time axis $\mathbf{T}$ represent duration of delays. The transition relation φ defined as

$$\varphi \subseteq X \times ((\mathbf{T} \mapsto \Sigma_c) \cup \Sigma_d) \times X,$$

includes transitions between states through discrete actions and time delays. The function $\mathbf{T} \mapsto \Sigma_c$ gives the trajectory of the data part of a hybrid process during a delay. Let σ be an element of $\mathbf{T} \mapsto \Sigma_c$, then a time transition is represented as $x \overset{\sigma}{\longmapsto} x'$, where $x, x' \in X$.

Other semantics for hybrid processes can be found in [Jif94] and [CJR96] which give a predicate semantics and a semantics in duration calculus respectively to processes defined in hybrid CSP.

• **Process Equivalence:** Equivalence on processes is defined in terms of bisimulation relations on hybrid transition systems. In [Mou05], three different notions of bisimulation for transition systems with data are given. A comparison of behaviour between processes with same data parts is made. The three notions of bisimulation are as follows:

i. State-less Bisimilarity
ii. Initially state-less Bisimilarity
iii. State-based Bisimilarity

The difference in these three notions is that the behaviour of the two processes can be compared for just a given data part or for all possible data parts. Consider two process terms P and Q and a set of all possible data values D.

- In state-less bisimilarity, at each transition step, from start till the termination, the behaviour of the two processes is compared in all possible data states. That is, P and Q are compared for all $d \in D$. If $\langle P, d\rangle$ makes a transition to $\langle P', d'\rangle$ and $\langle Q, d\rangle$ makes a similar transition to $\langle Q', d'\rangle$, then P' and Q' are again compared for all $d \in D$.
- In initially state-less bisimilarity, as the name indicates, only initially the behaviour of the two processes is considered in all possible data states. In all subsequent steps, the two processes are compared only in the data state resulting from the previous step. That is, if $\langle P, d\rangle$ makes a transition to $\langle P', d'\rangle$ and $\langle Q, d\rangle$ makes a similar transition to $\langle Q', d'\rangle$, then P' and Q' are compared only for data state d'.
- In state-based bisimilarity, the behaviour of the two processes is initially compared in a given data state. After the first step, the behaviour of the two processes is compared in the valuation resulting from the previous step.

• **Time determinism:** A process is said to be timed deterministic if passage of time by itself cannot resolve possible choices available to a process. On the other hand, if while delaying, an option of behaviour can be dropped in favor of another, then the process is said to be timed non-deterministic. Mathematically, time determinism is expressed as follows:

Let s, s' and s'' be process terms and m denote the duration of a delay , then

$$\text{if } s \overset{m}{\longmapsto} s' \text{ and } s \overset{m}{\longmapsto} s'', \text{ then } s' \equiv s''$$

In hybrid process algebras we have to cater for variable evolution during delays as well. Therefore, flow determinism is more relevant than time determinism. Flow determinism means that a unique flow in a hybrid system leads to a unique target state.

In literature for timed processes, see [BBR09], we find systems exhibiting three types of time determinism.

1. **Strong Time determinism:** In strong time determinism, an alternative composition of different process terms can delay, without resolving choices, under following conditions:
i. All terms in the composition must be able to delay.
ii. A common variable evolution during delays is possible for all process terms. Note if all process terms allow more than one possible variable evolution, then a variable evolution is chosen non-deterministically. So the resolution of choice between process terms is postponed but a choice between possible variable evolutions is resolved.

As soon as all process terms cannot delay together, the alternative composition cannot delay and choice must be resolved in favor of doing an action.
2. **Weak Time determinism:** In weak time determinism, an alternative composition between delayable process terms can delay with a variable evolution that is possible for all process terms, without resolving choices between process terms. When a process term cannot delay any further, then the choice is non-deterministically resolved between doing an action (of one of the process terms) or delaying according to the rest of the delayable process terms in alternative composition.
3. **Time non-determinism:** In time non-determinism, an alternative composition cannot delay while retaining choices. As soon as an alternative composition starts delaying, the choice between process terms has to be resolved even if both the process terms allow for the same variable evolution during delay.

HyPA follows a time non deterministic approach in alternative composition. ϕ-Calculus and Hybrid χ have strong time determinism. ACP_{hs}^{srt} has weak time determinism.

The next section introduces each process algebra and discusses both continuous and discontinuous behaviour of environment variables.

2.2 A Brief Introduction to Process Algebras for Hybrid Systems

In the line of the general characteristics presented in the last section, we now study the four process algebras. We try to determine to what extent the requirements mentioned in Section 2.1 have been met in all process algebras under study.

Admittedly, our approach is syntactic and a lot of notation related to different process algebras is introduced in this section. But we accompany the details with examples and it prepares the reader for the more semantic discussions coming later. We conclude the account of each process algebra with our impressions in a few lines. The section can also be referred back to later as the need arises.

In the end of this section, a summary of operators present in each process algebra is given in Table 2.5.

2.2.1 HyPA

HyPA [CR05] stands for hybrid process algebra. It is an extension of ACP. A HyPA specification consists of a set of environment variables. Environment variables are also called *model* variables in HyPA, indicating that they depend upon the system which is to be modelled.

The set of process terms P of HyPA can be given by a BNF expression, see Table 2.1.

Table 2.1: Syntax of HyPA

$P ::=$	δ	Deadlock	
	$\mid \epsilon$	empty process	
	$\mid a$	discrete, atomic actions	$a \in A -$ a set of actions
	$\mid c$	flow clause	$c \in C -$ a set of flow clauses $c \equiv (V \mid P_f)$
	$\mid d \gg P$	reinitialization operators	$d \in D -$ a set of reinitialization clauses $d \equiv [V \mid P_r]$
	$\mid P \blacktriangleright P$	disrupt operator	
	$\mid P \rhd P$	left disrupt operator	
	$\mid P \oplus P$	alternative composition	
	$\mid P \odot P$	sequential composition	
	$\mid P \parallel P$	parallel composition	
	$\mid P \mid P$	forced communication	
	$\mid P \lfloor\!\lfloor P$	left parallel operator	
	$\mid \partial_H(P)$	encapsulation operator	$H \subseteq A$
	$\mid \lvert[V \mid P]\rvert$	Variable Abstraction	

where V is a set of model variables and P_f, P_r are the reinitialization and flow predicates respectively, discussed in the following paragraphs.

HyPA contains constructs to manipulate environment variables during actions and delays of the hybrid system under consideration.The special features of HyPA are as follows:

- **Reinitialization clauses:** In HyPA, as actions are performed, by default the values of environment variables remain the same. To instantaneously modify environment variables, *reinitialization clauses* are used. A reinitialization clause is of the form $[\ V \mid P_r\ ]$, where V is the set of variables that are allowed to jump and P_r is the reinitialization predicate that must be satisfied by the old and new values of the variable. For example,

$$\left[\begin{array}{c|l} x & x^+ = 0 \\ r & r^+ = 2 * r^- \end{array}\right]$$

The variables with + superscript denote new values and the ones with − superscript denote old values of variables x and r.

The set V is omitted from a reinitialization $[\ V \mid P_r\ ]$, in case no variables are allowed to jump. The reinitialization clause then acts as a guard on variable values. For example,

$$[\ \text{temperature}^- \geq 20\ ] \gg \text{turnoff}$$

It means if the value of temperature is 20° or more, do action turnoff.

• **Flow clauses:** To model the behaviour of the environment variables while the system is idling, *flow clauses* are used in HyPA. A flow clause $(V \mid P_f)$ consists of a flow predicate P_f according to which the variables evolve, and a set of environment variables V. Variables whose values should remain continuous as a new flow clause takes over are mentioned in the variable part of the flow clause. Other variables that are not mentioned can jump initially. (Note that this is opposite to a reinitialization clause, where the variables that are allowed to jump are explicitly mentioned.) For a flow clause $(V \mid P_f)$, the initial jump of the variables not in V should be such that the new values satisfy the predicate P_f.

A flow predicate can be an algebraic or differential equation (or inequality) or a differential inclusion. In a HyPA model, a reinitialization clause is often used before a flow clause to set the initial values of the continuous variables. For example,

$$[\ x \mid x^+ = 30\ ] \gg \left(\ x \ \middle|\ \begin{array}{l} 0 \leq \dot{x} \leq 1 \\ x > 0 \end{array} \ \right)$$

The variable x is assigned 30 by the reinitialization clause before the flow clause. As x is declared to be continuous in the flow clause, therefore all possible trajectories of variable x will start with value 30. The derivative of x is not initialized. It can have any value between 0 and 1 at the start of the flow.

In HyPA the set of environment variables, the set of discrete actions, the set of flow predicates, the set of reinitialization predicates, the set of time points and the set of possible solutions to flow and reinitialization predicates are all parameters of the HyPA theory. Therefore depending upon the system to be modelled, the notion of solution to a flow predicate can vary.

• **Disrupts:** The flow clauses represent infinite, non-terminating behaviour. A disrupt operator ($\triangleright$ or $\blacktriangleright$) is defined in HyPA through which an action or a new flow clause can interrupt a previous flow clause. The system then continues to behave according to the action or the flow clause following the disrupt operator.

• **Representation of time:** There is no special operator to represent passage of time. A variable initialized to zero and with derivative equal to one can be used as a timer. For example, in the following specification, a process delays for 10 time units and then continues as process P.

$$[\ t \mid t^+ = 0\ ] \ \gg (t \mid \dot{t} = 1) \blacktriangleright \ [\ t^- = 10\ ] \ \gg P$$

The disrupt operator and the reinitialization clause $[\ t^- = 10\ ]$ intercept the flow of the system when variable t becomes 10.

• **Variable Abstraction:** HyPA is extended with a variable abstraction operator (see [BRC06]). The variable abstraction operator is denoted by $|[V \mid P]|$. The process $|[V \mid P]|$ behaves exactly as P, but in a transition system, the continuous and discontinuous changes to a variable in V appear to be arbitrary. The values in the transition system do not reflect the actual changes taking place. Another process composed with $|[V \mid P]|$ can only affect the values of variables in a transition system and cannot modify the actual values of local variables.

• **Semantics:** There are three different transition relations defined in the hybrid transition system of HyPA. They are: the action transition (to represent actions), the time transition (to represent gradual flows of systems) and the termination predicates (termination of a process).

- **Bisimulation:** Two types of bisimulation are defined in HyPA. One is called the *robust bisimulation* (denoted by $\approx_r$) that matches the notion of stateless bisimulation. The other called *bisimilarity* (denoted by $\approx$) matches the notion of initially stateless bisimulation.

Bisimilarity is only applicable for analysis of parts of the systems which have been linearized, i.e. the parallel operator has been removed from them. The parallel operator does not preserve bisimilarity. Two process terms that are bisimilar to each other may not remain bisimilar when composed in parallel with a third process. For example consider process terms X and Y, where,

$$\begin{array}{l} X:\ [\ x \mid x^+ = 1\ ] \ \gg a_1 \odot\ [\ x^- = 1\ ]\ \gg a_2 \\ Y:\ [\ x \mid x^+ = 1\ ]\ \gg a_1 \odot a_2 \end{array}$$

The symbol $\odot$ denotes sequential composition. X and Y are bisimilar to each other. The second reinitialization does not change the value of variable x but only acts as a guard. Let Z be another process also containing the environment variable x.

$$Z:\ [\ x \mid x^+ = 2\ ]\ \gg a_3$$

The process term $X \parallel Z$ is not bisimilar to $Y \parallel Z$. As a sequence of actions '$a_1a_3a_2$' is possible for $Y \parallel Z$, but performing a_2 after a_3 is not possible for $X \parallel Z$ because of the guard $[\ x^- = 1\ ]$. Bisimilarity is not a congruence with respect to the parallel operator because of variable sharing between the processes composed in parallel. In a parallel composition of $P \parallel R$, at any stage in the execution of P, R may change the valuation that can affect the behaviour of P. Initially state-less bisimilarity does not cater for possible interferences that change the valuation during the execution of a process.

On the other hand, robust bisimulation caters for possible interferences by processes running in parallel. Therefore it differentiates between the process terms X and Y. Robust bisimulation is a congruence for the parallel operator.

HyPA has a large set of axioms and some derivation rules. These axioms and derivation rules are sound with respect to robust bisimulation. A HyPA system specification can be simplified by repeatedly applying these rules and axioms. Every HyPA term can be written as a basic term which is free of the parallel operator. Elimination of parallel operator greatly simplifies a system. Two axioms that are not robust bisimilar but initially stateless bisimilar can be applied on system specifications without parallel operator. These axioms incorporate results from real analysis into process equations.

When recursion is used in a process algebraic specification, axiomatization alone does not suffice to reason about equivalence. In process algebra, there is an important theorem saying that recursive specifications in the *guarded form* have exactly one solution. This aids in reasoning about such specifications. This recursive specification and definition theorem has been proven for HyPA and ACP^{srt}_{hs}, but has not been proven for Hybrid χ and ϕ-calculus.

- **Time non-determinism:** HyPA's alternative composition operator is time non-deterministic. That means the choice resolution cannot be delayed even if the process terms in alternative composition may allow mutual evolution of variables. Consider the following HyPA expression,

$$\langle (x \mid \dot{x} = 1) \oplus (x \mid \dot{x} \geq 1), \{x \mapsto 0\} \rangle$$

The symbol $\oplus$ denotes alternative composition. Although the components of the alternative composition can perform a mutual flow, still, choice between them is resolved at the start of delay. The result of delaying for any duration t must be either one flow clause or the other and not both.

Remarks

HyPA is a conservative extension of ACP [BW90] and models from control theory. In HyPA there is no consistency concept to express and enforce the assumptions of a hybrid process about the environment as present in Hybrid χ and ACP_{hs}^{srt}. Another missing feature is that time determinism (postponing the resolution of choice, when a system is delaying) cannot be modelled in HyPA. That makes HyPA less suitable for pure timed applications without any physical continuous behaviour. HyPA models can be created that only have continuous flows without any algebraic actions in between which is sometimes useful in a control theoretic setting. HyPA is provided with a large set of axioms by which a HyPA specification can be rewritten into a simpler bisimilar form. We find HyPA a relatively simple process algebra with all the essential constructs for describing hybrid behaviour of processes.

2.2.2 ϕ-Calculus

Another process algebra for specifying and modelling hybrid systems is ϕ-calculus.

ϕ-calculus is an extension of π-calculus. π-calculus (see [Mil99]) is a process algebra to model *reconfigurable* systems. A reconfigurable system is one in which the sub-processes keep on making new connections among themselves. In a π-calculus system specification, a set of link-names is given. These link names serve as channels of communication between sub-processes. Basic actions in $\pi-$calculus are send and receive on a link. Messages sent and received comprise of values and link-names. Passing of a link-name to a process gives it access to other processes using the same link name. This changes the configuration of the system. The following example clarifies the point.

Example 1 *Let P and Q be two π-calculus processes defined as follows:*

$$\begin{array}{rcl} P & = & \nu a\bar{b}\langle a\rangle.P' \\ Q & = & b(x).Q' \end{array}$$

Let 'b' be a global link name. The process P declares a private link 'a', which is represented by νa. It then passes the link name 'a' on link 'b'. The process Q reads a message from link b, into a variable x. When P and Q run in parallel, the link name 'a' which is private to process P is transferred to process Q. In this way a new link of communication is opened between P and Q.

Hence, a system that changes its configuration can easily be modelled in π-calculus. In ϕ-calculus, in addition to link names, a set of environment names is given. In ϕ-calculus, messages exchanged on links can also consist of environment names besides values and link names.

In this section, we refer to [RS03] and mention the changes in a later improved version which has not been published.

The set P of ϕ-calculus process terms can be defined by the BNF expression in Table 2.2. For the BNF definition we need the following: a set of environment variable names X and their derivatives $\dot{X}$; the symbols x, y, z vary over environment variable names; the symbols a, b, c vary over link names; $\vec{x}, \vec{y}, \vec{a}, \vec{b}$ represent vectors of names; δ is a special action prefix with no corresponding $\bar{\delta}$. The action prefix δ is only present in the improved version of ϕ-calculus in order to allow arbitrary delays before a communication.

Table 2.2: Syntax of ϕ-Calculus

$P ::=$	0	Null process.
	$\nu x P$	Declares a private environment variable x for P.
	$\nu a P$	Declares a private link a for P.
	$P \mid P$	Parallel Communication
	$!P$	Replication. $!P \equiv !P \mid P \equiv !P \mid P \mid P \ldots$
	S	
$S ::=$	$\tau.P$	Silent action prefix
	$\delta.P$	Delay action prefix.
	aF	a-receive on link a and continue as abstraction F.
	$\bar{a}C$	$\bar{a}$-send on link a and continue as concretion C.
	$(S + S)$	Sums
	$[\ \gamma \rightarrow \vec{x} := \vec{e}\ ]\ .P$	Assignment action prefix
	$[\ \gamma \rightarrow \langle\text{resetlist}\rangle$ with $\langle\text{clauselist}\rangle\ ]\ .P$	Reset action prefix
$C ::=$	$(\nu\vec{y})\langle\vec{x}\rangle P \mid (\nu\vec{b})\langle\vec{a}\rangle P$ where $\vec{b} \subseteq \vec{a}, \vec{y} \subseteq \vec{x}$	Concretions
$F ::=$	$(\vec{x})P \mid (\vec{a})P$	Abstractions
$A ::=$	$F \mid C$	Agents

A concretion is a process term which is preceded by a send action that does not mention the link name on which message is sent. Examples of concretions are $\langle\vec{x}\rangle\,.\,P$, $\langle\vec{b}\rangle\,.\,P$ or simply P. In the first case a vector $\vec{x}$ of environment variables is sent. In the second case a vector $\vec{b}$ of link names is sent as a message. In the third case, the concretion P indicates that no message is sent during communication. Private links or variables can also be sent in messages. The concretion $\nu\,\vec{y}\langle\vec{x}\rangle\,P$, where $\vec{y} \subseteq \vec{x}$, indicates that among the $\vec{x}$ variables sent, the variables in vector $\vec{y}$ are private to the process term P.

Similarly an abstraction is a process which is preceded by a receive action that does not mention the link name on which message is received. Examples of abstractions are $(\vec{y})\,.\,P$, $(\vec{b})\,.\,P$ or simply P. Passing of a private link or variable in a message increases the scope of that link or variable.

Concretions and abstractions are called *agents*. They were introduced in π-calculus by

Milner (see [Mil99]) in order to make the semantics easier. With the help of agents, the labels of the send or receive transitions can consist of just the link name and the details of the message exchanged are left with the resulting agents.

Special features of ϕ-calculus to represent hybrid processes are as follows:

- **Data:** The data part in ϕ-calculus is called an *environment.* A ϕ-calculus process is a pair (E, P), where E is the environment and P is the ϕ-calculus process term. An environment consists of the valuation of environment variables and a set of flow constraints on variables and their derivatives. The variables and their derivatives must evolve according to these constraints as the process (E, P) idles.

- **Environmental actions:** Modifications to the environment are carried by means of *environmental* actions. There are two types of environment actions:

i. **assignments**-that instantaneously modify the variable valuation. For example,

$$[\ \gamma \rightarrow \vec{x} := \vec{e}\ ]$$

If the predicate γ is true, assign (a vector of values) $\vec{e}$ to (a vector of variables) $\vec{x}$.

ii. **resets**-that modify the set of flow constraints in the environment and thus modify the continuous behaviour of the whole system.

$$[\ \gamma \rightarrow \langle\text{resetlist}\rangle \text{ with } \langle\text{clauselist}\rangle\ ]\ .P$$

The symbol γ is a predicate. The list $\langle$resetlist$\rangle$ is a list of environment variables whose flow constraints need to be modified. The list $\langle$clauselist$\rangle$ consists of a new set of flow constraints that will replace the old flow constraints of the reset variables.

For example, consider the reset action

$$[\ l \geq 10 \rightarrow \text{reset } \dot{l} \text{ with } \{\dot{l} \mid \dot{l} = 0\}\ ]$$

The predicate $l \geq 10$ is the guard of the reset action. The list $\dot{l}$ is the list of variables whose constraints will be updated and $\{\dot{l} \mid \dot{l} = 0\}$ is a list of new constraints.

- **Delay Behaviour:** The action prefixes, $\bar{a}$ (send action), a (receive action), τ (communication action), δ (the delay action) and environmental actions (assignments and resets) with false guards are all delayable. They can delay as long as the environment associated with the process term they are prefixing can delay. For example, consider the process given below: (Recall that a $\phi-$calculus process is a pair (E, P)).

$$(\left(\begin{array}{c} x : 0 \\ \{\dot{x} \mid \dot{x} = 1\}, \\ \{x \mid x \leq 100\} \end{array}\right),\ \bar{a}\,.Q')$$

The above process can delay for no more than 100 time units according to the constraints given in the environment.

$$(\left(\begin{array}{c} x : 0 \\ \{\dot{x} \mid \dot{x} = 1\}, \\ \{x \mid x \leq 100\} \end{array}\right),\ \ [\ x = 50\ ]\ .Q')$$

The symbol $[\ x = 50\ ]$ denotes an environmental action with predicate $x = 50$ and no assignment to a variable. The above process must delay for 50 time units till the predicate $(x = 50)$

becomes true. Because of the guarded action, the above process cannot proceed as Q' before x becomes 50. As soon as the predicate becomes true, the action $[\ x = 50\ ]$ is discharged and it proceeds as process Q'.

In [RS03], the flow of variables is given by differential equations that are valuations of derivatives ($\dot{X}$) to the set of continuously differentiable functions. As a result non-determinism (as in differential inclusions) in variable flows was not allowed. In the improved version of ϕ-calculus, this restriction has been relaxed and the flow of environment variables are given by a set of flow constraints on environment variables containing algebraic equations (inequalities) and differential equations (inclusions). In the improved version, the allowed trajectories of variables are again continuously differentiable functions of time.

Environmental actions with true guards are urgent. We call environmental actions with true guards *enabled actions*. If an enabled environmental action appears after a send or receive action on a link that has a possibility of communicating with its counterpart in parallel, then the effect of such an action is that it makes communication urgent on that link. Consider the following process,

$$(\begin{pmatrix} t : 0 \\ \{\dot{t} \mid \dot{t} = 2\} \end{pmatrix}, \bar{a}\,.\,[t \geq 10 \rightarrow x := 0] \mid a)$$

The communication on link a can delay for 5 seconds. After 5 seconds the guard ($t \geq 10$) becomes true and the communication (resulting in a silent action) must take place. Making communication urgent in this way is required to preserve weak bisimulation between processes as explained further on, while we discuss bisimulation for ϕ-Calculus processes.

It can be desirable in certain cases to be able to express an arbitrary delay before a communication takes place. An arbitrary delay before a communication on a link is added by prefixing the delay action δ before one of the send or receive actions on the link. The delay action prefix is arbitrarily delayable. For Timed CCS (see [MT90]), a delay prefix (also denoted by δ) has been defined that adds arbitrary delays before a process. The idea behind the delay prefix in ϕ-calculus is similar.

- **Representation of passage of time:** The following process term specifies a delay of 10 time units before continuing as the process term P.

$$\begin{array}{l} \nu\, t\,(\ [\ t := 0\ ]\ .\ [\ \text{ reset } t, \dot{t} \text{ with} \\ \{t \mid t \leq 10\}, \{\dot{t} \mid \dot{t} = 1\}\ ]\ .\ [\ t = 10\ ]\ .P) \end{array}$$

The guard $[t = 10]$ placed before the process P ensures that P cannot be executed until the guard becomes true. When the guard becomes true, the environmental action $[\ t = 10\ ]$ (which has an empty assignment or an empty reset list) is executed immediately. The flow constraint $\{t \mid t \leq 10\}$ does not allow the environment to delay any longer. The process P must perform an environmental action immediately in order to allow the environment to flow.

- **Variable Abstraction:** A variable z local to a process is declared by $\nu z P$.

- **Semantics:** The behaviour of a process (E, P) is described by four kinds of transitions. They are as follows:

i. a π-action transition(sending and receiving actions on a link or the communication action τ);

ii. an environmental action transition (the assignment or reset actions);
iii. a delay action transition (a transition that discharges the prefix δ) and;
iv. a flow transition representing a delay.

The execution of a π action or a delay action δ simply removes the action prefix in front of a process P and does not affect the accompanying environment.

An environmental action is an assignment to a variable or a reset of a flow constraint. Its execution modifies both the environment and the process term.

$$\begin{array}{ll} & (\left(\begin{array}{c} z : 10 \\ \emptyset \end{array} \right) , \ [\ z' := 60\ ]\ .\bar{c}\langle z' \rangle .0\ ) \\ \xrightarrow{[z':=60]} & (\left(\begin{array}{c} z : 10, z' : 60 \\ \emptyset \end{array} \right) , \ \bar{c}\langle z' \rangle .0\ ) \end{array}$$

A process term P in an environment E, can delay as long as the flow constraints in E are satisfied and P is delayable. A process term P is delayable till no guard of an environmental action prefixing P (or prefixing P after a series of silent actions) is true. A flow transition modifies only the environment and not the process term. A flow transition is of the form

$$(E\ ,\ P) \xrightarrow{x[0,t)} (E'\ ,\ P),$$

where $x[0,t)$ describes the evolution of the variables during delay. t indicates the duration of the delay. $x[0,t)$ must be a continuously differentiable function of time. In [RS03], the function $x[0,t)$ describes the evolution of only the variables that are in the valuation of E. In [RS], the authors allow the function $x[0,t)$ to describe the evolution of control variables that are not present in the valuation of E.

- **Time determinism:** The alternative composition operator of $\phi-$calculus is strong time deterministic. The alternative composition of two or more process terms can delay as long as all the process terms and the given environment can delay. When one of the process terms cannot delay any further, choice is resolved in favor of an action of one of the component process terms.

In an alternative composition in ϕ-Calculus, the question of agreement on variable evolution by alternatives does not arise as it arises in other process algebras. This is because flow clauses are not a part of process terms, but a part of the environment associated with the process terms. For alternative composition, the individual delay behaviour of component process terms in the same environment is considered. Therefore a conflict due to different flow clauses cannot arise here or in parallel composition (see section 2.5) as in Hybrid χ or ACP^{srt}_{hs}. However the duration of the delay of alternative composition is effected by its component process terms.

Consider the following process:

$$(\left(\begin{array}{c} x : 0 \\ \{\{\dot{x} \mid \dot{x} \geq 0\} \\ \{x \mid x \leq 50\}\} \end{array} \right) , \ (\ (\bar{a}\langle x \rangle .\ [\ x \geq 30\ ]\ .P)\ \mid\ (a(y).0\ +\ b(y).Q)\ ),$$

where P and Q are two ϕ-calculus process terms. The above process can delay as long as the value of x remains less than 30. When $x = 30$, the action a in $(a(y).0\ +\ b(y).Q)$

becomes urgent. Then the alternative composition cannot delay further. The choice in $(a(y).0 + b(y).Q)$ is resolved in favor of any of the two process terms depending upon the first action performed.

- **Bisimulation:** Two kinds of bisimulation, viz weak bisimulation and strong are defined. In strong bisimulation, two processes are related if they can mimic each others transitions. In weak bisimulation, two processes are related if they can mimic each others transitions and their transitions are matched while abstracting from all τ transitions. See [BW90, BBR09], for more on weak bisimulation.

Bisimulation is defined in two steps. In the first step, two ϕ-calculus process terms are compared with respect to discrete behaviour. Action prefixes and environmental actions are considered in this step. Environmental actions are considered to be able to do transitions like other action prefixes without regard to the environment. In the second step, two theorems are introduced, one for strong discrete bisimulation and one for weak discrete bisimulation. The theorems state that two strongly discretely bisimilar processes (or weakly bisimilar processes) placed in an environment E mimic each other's flow behaviour.

As mentioned before, (in the *Delay Behaviour* of a ϕ-calculus process), a τ action preceding an environmental action with a true guard is made urgent to preserve weak bisimulation. This is explained below. Consider the following example:

$$P = \nu a(a.\ [\ x := 1\ ]\ ||\bar{a}) \text{ and } Q =\ [\ x := 1\ ]$$

In P, ν declares a private link a. a is a receiving action on link a. $\bar{a}$ represents a sending action on link a. In this example of sending and receiving no messages are being exchanged. $[\ x := 1\ ]$ is an assignment with a **true** guard. We can see that process terms P and Q are weakly bisimilar according to the first step. That is they are weakly bisimilar with respect to action behaviour. Consider an environment E:

$$\left(\begin{array}{c} x : 0 \\ \{\{x \mid \dot{x} = 1\}\} \end{array} \right).$$

Process Q cannot wait as it has an environmental action with a true guard. If the τ action preceding the assignment $[\ x := 1\ ]$ is not urgent then process P can wait. Therefore the delay behaviour of two processes that are weakly bisimilar with respect to actions, can differ if τ actions are unconditionally delayable.

In [RS03], this requirement of urgent communication to preserve weak bisimulation has not been covered at all in the definition of the replication operator. The rule for delay of a replication operator is given below:

$$\frac{(E, P) \xrightarrow{x[0,t)} (E', P)}{(E, !P) \xrightarrow{x[0,t)} (E', !P)}$$

Consider a process P defined as:

$$P = \bar{a}\langle 1\rangle.[x := 2].0 + a(x)$$

Process P is delayable as both alternatives are delayable. Now the operator $!P \equiv P|!P$. But the process $P|!P$ is not delayable according to the Par rule of [RS03]. This error has been corrected in [RS].

There are no axioms available for equational reasoning in ϕ-calculus.

Remarks

We find that in a ϕ-calculus process, since flows are part of the environment, the interaction between the variable evolutions of component processes does not exist as present in other process algebras.

There are no consistency predicates defined in ϕ-calculus. The flow clauses defined in a ϕ-calculus process environment act like flows of hybrid automata. Regarding update of environment variables, in ϕ-calculus, an environment variable can only be initialized to a single value by an assignment and it cannot be initialized according to a condition as in other process algebras, (for example, the reinitilizations in HyPA reset a variable according to a predicate). ϕ-Calculus does not have any axioms for equational reasoning.

An advantage of ϕ-calculus is that being an extension of π-calculus representing reconfigurable hybrid process is easy. Also we find the semantics of ϕ-calculus simple as compared to those of Hybrid χ and ACP^{srt}_{hs}.

2.2.3 Hybrid χ

Hybrid χ is another process algebra for modelling hybrid systems. Hybrid χ is an extension of the language χ, a modelling and simulation language for industrial systems [MFR95, MFRvdN95]. The original χ language was a discrete event language. The language χ was later extended with hybrid constructs (see [Fab99, BR00]) and was given a formal semantics (see [BMR$^+$06]).

The set of process terms P can be defined with the help of a BNF expression given in Table 2.3.

Special features of Hybrid χ are as follows:

- **Data:** The environment variables in Hybrid χ have been divided into different categories depending on their behaviour during delays or action execution.

During delays the following five categories of variables can be recognized:

i. **Discrete (D)**: Their values remain constant as a Hybrid χ process delays.

ii. **Continuous (C):** Their values vary over time with trajectories that are absolutely continuous functions of time.

iii. **Dotted Continuous ($\dot{C}$):** They are the derivatives of continuous variables. Their trajectories can contain discontinuities.

iv. **Algebraic (L):** They can also vary over time with a possibly discontinuous trajectory.

v. **A special variable 'time':** It represents time passed since the beginning of the process.

Continuous and Discrete variables are further divided into *jumping* (**J**) or *non-jumping* variables depending on whether their values can change arbitrarily when the system performs an action or not.

A Hybrid χ process is a triple $\langle p, \sigma, E\rangle$, where, p is a Hybrid χ process term; σ is a variable valuation; and E is the Hybrid χ environment. The **variable valuation** consists of the valuations of discrete, continuous variables and of the special variable time.

Environment (E) of a Hybrid χ process consists of the following information:

i. information about the categorization of variables, i.e. to which category an environment variable belongs;

Table 2.3: Syntax of Hybrid χ 1.0 [BMR+06]

$p ::=$	$W : r \gg l_a$	action predicate	W r l_a	set of variables jump predicate action label
	$\mid u$	delay predicate	u	predicate
	$\mid \delta$	Deadlock		
	$\mid \perp$	Inconsistent process		
	$\mid [p]$	any delay		
	$\mid u \curvearrowright p$	signal emission	u	predicate
	$\mid p;\ p$	sequential composition		
	$\mid b \rightarrow p$	guard	b	predicate
	$\mid p \,[\!]\, p$	alternative composition		
	$\mid p \parallel p$	parallel composition		
	$\mid h!!\mathbf{e}_n$	send process term	h $\mathbf{e}_n$	channel expression vector
	$\mid h??\mathbf{x}_n$	receive process term	h $\mathbf{x}_n$	channel variable vector
	$\mid \partial_A(p)$	action encapsulation	A	Set of action labels
	$\mid \upsilon_H(p)$	urgent communication	H	Set of channels
	$\mid X$	recursion variable		
	$\mid \iota_{J^+}(p)$	jump enabling	J^+	set of variables
	$\mid \,\vert[\ {}_V\sigma_\perp, C, L`\vert'p\ ]\vert$	variable scope	V $\sigma_\perp$	set of variables valuation
	$\mid \,\vert[\ {}_H H`\vert'p\ ]\vert$	channel scope	H	set of channels
	$\mid \,\vert[\ {}_R R`\vert'p\ ]\vert$	recursion scope	R	set of recursive definitions.
	$\mid p_{ext}$	syntactic extensions		

ii. a set of channel names; and
iii. a definition of recursive variables

Formally an environment is a five tuple, i.e. (C, J, H, L, R), where

1. **C** is the set of continuous environment variables
2. **J** is the set of environment variables declared as *jumping*
3. **L** is the set of algebraic variables
4. **H** is a set of channel names
5. **R** is a set of recursive definitions.

R is of the form $\{X_1 \mapsto P_1, \ldots, X_n \mapsto P_n\}$, where $X_1, \ldots, X_n$ are recursive variables and $P_1, \ldots, P_n$ are Hybrid χ process terms.

- **Action predicates:** Action predicates enable instantaneous changes to variable values. An action predicate is of the form $W : r \gg l_a$, where W is a set of non-jumping variables, r is a predicate and l_a denotes the label a of the action. The predicate r is defined in terms of current and new values of variables. The current values or values before the execution of action are denoted by $variable^-$, a variable name with a '$-'$ superscript and the new values or the values after the execution of action are denoted by plain variable names. An example of an action predicate is $\{x\} : x^- \geq 100 \wedge x \leq -1400 \gg \tau$.

- **Guards:** Hybrid χ has a guard operator that can place conditionals before any Hybrid χ process term. A guarded process term is of the form $b \rightarrow p$, where b is a predicate on environment variables.

The guards in Hybrid χ are delayable. A false guard can delay according to any delay predicate (provided that the flow requirements of variables of different categories are met), till the guard becomes true.

- **Delay predicates:** The delay behaviors of environment variables are determined by their categories and can be further restricted by constructs called delay predicates. Delay predicates can be differential equations, differential inclusions or algebraic equations. The set of possible solutions of delay predicates is a parameter to the theory of Hybrid χ and can be adapted according to the problem at hand.

For most systems, continuous functions are taken for continuous variables. Continuous variables are never allowed to jump during a delay (as they can in HyPA at the start of a new flow clause). The delay predicates model non-terminating behaviour. There is no disrupt operator (as defined in HyPA) in Hybrid χ . In Hybrid χ , a delay predicate often appears in alternative composition with other χ constructs that determine when the delay must end. An alternative composition is strongly time deterministic. As soon as a delay predicate cannot delay or an action or a guard alternatively composed with a delay predicate becomes urgent, the system stops delaying and the undelayable action is performed. Consider the following process:

$$\begin{array}{l}\langle \dot{x} = 52 \ [] \ x = 0 \rightarrow \{\} : \text{true} >> \text{pass}, \\ \{x \mapsto -1400, \text{time} \mapsto 0\}, \\ (\{x\}, \emptyset, \emptyset, \emptyset, \emptyset)\rangle\end{array}$$

A continuous variable x is defined and is initialized to -1400. The guard $x = 0$ is false initially. The given process delays according to the differential equation $\dot{x} = 52$. The variable

x becomes 0 after 26.923 seconds. At time 26.923, the process cannot delay and action *pass* takes place immediately.

- **Specifying assumptions about the environment:** In Hybrid χ , assumptions about the environment can be specified by a signal emission operator or by delay predicates.

i. **Signal Emission:** The signal emission operator (denoted by $\curvearrowright$), indicates assumptions about the environment variables at any stage in a process term. It has been inspired from ACP_{ps} (see [BB97]). For example,

$$(\texttt{current} \leq 0) \curvearrowright P,$$

is a process term with an initial state ($\texttt{current} \leq 0$).
ii. **Delay Predicates:** Delay predicates act as invariants. They represent conditions about environment variables that must prevail over a certain period of time during which the process delays.

The semantics of Hybrid χ is such that all actions and time transitions must satisfy a process's assumptions about the environment. If the assumptions cannot be satisfied, the process deadlocks.

- **Time determinism:** The alternative composition operator of Hybrid χ is strongly time deterministic. A consequence of this choice is that the deadlock process is not a neutral element for alternative composition. I.e. the axiom $\delta + x = x$ does not hold in Hybrid χ .

- **Scoping operators:** Scoping operators enable definitions of environment variables, channels and recursive variables whose scope is limited to a process term. Three kinds of scope operator process terms are defined. They are:

i. Variable scope operator process term;
ii. Channel scope operator process term; and
iii. Recursion Scope operator process term

Other process terms in composition with a scope operator process term cannot access or modify any of its local channels or local variables. Unlike ϕ-calculus, the scope of a local variable or channel cannot be widened, as only values not names are exchanged in a Hybrid χ communication.

- **Syntactic extensions:** A user friendly syntax has been introduced for Hybrid χ process terms by means of syntactic extensions. These syntactic extensions do not increase expressiveness of Hybrid χ but improve the readability of Hybrid χ specifications. We find the delay operator (denoted by $\triangle d$) and process instantiations to be particularly useful syntactic extensions. Process instantiations and process definitions enable re-use of process terms. In a process instantiation, parameters can be passed to a process definition and the same process definition can be re-used for defining different processes.

For a discussion of other operators see [BMR$^+$06].

- **Semantics:** The operational semantics associates a hybrid transition system with every process. Four kinds of transition relations are described. They are action transitions; termination transitions; time transitions and consistency predicates.

The consistency predicates provide the semantic support for syntactical constructs representing assumptions on the environment variables. The consistency predicate enforce restrictions on the behaviour of a process like that of invariants and flows (combined) in a hybrid automaton.

- **Bisimulation:** In Hybrid χ , state-less bisimilarity is chosen for determining equivalence of process terms.

A set of axioms representing equivalence between process terms is also given. These axioms are sound with respect to state-less bisimilarity. The set of axioms is not comprehensive enough to allow algebraic manipulation as in HyPA or ACP_{hs}^{srt}.

Remarks

Hybrid χ is provided with a large number of operators. The scoping and process instantiation operators allow modular design of processes. The semantics of Hybrid χ is complex with a lot of detail (like extended valuations) appearing on the labels for action transitions. The semantics of the guard operator is cumbersome. Hybrid χ has a small set of axioms for equational reasoning. The recursive specification and definition theorems necessary for analysis of recursive specifications has not been proven for Hybrid χ. Translations from Hybrid χ to hybrid automata, see[BJM$^+$03], and a number of control theory formalisms for describing hybrid systems, see [BRS$^+$07], have been defined.

As mentioned in the introduction to the chapter, now a second version of Hybrid χ is available [BHR$^+$08]. The development of Hybrid χ 2.0 is motivated by the following factors:

1. To make the semantics of Hybrid χ simpler.
2. To be closer to hybrid automata.
3. To be able to model correctly a wider variety of industrial systems. The Hybrid χ 2.0 has constructs to resolve hidden constraints that can be present in a multi-variable system. For example, Hybrid χ 2.0 identifies the systems P and Q defined by the following sets of predicates as bisimilar:

$$\begin{array}{llcl} P: & x = 1 & \text{and } Q: & x = 1 \\ & y = \dot{x} & & y = 0 \end{array}$$

For more about Hybrid χ 2.0, see [BHR$^+$08].

2.2.4 ACP_{hs}^{srt}

ACP_{hs}^{srt} is an extension of ACP with standard relative timing (denoted by ACP^{srt}, see [BM02a]) and ACP with propositional signals (denoted by ACP_{ps}, see [BB97]).

The set of ACP_{hs}^{srt} process terms can be described by the BNF expression given in Table 2.4.

In many ways HyPA and ACP_{hs}^{srt} are similar. Both are extensions of ACP. Both have a large set of axioms and two notions of bisimulation, i.e., stateless and initially stateless bisimulations. In this section we summarize what ACP_{hs}^{srt} has to offer.

- **Data:** The data part of an ACP_{hs}^{srt} process is simply a valuation of environment variables. During actions, by default all variables can jump to arbitrary values. This is unlike HyPA, where the values remain the same.

Table 2.4: Syntax of ACP_{hs}^{srt}

$P ::=$	$\tilde{\tilde{a}}$	discrete, undelayable action	$a \in A$ a set of actions
	$\mid \tilde{\tilde{\delta}}$	Deadlock	
	$\mid \perp$	Inconsistent process	
	$\mid \nu_{rel}(P)$	relative timeout	
	$\mid \sigma_{\mathsf{rel}}^{r}(P)$	relative delay	$r \geq 0$
	$\mid \psi \ {}^{\wedge}\!\blacktriangle\ P$	signal emission	$\psi -$ a state proposition
	$\mid \psi :\rightarrow P$	conditional	$\psi -$ a state proposition
	$\mid \phi \curvearrowright_{V} P$	signal evolution	$\phi -$ a state proposition
			$V -$ a set of model variables
	$\mid \chi \ulcorner\!\blacktriangledown P$	signal transition	$\chi -$ a transition proposition
	$\mid P \cdot P$	sequential composition	
	$\mid P + P$	alternative composition	
	$\mid P \parallel P$	parallel composition	
	$\mid P \mid P$	forced communication	
	$\mid P \parallel\!\!\!\lfloor P$	left parallel operator	
	$\mid \partial_{H}(P)$	encapsulation operator	$H \subseteq A$

- **Delay Operator and Time-out Operator:** A relative delay between two process terms is described via a delay operator, denoted by $\sigma^{r}_{\mathsf{rel}}$, where r represents the duration of the delay. Corresponding to the relative delay operator $\sigma^{r}_{\mathsf{rel}}$, Hybrid χ has a syntactic extension $\triangle r$.

To enforce a process term to perform an action at its start, a relative time-out operator, denoted by υ_{rel}, is used. If the only starting option available for a process term is to delay then the process term deadlocks.

The relative delay and relative time-out operators are operators of ACP with standard relative timing (ACP^{srt}). The concept of immediate actions and immediate deadlock, represented by action name with a superscript $\tilde{\tilde{}}$, are also taken from ACP^{srt}.

- **Signal Transition Operator:** Discrete changes in the environment variables are modelled by means of a signal transition operator (denoted by $\ulcorner\!\blacktriangledown$). The signal transition operator is similar to the renitializations of HyPA and action predicates in Hybrid χ. A difference is that the reinitializations of HyPA are effective in both cases whether used with actions or flows. On the other hand, the signal transition operator is effective (in changing the variable values) only when used with an action. The jump predicate of a signal transition operator is known as a transition proposition. The values of variables before a transition are represented by $^{\bullet}$`variable` and the values of variable after a transition are represented by `variable`$^{\bullet}$. For example, a process term $(x^{\bullet} = {}^{\bullet}x) \ulcorner\!\blacktriangledown \tilde{\tilde{a}}$, represents an immediate action a with no change in variable x. Note that environment variables in ACP^{srt}_{hs} are by default jumping when actions are performed.

- **Signal Evolution Operator:** A signal evolution operator (denoted by $\ulcorner\!\blacktriangledown_{V}$) is used to describe the evolution of variables during delays. The trajectories of environment variables are piece-wise continuously differentiable, real valued functions of time. The evolution operator takes a predicate, called an evolution proposition that restricts the behaviour of variables. An evolution proposition can be a differential equation, differential inclusion, algebraic equation, algebraic inequality or a boolean expression. The evolution operator can also declare a set of variables to be smooth. The trajectory of a variable that is declared smooth must be continuously differentiable. For example, in $(x \leq 50 \wedge 20 \leq v \leq 100) \ulcorner\!\blacktriangledown_{\{v\}} \sigma^{5}_{\mathsf{rel}}(p)$, the value of variables v and $\dot{v}$ must vary continuously, whereas the variable x and its derivative can jump finitely many times during the delay.

- **Guards:** A guard operator is used to represent conditional actions or delays. The guards in ACP^{srt}_{hs} are not delayable as defined in Hybrid χ or ϕ-Calculus. If a guard is false, the guarded process deadlocks. Delayable guards can be modelled in ACP^{srt}_{hs} (as well as in HyPA) .

- **Representing environment invariants:** ACP^{srt}_{hs}, like Hybrid χ , has a signal emission operator from ACP_{ps}. Note that the symbols representing signal emission are different in ACP^{srt}_{hs} and in Hybrid χ. In ACP^{srt}_{hs}, it is denoted (as in ACP_{ps}) by $\wedge\!\!\blacktriangle$. The evolution proposition of ACP^{srt}_{hs}, like delay predicates of Hybrid χ , represents system invariants. The semantics of ACP^{srt}_{hs} ensures that assumptions about the environment described by signal emissions or evolution propositions always hold.

- **Integration:** ACP^{srt}_{hs} is extended with integration over time intervals. The intuition behind integration is to be able to model processes that have the capability of performing an action at any point in a given time interval. Integration represents a choice over a set (that can be infinite) of process terms. The notation σ^*_{rel} is an abbreviation for integration over the interval $[0, \infty)$, i.e. σ^*_{rel} represents an indefinite delay.

Integration over time intervals can be expressed in other process algebras also. For example, $\int_{r\in[2,4]}\sigma^r_{\mathsf{rel}}(P)$ can be expressed as follows:

$$\text{HyPA:} \quad [t \mid t^+ = 0] \gg \left(t \,\middle|\, \begin{array}{l} \dot{t} = 1 \\ t \le 4 \end{array} \right) \rhd [t^- \ge 2] \gg P$$

$$\begin{array}{rl} \phi\text{-Calculus:} & \nu t[t := 0].[\text{ reset } t, \dot{t} \text{ with } \{\dot{t} \mid \dot{t} = 1\}, \\ & \{t \mid t \le 4\}].[t \ge 2].P \end{array}$$

$$\text{Hybrid } \chi\text{:} \quad \{n\} : n \in [2,4] \gg \tau;\ \triangle n;\ P$$

- **Variable Abstraction:** A number of attempts have been made to define a variable abstraction operator for ACP^{srt}_{hs} (see [Kha05]). The preference of preserving flow determinism after variable abstraction makes the definition of a variable abstraction operator in structural operational semantics difficult (see Section 2.3).

In [Kha05], a graph semantics is given to ACP^{srt}_{hs} and a variable abstraction operator is defined in the graph model.

- **Semantics:** In the hybrid transition system of ACP^{srt}_{hs}, five types of transition relations are defined. They are: the action step; the time step; the termination step; the signal relations and the discontinuity relations.

The signal of a process term is actually a proposition that states the assumptions of the process term t about the environment variables.This concept has been inspired from the root signal operator of ACP_{ps} [BB97]. The signal relations in the semantics of ACP^{srt}_{hs} are like the consistency relations of Hybrid χ .

In ACP^{srt}_{hs}, the continuity requirements of a **delayable** process term in parallel composition with another are protected. For example, the discontinuities caused by an action transition from a state with valuation $\{x \mapsto 20, \dot{x} \mapsto 1\}$ to a state with valuation $\{x \mapsto 25, \dot{x} \mapsto 1\}$ are not allowed by the process term

$$(x \le 30) \curvearrowright_{\{x\}} \sigma^5_{\mathsf{rel}}(P),$$

as variable x is declared continuous by it but are allowed by the process term,

$$(x \le 30) \curvearrowright_{\emptyset} \sigma^5_{\mathsf{rel}}(P).$$

In order to find out whether a given transition respects the continuity requirements of a process, discontinuity relations are defined.

- **Weak Time determinism:** ACP^{srt}_{hs} takes a weak time deterministic approach in alternative composition. An alternative composition can delay without resolving choices between process terms, when all its components can delay according to a common trajectory. An alternative composition can also delay for a certain duration if one alternative cannot delay (with any variable evolution) for that duration, while others can delay with a common trajectory for that duration. In the latter case, a delay is said to resolve choices.

Consider the following process term:

$$(\dot{x} = 1) \curvearrowright_{\{x\}} \sigma^{5}_{\mathsf{rel}}(\tilde{\tilde{a}}) + (\dot{x} \geq 0) \curvearrowright_{x} \sigma^{10}_{\mathsf{rel}}(\tilde{\tilde{b}})$$

In a starting valuation $\{x \mapsto 0, \dot{x} \mapsto 1\}$, it can delay for 5 time units with trajectory $(\dot{x} = 1)$, without resolving the choice. It can also delay when one of the process terms cannot delay for a certain duration with **any** evolution and the other can delay for that duration. I.e., the given alternative composition can evolve for 6 time units with evolution $\dot{x} = t + 1$, resolving the choice in favor of $(\dot{x} \geq 0) \curvearrowright_{x} \sigma^{10}_{\mathsf{rel}}(\tilde{\tilde{b}})$. Note that t represents time since the start of delay. The expression $\dot{x} = t + 1$ agrees with $\dot{x} = 1$ at time $t = 0$, but diverges as the time progresses. The signal of the given alternative composition is true at time 0.

After 6 time units the alternative composition evolves into the following term:

$$((\dot{x} \geq 0) \curvearrowright_{\{x\}} \sigma^{4}_{\mathsf{rel}}(\tilde{\tilde{b}}))$$

A property of process algebras with timing, known as *time-interpolation* of processes, states that a time transition can always be broken into smaller time transitions.

Mathematically, time interpolation can be expressed as follows,

If s, s' and s'' are process terms and m, m_1, m_2 denote time durations,

$$\begin{aligned} & s \xmapsto{m} s' \implies \exists(s'', m_1 \text{ and } m_2), \text{ such that} \\ & m = m_1 + m_2 \text{ and } s \xmapsto{m_1} s'' \text{ and } s'' \xmapsto{m_2} s' \end{aligned}$$

In the semantics of ACP^{srt}_{hs}, the property of time interpolation does not always hold. Consider the time transition mentioned last of the alternative composition $(\dot{x} = 1) \curvearrowright_{\{x\}} \sigma^{5}_{\mathsf{rel}}(\tilde{\tilde{a}}) + (\dot{x} \geq 0) \curvearrowright_{x} \sigma^{10}_{\mathsf{rel}}(\tilde{\tilde{b}})$.

The time transition,

$$\begin{aligned} & \langle(\dot{x} = 1) \curvearrowright_{x} \sigma^{5}_{\mathsf{rel}}(\tilde{\tilde{a}}) + (\dot{x} \geq 0) \curvearrowright_{\{x\}} \sigma^{10}_{\mathsf{rel}}(\tilde{\tilde{b}}), \{x \mapsto 0\}\rangle \\ & \xmapsto{6, \{\dot{x} = t+1\}} \langle((\dot{x} \geq 0) \curvearrowright_{x} \sigma^{4}_{\mathsf{rel}}(\tilde{\tilde{b}})), \{x \mapsto (t^2/2 + t)\}\rangle, \end{aligned}$$

that is allowed by the semantics of ACP^{srt}_{hs}, cannot be broken into durations smaller than 5.

- **Bisimulation:** Two kinds of bisimulation relations on process terms are defined.

i. A state-based bisimulation simply called *bisimulation.* The bisimulation is not preserved by parallel operator just initially state-less bisimulation is not preserved by the parallel operator in HyPA.
ii. A state-less bisimulation called interference-compatible bisimulation in [BM05], abbreviated as *ic-bisimulation.* Ic-bisimulation (denoted by $\underline{\leftrightarrow}$), is comparable to the robust bisimilarity of HyPA and the state-less bisimilarity of Hybrid χ. Ic-bisimulation is preserved by the parallel operator.

ACP^{srt}_{hs} is provided with a large set of axioms. As mentioned previously, the choice operator is non-associative in ACP^{srt}_{hs}, the axiom for time determinism (SRT3) does not hold in the current semantics, and a few less important axioms (HSE7, HSE13, HSSRCM, INT10SR, INT11, CM4) are unsound. For a discussion towards correcting these errors, please see Chapter 3.

Like HyPA, ACP^{srt}_{hs} has some axioms and lifting rules for drawing conclusions about environment variables that can only be used in the absence of a parallel operator.

All process terms of ACP^{srt}_{hs} can be reduced to a basic form without the parallel operator. An elimination theorem for all closed process terms of ACP^{srt}_{hs} is given in [BM05]. An elimination theorem for $ACP^{srt}_{hs} + INT$ (ACP^{srt}_{hs} extended with integration) is not given in [BM05]. We find that not all closed process terms of $ACP^{srt}_{hs} + INT$ can be reduced to a basic form. An example of the process term that cannot be reduced to a basic term with the current set of axioms is $\int_{u\in[0,\infty)}\sigma^u_{\mathsf{rel}}(\tilde{\tilde{a}}) \mathbin{\|\!\lfloor} \phi \curvearrowright_V \int_{u\in[0,\infty)}\sigma^u_{\mathsf{rel}}(\tilde{\tilde{b}})$. This is because counterparts of axioms SRCM1bPS and HSSRCM (given below) for integration are missing in $ACP^{srt}_{hs} + INT$.

(SRCM1bPS):
$$\sigma^r_{\mathsf{rel}}(x) \mathbin{\|\!\lfloor} (\nu_{rel}(y) + z) = \sigma^r_{\mathsf{rel}}(x) \mathbin{\|\!\lfloor} z + \partial_A(\nu_{rel}(y))$$

(HSSRCM):
$$\sigma^r_{\mathsf{rel}}(x) \mathbin{\|\!\lfloor} (\phi \curvearrowright_V \sigma^r_{\mathsf{rel}}(y) + z) = \sigma^r_{\mathsf{rel}}(x) \mathbin{\|\!\lfloor} (\sigma^r_{\mathsf{rel}}(\phi \curvearrowright_V y) + z) + \phi \curvearrowright_V \sigma^r_{\mathsf{rel}}(\tilde{\tilde{\delta}})$$

Remarks

ACP^{srt}_{hs} is provided with a large number of operators for description of hybrid processes. The recursive specification and definition theorem has also been proven for ACP^{srt}_{hs}. Due to the choice of a time deterministic semantics, defining a variable abstraction operator with operational semantics is difficult and has not been achieved yet. A variable abstraction operator in a graph model of ACP^{srt}_{hs} (see [Kha05]) has been defined, but that model needs to be corrected in the light of the errors found in ACP^{srt}_{hs}. ACP^{srt}_{hs} has constructs to limit discontinuities in values of continuous variables in a parallel composition. ACP^{srt}_{hs} does not strive for simplicity. Like Hybrid χ, it has a large number of notations and much detail in its semantics. For example the presence of discontinuity relations makes the semantics of the parallel operator difficult. Also we find that dealing with time and variable evolutions through two separate operators, the relative delay operator (σ_{rel}) and the signal evolution operator ($\curvearrowright$), makes analysis of a specification more complex. While deriving a transition for a process, care must be taken that both the evolution proposition and the delay operator allow the desired delay under the given environment conditions (variables valuation). The paper [BM06] discusses the relation between ACP^{srt}_{hs} and the formalism of hybrid automata [Hen96].

The completeness of axioms is not claimed in [BM05]. As observed shortly before, a parallel operator cannot always be removed from a closed process term of $ACP^{srt}_{hs} + INT$. We propose the following axiom for $ACP^{srt}_{hs} + INT$:

$$\sigma^r_{\mathsf{rel}}(x) \mathbin{\|\!\lfloor} (\phi \curvearrowright_V (\int_{v<r}\sigma^v_{\mathsf{rel}}(\nu_{rel}(y)) + \sigma^r_{\mathsf{rel}}(z))) = \phi \curvearrowright_V \sigma^r_{\mathsf{rel}}(x \mathbin{\|\!\lfloor} (\phi \curvearrowright_V z)) \qquad \text{(INT18)}$$

We choose to call this equation as axiom INT18. We find this equation necessary to eliminate the parallel operator from the train gate controller specification given in Section 2.6. As can be seen, Equation (INT18) is not as general as HSSRCM. But with more effort, a general equation regarding the delay behaviour of left merge can be found. The semantics of the choice operator in ACP^{srt}_{hs} will be modified. Therefore making an effort to (more) completely axiomatize the current semantics of $ACP^{srt}_{hs} + INT$ is not useful.

2.2.5 Summary

A syntax summary of the four process algebras is given in Table 2.5.

Table 2.5: A syntactic comparison of Hybrid Process Algebras

	ACP^{srt}_{hs}	HyPA	Hybrid χ	ϕ-Calculus
Empty process	Not present	ϵ	skip-a syntactic extension stands for $\emptyset$: true $\gg \tau$	
Deadlock	$\tilde{\tilde{\delta}}$-undelayable deadlock	δ-undelayable	δ-undelayable	0-delayable.
Inconsistent Process	$\perp$	Not present	$\perp$	Not present
Actions	$\tilde{\tilde{a}}$ atomic, undelayable valuation changes	a atomic, undelayable No change in valuation	$\emptyset$: True $\gg a$, a is the action label, atomic, undelayable Changes for variables in $J, \dot{C}, L$	$a \in \{N, \bar{N}\} \cup \{\tau\} \cup \{\delta\}$ atomic, delayable No change in valuation
Communication actions	same as actions. A partial comm function (γ) given, that defines which actions communicate Data exchange can be modelled	same as actions. A partial comm function (γ) given, that defines which actions communicate Data exchange can be modelled	$ch??\mathbf{x}_n$: receive values from ch, store in var $x_1, \ldots, x_n$ $ch!!\mathbf{e}_n$: send values of expr. $e_1, \ldots, e_n$on ch.	$a(\overrightarrow{y})$: receive from link a save in $\vec{y}$ $\bar{a}\langle\vec{x}\rangle$: send $\vec{x}$ on link a a-link name, $\vec{x}$-vector of values or names $\vec{y}$-vector of variables
Flow clauses \ Evolution Op.	$\phi \ulcorner\!\blacktriangledown_V P$ ϕ – a delay predicate variables $\in V$ are of class C^∞	$(V \mid P_f)$ P_f – a delay predicate variables $\in V$ cannot jump at the start of delay	u an algebraic, differential, equation or inequality	flow constraints are part of the environment
Instantaneous modifications	$\chi \ulcorner\!\blacktriangledown P$ χ – a jump predicate in terms of old & new values of variables	$d \gg P$ d – of the form $[V \mid P_r]$ Variable $\in V$ allowed to jump P_r a jump predicate	$W : r \gg a$ W-a set of non-jumping variables r-a jump predicate a-an action with label a	Env. Actions (i)-assignments $[\ \gamma \rightarrow \vec{\mathbf{x}} := \vec{\mathbf{e}}\ ]$ γ-a predicate, $\vec{x}$-variable vector $\vec{e}$-expression vector (ii)-resets $[\gamma \rightarrow$ reset $\langle$list1$\rangle$ with $\langle$list2$\rangle]$ $\langle$list1$\rangle$-list of reset variables $\langle$list2$\rangle$-list of new clauses
Delay Operator	$\sigma^r_{\mathsf{rel}}(P)$ σ_{rel}-relative delay operator r-delay duration	can be modelled	$[P]$-any delay operator adds an arbitrary delay before P $\triangle_r P$-delay operator adds a delay of r time units before P $\triangle r \triangleq \triangle_r$ skip	all constructs are delayable if the env. can delay except an env action with true γ
Time out	$\nu_{rel}\ (P)$ ν_{rel} – relative time out $\nu_{rel}\ (P)$ cannot delay initially	can be modelled	can be modelled	can be modelled
Signal emission	$\psi \wedge\!\!\blacktriangle P$ ψ – a predicate on variables & their derivatives	Not present	$\psi \ulcorner\!\blacktriangledown P$ ψ – a predicate on variables & their derivatives	Not present

Continued on Next Page. . .

	ACP^{srt}_{hs}	HyPA	Hybrid χ	ϕ-Calculus
Guards	$\psi :\rightarrow P$ ψ − a predicate on variables & their derivatives Undelayable	special reinitializations	$b \rightarrow P$ b–a predicate on variables & dotted variables Delayable	$[\ \gamma\ ]\ .P$ γ − a predicate on variables & their derivatives Delayable
Disrupt Operator	Not present	▶ -disrupt operator $P \blacktriangleright Q$–Q can interrupt P $\rhd$ −Left disrupt $P \rhd Q$–First a part of P is executed then $P \blacktriangleright Q$	Not present	Not present
Alternative Composition	$P + Q$ Weak Time deterministic	$P \oplus Q$ Time non-deterministic	$P\ [\!]\ Q$ Strong Time deterministic	$P + Q$ Strongly Time deterministic
Parallelism	$P \parallel Q$ − parallel merge $P \lfloor\!\lfloor Q$ − left merge $P \mid Q$ − Communication merge	$P \parallel Q$ − parallel merge $P \lfloor\!\lfloor Q$ − left merge $P \mid Q$ − Communication merge	$P \parallel Q$ Parallel Operator	$P \parallel Q$ Parallel Operator
Encapsulation Operator	$\partial_H(P)$ H − a set of actions all $a \in H$ are blocked	$\partial_H(P)$ H − a set of actions all $a \in H$ are blocked	$\partial_H(P)$ H − a set of actions all $a \in H$ are blocked	Not present
Local Variable Operator	Only in graph model	$\vert[V \vert P]\vert$V-a set of variables to abstract from	$\vert[\ {}_V\sigma_\perp, C, L \mid p\ ]\vert$ variable scope operator declares discrete, continuous & algebraic variables local to p	$\nu x\ P$ declares a variable x which is local to P
Local Channel Operator	Not present	Not present	$\vert[\ {}_H H \mid p\ ]\vert$ channel scope operator declares a set of channels H local to p	$\nu a\ P$ declares a channel a which is local to P
Local Recursive Definition Operator	Not present	Not present	$\vert[\ {}_R R \mid p\ ]\vert$ recursion scope operator declares a recursive definition R local to p	Not present
Replication	can be modelled by recursion	can be modelled by recursion	can be modelled by recursion	$!P \equiv P\Vert!P$
Urgent communication	can be modelled all actions are undelayable	can be modelled	$\upsilon_H(P)$ H − a set of channels communication on all channels in H must take place as soon as possible	can be modelled
Integration	$\int_{u \in r} \sigma^u_{\mathsf{rel}} P$ r − a time interval $\int_{u \in r} \sigma^u_{\mathsf{rel}} P$ − can delay for any time b/w min & max of r.	can be modelled	can be modelled	can be modelled
Localization Operator	$x \nabla P$ localizes discontinuities of x if P can delay	Not present	Not present	Not present

2.3 Flow-determinism

An important discussion in the hybrid process algebra community, is whether hybrid models should be *flow-deterministic*. In other words, whether the observation of a single flow might lead to different states[1].

In terms of the hybrid transition system defined in Section 2.1, we define flow determinism as:

Definition 1 *A hybrid transition system* $(\boldsymbol{X}, \Sigma, \boldsymbol{T}, \varphi)$ *is* flow deterministic *when for any flow* $\rho \in (\boldsymbol{T} \mapsto \Sigma_c)$ *we have that* $x \overset{\rho}{\longmapsto} x' \wedge x \overset{\rho}{\longmapsto} x''$ *implies* $x' = x''$*.*

Underlying this discussion is the observation in timed process algebras, that the passage of time does not actively choose between processes (although sometimes alternatives may be dropped due to a time-out). The property of time-determinism, is useful in the analysis of timed systems, and some process algebras, for example [BM05], would like to retain it when studying hybrid systems. Others, for example [CR05] , however, argue that in hybrid systems the flow of (hidden) variables can lead to choices as time passes, meaning that non-determinism with respect to flows can occur and should be modelled.

In the four process algebras under study, the question of retaining flow-determinism plays a major role in two syntactic constructs: the choice operator and the abstraction from variables. In the next two subsections, we discuss the consequences of the different choices that are made in each of the process algebras.

Flow-deterministic choice

In ordinary process algebra that only describes discrete behaviour, when a choice is made between two processes, this choice is made non-deterministically. This means, that if the external interaction of the two processes is the same, one cannot influence the choice that is being made internally. In hybrid processes, the same holds for choices that are based on atomic actions, but a different solution is sometimes used for a choice that is based on continuous behaviour.

- HyPA makes a non-deterministic choice between flows in exactly the same way as it is done for actions, therefore flow-determinism is in no way enforced.

- In ϕ-calculus, an environment has a unique flow with it [RS03], and changes in the environment are made through non-deterministic reset actions on the environment. Interestingly, this means that a non-deterministic choice between flows is made (through actions), while the resulting hybrid transition system is still flow-deterministic.

- Hybrid χ enforces synchronization of flows in all alternatives, meaning that all alternatives must be able to execute a common flow transition. Thus, a choice between flow-deterministic processes results in a flow-deterministic composition. However, if the processes cannot agree on a common flow, a deadlock occurs rather than that a choice between the processes is made.

[1] The notion of flow-determinism is not to be confused with the control notion of *observability*[DB95, PW98]. With observability, one can even know the end-state of an observed transition if the starting state is unknown.

- For ACP_{hs}^{srt} the semantics is such that alternative processes must synchronize their flows whenever they have the same duration, thus delaying the choice between processes in a similar fashion as χ does. However, if a flow with a certain duration is not possible in some of the alternatives, then these alternatives are dropped while the choice between the other alternatives is delayed until after the flow. As with χ, this solution preserves flow-determinism but as we describe further on, there are still some peculiarities, (also the choice operator turns out to be non-associative).

In figure 2.1, we have indicated how the choice operator behaves when composing a process P given by $\dot{x} = x$ and a process Q given by $\dot{x} = x^2$, or, in process algebraic notation:

$$
\begin{array}{lll}
\text{HyPA} & : & [x \mid x^+ = 0] \gg ((x \mid \dot{x} = x) \oplus \left(x \mid \dot{x} = x^2\right)) \\
\phi\text{-calculus} & : & [x := 0].([\text{reset } x, \dot{x} \text{ with } \{x \mid \dot{x} = x\}].0 \\
 & & +[\text{reset } x, \dot{x} \text{ with } \{x \mid \dot{x} = x^2\}].0) \\
\text{Hybrid } \chi & : & (x = 0) \curvearrowright (\dot{x} = x \mathbin{[\,]} \dot{x} = x^2) \\
ACP_{hs}^{srt} & : & (x = 0) \wedge\!\!\blacktriangle ((\dot{x} = x) \curvearrowright\!\!\blacktriangledown_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}})+ \\
 & & (\dot{x} = x^2) \curvearrowright\!\!\blacktriangledown_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}}))
\end{array}
$$

Note, that P and Q have a common solution $x(t) = 0$ for the initial condition $x = 0$.

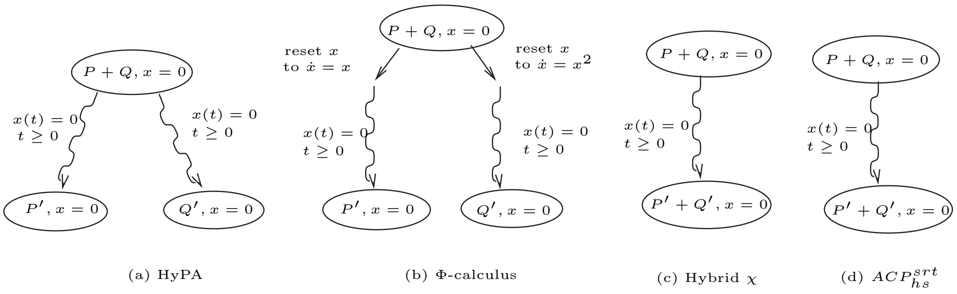
None of the hybrid process algebras take the view that alternatives should only synchronize if they are able to, and the choice between delaying alternatives that cannot synchronize is resolved non-deterministically. (Note: Taking this approach also corrects the non-associativity of Choice as has been recognized by the authors of [BM05] and proposed in Chapter 3, Section 3.7). At present, the algebra ACP_{hs}^{srt} as given in [BM05] puts too much emphasis on the duration of the flows. This introduces another anomaly (besides the non-associativity of Choice) in ACP_{hs}^{srt}, as is illustrated by the following example in which synchronization is not possible.

In the following, we give an example in which synchronization is not possible.

Consider a process P associated with the constrained differential equation $(\dot{x} = 1 \wedge x \leq 1)$ and a process Q associated with $(\dot{x} = x + 1 \wedge x \leq 1)$. We now study the choice between these processes, when x is initially set to 0. This choice is modelled in the four process algebras as follows:

$$
\begin{array}{lll}
\text{HyPA} & : & [x \mid x^+ = 0] \gg ((x \mid \dot{x} = 1 \wedge x \leq 1) \oplus \\
 & & (x \mid \dot{x} = x + 1 \wedge x \leq 1)) \\
\phi\text{-calculus} & : & [x := 0].([\text{ reset } x, \dot{x} \text{ with } \{x \mid \dot{x} = 1\}, \\
 & & \{x \mid x \leq 1\}].0 + [\text{ reset } x, \dot{x} \text{ with} \\
 & & \{x \mid \dot{x} = x + 1\}, \{x \mid x \leq 1\}].0) \\
\text{Hybrid } \chi & : & (x = 0) \curvearrowright ((\dot{x} = 1 \wedge x \leq 1) \mathbin{[\,]} \\
 & & (\dot{x} = x + 1 \wedge x \leq 1)) \\
ACP_{hs}^{srt} & : & (x = 0) \wedge\!\!\blacktriangle ((\dot{x} = 1 \wedge x \leq 1) \curvearrowright\!\!\blacktriangledown_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}}) \\
 & & +(\dot{x} = x + 1 \wedge x \leq 1) \curvearrowright\!\!\blacktriangledown_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}}))
\end{array}
$$

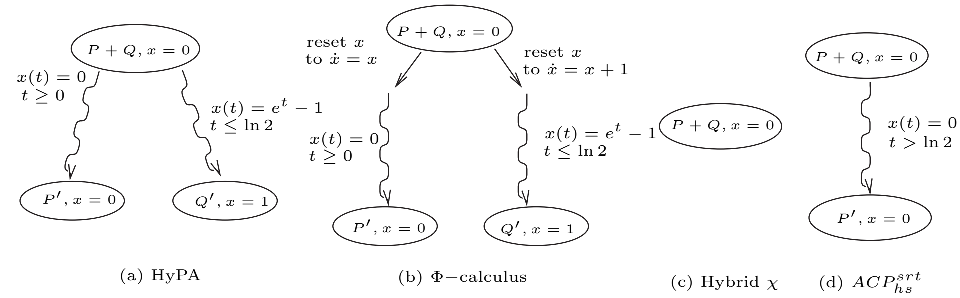
Starting with an initial value of $x = 0$, P can flow for 1 time units with $x(t) = t$, while Q has a different flow $x(t) = e^t - 1$ until $t = \ln 2$. This means, as illustrated in figure 2.2, that HyPA generates two separate flow transitions, one representing a choice for P and one a choice for Q, the ϕ calculus chooses the reset of either P or Q and then executes the chosen flow, and Hybrid χ gives a deadlock since P and Q cannot synchronize . ACP_{hs}^{srt} executes behaviour that, in our opinion, is rather strange. In their semantics, it is not possible to perform a flow



(a) HyPA (b) Φ-calculus (c) Hybrid χ (d) ACP^{srt}_{hs}

(a) Non deterministic Choice (b) Non deterministic Choice (c) Flow deterministic Choice
(d) Flow deterministic Choice

Figure 2.1: Synchronization of flows



(a) HyPA (b) Φ−calculus (c) Hybrid χ (d) ACP^{srt}_{hs}

(a) Non-deterministic Choice (b) Non deterministc Choice (c) $P + Q$ cannot flow
(d)Process flowing longer is chosen

Figure 2.2: Non-synchronous flows

shorter than or equal to $\ln 2$ (because that would require synchronization), but it is possible to execute the flows of P that have a duration longer than $\ln 2$, because Q has no alternatives of this length. As mentioned before, we expect that a slight change in the semantics, with less emphasis on the duration of flows, will fix this problem.

Variable abstraction

In order to support the analysis of hybrid systems through abstraction, all hybrid process algebras (except ACP^{srt}_{hs}) have been extended with a mechanism to hide certain flow variables. A variable abstraction operator has been defined for the graph model of ACP^{srt}_{hs} in [Kha05]. In general hiding a variable x of process p appears as follows in the different algebras.

$$\begin{array}{lcl} \text{HyPA} & : & [\![\{x\} : p\,]\!] \\ \phi\text{-calculus} & : & \nu\, x\, p \\ \text{Hybrid } \chi & : & [\![_{\mathrm{V}}\, \{x\} :: p\,]\!] \\ ACP^{srt}_{hs} & : & x \triangle p \end{array}$$

With respect to the preservation of flow-determinism while abstracting from variables, an important discussion is still going on. In Hybrid χ and HyPA the view is taken that flow-determinism need not be preserved after abstraction from variables, because those unobservable variables may become responsible for choices that are made over time. In particular, abstracting from all flow-variables leads to a timed transition system that is not time-deterministic. From a control-theory perspective, this seems to be the right view on abstraction, because the moments of choice remain important for the controllability and observability of a hybrid system using continuous variables.

In ACP^{srt}_{hs} the view is taken that abstraction from all variables should result in a time-deterministic transition system, and hence abstraction from one variable should preserve flow-determinism. An idea to define a variable abstraction operator for ACP^{srt}_{hs} is to give time stamped semantics (see [BR04], where timed ACP is given time stamped semantics) to the algebra and then extend it with a variable abstraction operator. In a time stamped semantics, delay transitions are **replaced** by the predicates of possibility of a delay. In the absence of delay transitions, the issue of preserving time determinism does not arise.

In the ϕ-calculus, uniqueness of solutions is assumed for the autonomous differential equations [RS03], which leads to flow-deterministic transition systems regardless of the view on abstraction. The abstraction operator in ϕ-calculus does not hide the flow of a variable from the label of a flow transition as is done in Hybrid χ and HyPA. The operator ν excludes other processes from accessing a local variable of a process by renaming variables with common names.

Interestingly, the abstraction from actions by renaming to an internal action τ, has not been developed for any of the hybrid process algebras yet. But then again, the discussion on how to treat the τ action has not come to an end in timed-process algebras either [RvW07].

2.4 Discontinuities

In this section, we discuss discontinuities in environment variables that occur during the execution of a hybrid process.

In a hybrid system the evolution of environment variables is not always continuous. We can study discontinuities in following different scenarios:

1. Discontinuities during actions (Resets)

2. Discontinuities during delays (Jumps)

3. Discontinuities in Mode switching–i.e. discontinuities as one delay predicate is taken over by another

4. Discontinuities when process terms are composed in parallel

We discuss these different forms of discontinuities one by one:

1. **Discontinuities during actions:**

   Discontinuities accompanying an action are incorporated in a system model by action predicates (Hybrid χ), reinitializations (HyPA), assignments or resets (ϕ-calculus) and signal transition propositions (ACP^{srt}_{hs}).

   In ϕ-calculus, variables are updated to specific values where as in other process algebras, variables can be updated arbitrarily according to a given condition.

2. **Discontinuities during delays:**

   In HyPA, the solution to flow predicates is a parameter to the theory. Most practical cases require some continuity requirements on variables to be fulfilled. Therefore most solutions are continuous or continuously differentiable functions of time.

   In ϕ-calculus, the variables of trajectories are required to be continuously differentiable functions of time.

   In Hybrid χ, variables are divided into different categories depending upon their continuity requirements during a delay. The solution to delay predicates is a parameter to the theory of Hybrid χ and can be adopted according to the problem at hand. In [BMR$^+$06], we find that the trajectories of algebraic and dotted continuous variables can be discontinuous; the continuous variables evolve continuously; whereas the values of discrete variables remain constant during a delay.

   In ACP^{srt}_{hs}, solutions to evolution propositions are piece-wise continuously differentiable functions of time. The variables that are specifically declared as smooth by evolution operator cannot jump during the delay.

3. **Mode Switching:**

   In HyPA, one flow clause can be taken over by another clause through a disrupt operator. At the start of a new flow clause variables are allowed to jump arbitrarily unless specifically mentioned to remain continuous. Hence, mode switching can take place without an intermediate action.

   In ϕ-calculus, in mode switching the environment of a process needs to be updated by a *reset* action. The reset action removes the flow clauses of a variable (whose evolution needs to be switched) from the environment and replaces them by other flow clauses. The values of the variables do not jump during such a reset. If an initialization is required as new flow clauses take over, the variable has to be assigned a value. Thus mode switching and resets of variables required with it, are accomplished in two subsequent environmental actions.

   In ACP^{srt}_{hs}, one evolution proposition cannot be taken over by another evolution proposition (that offers solutions different from the last one) without executing an action in between. The actions in ACP^{srt}_{hs} allow arbitrary jumps of variables, therefore with an action in between, the problem of setting up initial conditions for the new evolution proposition does not occur. Hence, mode switching requires an action.

   In Hybrid χ , a delay predicate u represents never terminating infinite behaviour. There is no disrupt operator in Hybrid χ . Usually a delay predicate appears in alternative composition with other process terms and the whole system ends delaying as soon as the environment variables reach the limits enforced by the delaying predicate or an action of a process term in alternative to u becomes urgent. Thus, for mode switching in Hybrid χ , an action is required in between two delay predicates. The required variable discontinuities can then be taken care of in the action predicate.

4. **Discontinuities allowed in a parallel composition:**

   As mentioned before in Section 2.1, all process algebras are provided by a means of declaring environment variables local to a process. The variable abstraction operator for ACP^{srt}_{hs} has been defined in a graph model (see [Kha05]). The preference for time determinism (or we can say flow determinism) in ACP_{hs}^{srt} makes it difficult to give an operational semantics to the variable abstraction operator. On the other hand, ACP_{hs}^{srt} introduces some special constructs to inhibit variable discontinuities when two or more processes are composed in parallel. We discuss them below.

In parallel composition between two or more processes that share environment variables, a situation can arise when a process tries to update a variable that is declared as continuous by another process. A feature that is present in ACP_{hs}^{srt} and is not present in other process algebras is to disallow modifications or discontinuities to a variable declared as continuous by a **delayable** process in parallel composition with another process. This is done by means of a discontinuity operator and discontinuity relations. A discontinuity operator when applied on a process term yields the transitions from which only those discontinuities result that are allowed by the process term–i.e. discontinuities that do not effect a continuous variable of the term. In ACP_{hs}^{srt}, a variable is declared continuous through the evolution operator. This continuity restriction holds only if the process is delayable. The semantic rules of parallelism ensure that a process in parallel composition with a delayable process can only perform action transitions that do not modify a continuous variable. This is done with the help of discontinuity relations.

To compare the behaviour of a parallel process in different process algebras, consider the following example:

Example 2 *Consider a process with an environment variable x. The process initializes the variable x to zero and then lets it continuously evolve with derivative 1 forever. We represent this process in different process algebras:*

$$\begin{array}{lcll}
\textit{Hybrid } \chi\textit{:} & P & = & (x=0) \mathbin{\curvearrowright} (\dot{x}=1) \\
\textit{HyPA:} & P & = & [\ x \mid x^{+}=0\ ] \gg (x \mid \dot{x}=1). \\
\phi\textit{-Calculus} & P & = & [\ x := 0\ ]\ .\ [\ \textit{reset } x \textit{ with } \{x \mid \dot{x}=1\}\ ]\ .0. \\
ACP_{hs}^{srt} & P & = & (x=0) \mathbin{\wedge\!\blacktriangle} (\dot{x}=1) \mathbin{\curvearrowright}_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}}).
\end{array}$$

Let's study its behaviour in parallel composition with another process that tries to reset variable x every four seconds. The process can be defined through recursion as follows:

$$\begin{array}{lcll}
\textit{Hybrid } \chi\textit{:} & R & = & \vartriangle 4;\ \{x\} : x = 0 \gg \tau;\ R. \\
\textit{HyPA:} & R & = & [\ t \mid t^{+}=0\ ] \gg (t \mid \dot{t}=1) \rhd [\ t^{-}=4\ ] \\
 & & & \gg\ [\ x \mid x^{+}=0\ ] \gg R. \\
\phi\textit{-Calculus} & R & = & \nu t\ [\ t := 0\ ]\ .\ [\ \textit{reset } t \textit{ with } \{t \mid \dot{t}=1\}\ ]\ .R'. \\
 & R' & = & [\ t = 4 \rightarrow x := 0\ ]\ .\ [\ t := 0\ ]\ .R'. \\
ACP_{hs}^{srt} & R & = & \sigma_{\mathsf{rel}}^{4}((x^{\bullet}=0) \mathbin{\Rsh} \tilde{\tilde{a}} \cdot R).
\end{array}$$

Note that in ACP_{hs}^{srt}, we don't need to model passage of time through a clock variable. We have an operator σ_{rel}. In Hybrid χ , the passage of time is modelled by means of a syntactic extension $\vartriangle t$, where t is any non-negative real value.

Below, we examine how $P \parallel R$ behaves in different process algebras.

(a) In ACP_{hs}^{srt}, $P \parallel R$ will delay for four time units and evolve with trajectory $\dot{x}=1$.

$$\begin{array}{l}
\langle (x=0) \mathbin{\wedge\!\blacktriangle} (\dot{x}=1) \mathbin{\curvearrowright}_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}}) \parallel \sigma_{\mathsf{rel}}^{4}((x^{\bullet}=0) \mathbin{\Rsh} \tilde{\tilde{a}} \cdot R)\ ,\ x \mapsto 0 \rangle \\
\xmapsto{4,\dot{x}=1} \langle (\dot{x}=1) \mathbin{\curvearrowright}_{\{x\}} \sigma_{\mathsf{rel}}^{*}(\tilde{\tilde{\delta}}) \parallel \sigma_{\mathsf{rel}}^{0}((x^{\bullet}=0) \mathbin{\Rsh} \tilde{\tilde{a}} \cdot R)\ ,\ x \mapsto 4 \rangle
\end{array}$$

Here $x \mapsto 0$ indicates the initial valuation and $x \mapsto 4$ indicates the final valuation. At this point the process at the right hand side cannot wait any further. It must perform action $\tilde{\tilde{a}}$ immediately. The action $\tilde{\tilde{a}}$ is accompanied with a transition proposition that modifies the value of variable x. The process at the left hand side can still delay. Any action performed by process on the right must satisfy the continuity constraints of the process on the left. The continuity requirements, determined through discontinuity relations, of the process on the left hand side, state that the value of x and its derivative cannot be modified. Therefore process R cannot perform action a. After four time units R cannot wait. Therefore the parallel composition cannot wait and the process $P \parallel R$ deadlocks after four time units.

(b) Now we consider the evolution of $P \parallel R$ in Hybrid χ . A Hybrid χ process consists of a process term, a variable valuation and an environment. The process must start with a value of 0 for x. We want the trajectory of x to be absolutely continuous. Therefore it is declared as a continuous non-jumping variable. The environment E is $(\{x\}, \emptyset, \emptyset, \emptyset, \emptyset)$. (Remember that an environment in Hybrid χ is a tuple (C, J, L, H, R). See section 2.2.3.)

$$\langle P \parallel R \ , \ \{x \mapsto 0, \text{time} \mapsto 0\} \ , \ E\rangle$$

$$\xmapsto{4,\rho} \langle \dot{x} = 1 \| \vartriangle 0; \ \{x\} : x = 0 \gg \tau; \ R \ , \ \{x \mapsto 4, \text{time} \mapsto 4\} \ , \ E\rangle \quad \ldots(2)$$

(ρ indicates that $\dot{x} = 1$).

$$\xrightarrow{\tau} \langle \dot{x} = 1 \parallel \{x\} : x = 0 \gg \tau; \ R \ , \ \{x \mapsto 4, \text{time} \mapsto 4\} \ , \ E\rangle$$

($\vartriangle$ 0 terminates with a τ action.)

$$\xrightarrow{\tau} \langle \dot{x} = 1 \parallel R \ , \ \{x \mapsto 0, \text{time} \mapsto 4\} \ , \ E\rangle$$

(the derivation repeats itself from step (2)).

Thus after every four seconds x is initialized and during delays the process evolves with trajectory $\dot{x} = 1$.

(c) The process $P \parallel R$ will behave the same (as it behaves in Hybrid χ) in HyPA and ϕ-calculus.

(d) In ϕ-calculus, we can declare x to be a local variable of process P. Then x referred to in R is not the same x as in P. In fact x from process P, will be given a fresh name with respect to all the names in the environment and process P. That way R will not be able to effect the trajectory or assign any values to local x of P. The variable scope operator of Hybrid χ can confine the scope of a variable to a process term and hence can prevent other processes from accessing or modifying it. The variable declared in a variable scope operator are invisible to processes outside the scope operator. Similar effect can be achieved in HyPA by variable abstraction operator.

Declaring variables to be local is a different approach to that of discontinuity relations in ACP^{srt}_{hs}. In this approach, a variable declared continuous is visible to all processes in parallel . But these processes cannot modify the variable if the process declaring it continuous can delay. Note that the semantics of ACP^{srt}_{hs} enforces the continuity requirements of a continuous variable, *only* when the process declaring it is delayable. If a process declaring a variable continuous is not delayable, then the continuity requirements are not enforced.

We continue further with what variable discontinuities are allowed in a process algebra. The algebra ACP^{srt}_{hs} is extended with a localization operator, denoted by $\triangledown$. Localization of a variable to a process will not inhibit discontinuities caused by actions from within the process, but any outside process will not be able to modify the localized variable when the given process can delay.

We illustrate the effect of this operator by means of the following example:

Example 3 *Consider the example of a ball moving on a smooth horizontal surface. The ball decelerates because of friction between the ball and the horizontal surface. The force of friction on the ball is given by $F = \mu N$, where μ is the coefficient of friction and N is the normal force at the point of contact. As the ball progresses, the material of the horizontal surface keeps on varying after regular intervals and so does the frictional constant. There is friction due to air resistance also, but we disregard this. Therefore, at any instant the deceleration of the ball is greater than that caused by the ground friction alone, i.e. $\dot{x} \leq -(y(time)N)/m$, where m is the mass of the ball, $y(time)$ denotes the value of frictional constant at instant* time*. The motion of ball with initial velocity $\dot{x} = 200$, can be specified in ACP^{srt}_{hs} as follows:*

$$P = (\dot{x} = 200) \;{}^{\wedge}\!\blacktriangle\; (\dot{x} \leq -(yN)/m) \;\ulcorner\!\blacktriangledown_{\{x\}}\; \sigma^{*}_{\mathsf{rel}}(\tilde{\tilde{\delta}})$$

The change in frictional constant after every r time units can be expressed as follows:

$$Q = \sigma^{r}_{\mathsf{rel}}((y^{\bullet} = f(time)) \;\ulcorner\!\blacktriangledown\; \widetilde{\widetilde{modify\mu}} \cdot Q)$$

A parallel composition between the two processes specifies the whole system. Notice in $P \parallel Q$, x is a continuous variable. No discontinuities for x are allowed when $P \parallel Q$ is placed in composition with other processes, whereas the discontinuities for y are allowed. We would like to restrict the discontinuities to y to only to $P \parallel Q$. The localization operator applied to the parallel composition, $y\triangledown(P \parallel Q)$, will update the value of y after every four seconds but will not allow any third process to update y after the action $\widetilde{\widetilde{modify\mu}}$ has taken place. (Note that the variable time will also evolve as a continuously differentiable variable (like the variable x) with constant derivative 1. We omit the details of its evolution.)

Concluding Remarks

We observed that in all process algebras, continuously or continuously differentiable functions are allowed as trajectories of continuous variables. In fact in HyPA and Hybrid χ, the solutions to flows can be parameters of the theory and can be chosen considering the problem at hand. In all process algebras (except HyPA), mode switching requires an action. ACP^{srt}_{hs} does not define a variable abstraction operator in operational semantics. The process algebra ACP^{srt}_{hs} provides some other operators to inhibit discontinuities in parallel composition.

2.5 Parallelism

The extent to which parallel processes are allowed to interact with each other is an important property that determines usefulness of a process algebra. Hybrid CSP (see [Jif94]), is probably the first process algebra for specification of hybrid systems. A provision lacking in hybrid CSP was that processes in parallel composition were not allowed to influence each others variables. Process algebras for hybrid systems presented later allow more interaction between parallel processes. In this section, our aim is to present what interactions are allowed between parallel processes in each process algebra. This section is structured as follows: First we introduce the operators used to describe parallelism in each process algebra; then we describe how communication takes place; further on we discuss the synchronization in delay behaviour of parallel processes and finally we explain the restrictions (if any) on the action behaviour of parallel processes in each process algebra.

2.5.1 Operators of Parallel Composition

ACP^{srt}_{hs} and HyPA are extensions of ACP. They inherit operators for parallel composition from ACP. The operators for parallel composition are as follows: $P \parallel Q$-parallel merge; $P \mid Q$-communication merge; and $P \mathbin{|\!\!\lfloor} Q$-left merge. Primarily these operators have the same meaning as they had in ACP. The addition of data and data manipulation operators to process algebras ACP^{srt}_{hs} and HyPA add complexity to the semantics of parallel operators. Both HyPA and ACP^{srt}_{hs} have a large number of axioms for parallel operators. These axioms reflect interactions between reinitializations and flow clauses (in HyPA) or signal transitions and signal evolutions (in ACP^{srt}_{hs}) of processes composed in parallel. Hybrid χ has only one operator, i.e. parallel merge $\parallel$, for parallel composition. As shown in [Mol90], an extra operator is necessary for a finite axiomatization of parallel merge (see also [AFIL07]). Hybrid χ does not give a comprehensive set of axioms for its operators. ϕ-Calculus also has only a single operator (parallel merge $\parallel$) to describe parallelism. ϕ-Calculus does not give any axioms.

2.5.2 Communication

1. **HyPA:**

   In HyPA, there are no special send and receive actions. In HyPA, communication takes place as in ACP [BW90] by using a communication function. A communication function is denoted by γ and defined as $\gamma : A \times A \to A$, where A is a set of actions. The communication function defines which actions can communicate. It is a partial function. The communication defined is handshaking and synchronous, i.e. only two actions can communicate and simultaneous execution of actions is required for communication. The actions that can communicate can also execute independent of each other. In order to enforce communication between actions that can communicate, an encapsulation operator (∂_H) is defined that blocks the individual execution of these actions.

   In communication, P and Q must agree on any variable reinitializations accompanying their first actions, For example, the following process term is forced to communicate by the encapsulation operator ∂_H. It cannot perform action c which is the result of communication, because the communicating processes P and Q do not agree on the new value of variable x.

   $$\partial_{\{a,b\}}([x \mid x^+ = 30] \gg a \parallel [x \mid x^+ = 40] \gg b),$$
   $$\text{(where } \gamma(a,b) = c)$$

   The above process term deadlocks.

During communication, data can be exchanged between process terms as follows:

If some data element d_0 belonging to a data set D, is to be sent during communication, then a summation of parameterized actions with a parameter of type D, over all elements of D is used.

For example,

$$\partial_H(\mathrm{send}(d_0) \odot P \parallel \Sigma_{d \in D}\mathrm{rec}(d) \odot Q(d)),$$

is equivalent to

$$\mathrm{comm}(d_0) \odot \partial_H(P \parallel Q(d_0))$$

with the communication function and H defined as follows:

$$\begin{array}{l} \gamma(\mathrm{send}(d), \mathrm{rec}(d)) = \mathrm{comm}(d), \text{ for all } d \in D \\ H = \{send(d), rec(d) \mid d \in D\}. \end{array}$$

This will have an effect similar to

$$h!!d_0;\ P \parallel h??x;\ Q(x)$$

in Hybrid χ .

2. **ϕ-Calculus:**

   In ϕ-calculus, communication takes place as in π-calculus [Mil99]. A specification consists of a set of link names. The most basic actions are send and receive on a link name. Simultaneous execution of send and receive on a link name results in communication (observed in the form of a silent action). Messages exchanged in a ϕ-calculus specification comprise of values, link names and environment variable names. Passing of a private link name or private variable to other processes widens the scope of the link or variable name.

   In ϕ-calculus, environmental actions of parallel processes take place in an interleaving fashion. They can act as guards before communicating actions but cannot be attached as jump predicates to communication actions as reinitializations in HyPA or signal transitions in ACP^{srt}_{hs}.

   Consider the following process:

   $$\nu a\ [\ x := 0\ ]\ (\ [\ x := 10\ ]\ .\bar{a}\langle x\rangle \parallel\ [\ x := 20\ ]\ .a(z))$$

   The variable x is shared between the parallel processes. The environmental actions $[x := 10]$ and $[x := 20]$ occur in an interleaving fashion. Therefore, the value of variable z after the communication depends on which action takes place last.

3. **Hybrid χ:**

   Hybrid χ has CSP-like send and receive actions. A set of channels is defined for a Hybrid χ process. Values represented by closed variable expressions can be exchanged in communication. Exchange of values between two process terms can take place by means of simultaneous send (denoted by $h!!\mathbf{e}_n$) and receive (denoted by $h??\mathbf{x}_n$) actions on the same channel, where h is the name of a channel, $\mathbf{e}_n$ represents a vector of closed variables expressions, the values of $e_1, \ldots e_n$ are sent on channel h and $\mathbf{x}_n$ is a variable vector in which the received values are stored.

The labels of send, receive or communication actions cannot be used in action predicates (see Section 2.2.3 for definition of action predicates). The definition of an action predicate allows action labels only from the set A_{label}, which is disjoint from A_{comm} that contains the labels of send, receive and communication action. Without a jump predicate to control the future values of variables, jumping, dotted and algebraic variables can jump arbitrarily during communication.

In order to enforce communication between matching send and receive actions, an encapsulation operator is used. An example of communication between Hybrid χ processes is given below:

$$\partial_{iA}((x = 30) \to h!!x \parallel h??y)$$

The set iA contains individual send and receive actions on channel h. After communication, the value 30 is stored in variable y.

4. ACP^{srt}_{hs}

   Communication takes place in ACP^{srt}_{hs} via a communication function as it takes place in HyPA. Like HyPA, communicating process terms must agree on any changes in variable valuations accompanying their first actions, For example,

$$\partial_{\{a,b\}}((x^\bullet = 30) \ulcorner\!\!\blacktriangledown\, \tilde{\tilde{a}} \mid (x^\bullet = 40) \ulcorner\!\!\blacktriangledown\, \tilde{\tilde{b}}),$$
$$\text{(where } \gamma(a, b) = c\text{)}$$

   Process terms in the above example cannot communicate as they do not agree on variable valuations after their first actions.

2.5.3 Delay behaviour

The process terms in parallel composition must synchronize during a delay. i.e. the process terms have the same variable evolution during the delay. A parallel composition can delay as long as all its component process terms can delay. In ϕ-Calculus, the flow constraints on variables are given in the environment and not in the process terms. The behaviour of process terms in parallel is observed in the same environment. Therefore, the condition that processes in parallel must agree on a common flow looses its meaning in ϕ-Calculus. This is further explained by an example while discussing delay in ϕ-Calculus.

We discuss delay of a parallel composition in each process algebra one by one.

- **HyPA:**

  In HyPA, delays are modelled by means of flow clauses. If two flow clauses are placed parallel to each other then the resulting composition can only flow in a way that is possible for both flow clauses. Consider the following parallel composition,

$$(\ [\ x \mid x^+ = 0\ ]\ \gg (x \mid \dot{x} = 1)) \parallel (\ [\ x \mid x^+ = 20\ ]\ \gg (\dot{x} \geq 0)).$$

  We denote the term on the left hand side by P and the term on the right hand side by R. The reinitialization clauses, $[\ x \mid x^+ = 0\ ]$ and $[\ x \mid x^+ = 20\ ]$, initialize x to values 0 and 20 respectively.

The flow clause in P declares x to be a continuous variable and allows it to flow only as the differential equation $\dot{x} = 1$. The flow clause in R *allows* x to jump initially and gives it more freedom in evolution. $P \parallel R$ will evolve according to $\dot{x} = 1$ and the trajectory of x will start with $x = 0$.

A parallel merge of two process terms can also delay, if one of the process terms terminates immediately and the other can delay.

Process terms composed with communication merge $|$ can delay in the same way. The left merge in HyPA, cannot delay initially. It must begin with an action of the process term on its left.

- **ACP^{srt}_{hs}:**

  A process term in ACP^{srt}_{hs} equivalent to $P \parallel R$, (as defined in HyPA) can be written as follows:

  $$((x = 0) \mathbin{\wedge\!\!\blacktriangle} (\dot{x} = 1) \curvearrowright_{\{x\}} \sigma^*_{\mathsf{rel}}(\tilde{\tilde{\delta}})) \parallel ((\dot{x} \geq 0) \curvearrowright_{\emptyset} \sigma^*_{\mathsf{rel}}(\tilde{\tilde{\delta}}).$$

  A communication merge $P \mid Q$ can delay in the same way.

  There is no empty process in ACP^{srt}_{hs}. A parallel or communication merge of processes can only delay if all constituent processes can delay with a common trajectory.

  The left parallel operator of ACP^{srt}_{hs} is also delayable. $P \mathbin{|\!\!\lfloor} Q$ can delay if the delay behaviors of both processes synchronize.

- **Hybrid χ:**

  In Hybrid χ , a parallel composition can delay only if all component process terms can synchronize during delay. Two delay predicates in parallel composition are as follows:

  $$(x = 0) \curvearrowright ((\dot{x} = 1) \parallel (\dot{x} \geq 0))$$

  The above term can delay with a trajectory of x with derivative of 1.

  The following parallel composition will delay as long as the guard of the second component remains false. The variable x will evolve with $\dot{x} = 1$ during the delay. A false guard can delay in any manner and the term $[\ ch??\ ]$ with any delay operator also has no restrictions on variable evolution.

  $$(x = 0) \curvearrowright ((\dot{x} = 1) \parallel (x = 50) \rightarrow ch!! \parallel\ [\ ch??\ ]\ )$$

  As soon as the guard becomes true, the send and receive actions on channel ch synchronize.

- **ϕ-Calculus:**

  In ϕ-calculus, the flow clauses are part of the environment. They are placed in the environment with a reset action. A reset action of the form [reset $\dot{x}$ with $\{\dot{x} \mid \dot{x} \geq 0\}$] will replace any flow clauses in the environment with $\dot{x}$ as a resetable name. Therefore in a parallel composition, we cannot see the result of interaction of flow clauses of process terms in parallel as we saw in the other process algebras.

  Consider the following process:

  $$\begin{array}{l}(\emptyset, [x := 0]\,.\,[\text{reset } \dot{x} \text{ with } \{\dot{x} \mid \dot{x} = 1\}]\,.\,0 \\ \quad\ \parallel \qquad [x := 20]\,.\,[\text{reset } \dot{x} \text{ with } \{\dot{x} \mid \dot{x} \geq 0\}]\,.\,0)\end{array}$$

We start the process with an empty environment, i.e., there are no variables or flow clauses yet in the environment. All environmental actions are with true guards. They can occur in any interleaving fashion. The process term $0 \parallel 0$ is delayable. The evolution and the initial value of x during the delay depends on the assignments and resets occurring last.

In order to see the interaction of two flow clauses, $\dot{x}$ needs to be removed as reset name from the reset list. This could be easily achieved if empty reset lists were allowed in reset actions, i.e. a reset action of the following form is allowed:

$$[\text{reset } \langle\rangle \text{ with } \{\dot{x} \geq 0\}]$$

The above reset action should not replace any flow clause (for example $\{\dot{x} \mid \dot{x} = 1\}$) from the environment.

A parallel composition of process terms can delay as long as each constituent of the parallel composition can delay. In ϕ-calculus, a silent action followed by an environmental action with true guard is urgent. Therefore it may happen that individual process terms can delay but their parallel composition cannot. For example, consider the following process term

$$\nu\, a(\nu\, x\ [\ x := 0\ ]\ .\ [\ \text{reset } \dot{x} \text{ with } \{\dot{x} = 1\}\ ]\ .\bar{a}(x)\,.0\ \parallel\ a(y)\,.[\gamma]\,.0).$$

In an empty environment after the assignment and reset actions, we get the following process:

$$(\left(\begin{array}{c} x : 0 \\ \{\{\dot{x} \mid \dot{x} = 1\}\} \end{array}\right),\ \nu\, a\,(\bar{a}(x)\,.0\ \parallel\ a(y)\,.[\gamma]\,.0)).$$

The above process can communicate on channel a immediately or it can delay before communication. The maximum duration for which the above process can postpone communication on channel a, depends on the predicate γ. As soon as γ is true, communication must take place.

In the improved version of ϕ-calculus, the delay prefix δ prefixed before a send or receive on a link can arbitrarily delay communication on that link.

2.5.4 Special Restrictions

The process algebras ACP_{hs}^{srt} and Hybrid χ allow specifying assumptions about the environment variables. Hence some restrictions exist on the action behaviour of parallel processes of ACP_{hs}^{srt} and Hybrid χ in order to protect assumptions of processes in parallel.

We explain them for both process algebras one by one below:

1. **ACP_{hs}^{srt}:** In ACP_{hs}^{srt}, in interleaving of actions, a process term cannot perform an action that results in a violation of the signal of another process in parallel. The signal of a process term is a proposition that states what assumptions about the environment are made by a process term. As explained in section 2.2.4, assumptions about the environment are specified through a signal emission or an evolution proposition.

   Consider the following process term. The action a cannot take place before action $\tilde{\tilde{b}}$, as the variable update accompanying it falsifies the signal of process term on the right.

   $$(x^{\bullet} = 4) \ulcorner\!\!\blacktriangledown\, \tilde{\tilde{a}} \parallel (x \leq 3) \wedge\!\!\!\blacktriangle\, \tilde{\tilde{b}}$$

Another example of restriction due to the signal of a process is as follows:

$$(x = 0) \mathbin{{\wedge}\!\blacktriangle} ((x^{\bullet} = 4) \mathbin{\ulcorner\!\!\blacktriangledown} \tilde{\tilde{a}} \parallel (\dot{x} = 1 \wedge x \leq 3) \mathbin{\curvearrowright\!\!\blacktriangledown} \sigma^{1}_{\mathsf{rel}}(\tilde{\tilde{b}}))$$

Action $\tilde{\tilde{a}}$ cannot be performed as the signal of the right hand process requires x to be less than or equal to three.

In HyPA there are no signal emission operator or signal relations as in ACP^{srt}_{hs}. Therefore, there is no such restriction on a HyPA process. The following process term can do action a before action b.

$$[\ x \mid x^{+} = 4\ ] \gg a \parallel [\ x \mid x^{-} \leq 3\ ] \gg b$$

The reinitialization $[\ x \mid x^{-} \leq 3\ ]$ acts only as a guard and not as a signal emission. After performing action a, the above process term will deadlock, as the guard on the right becomes false.

In ACP^{srt}_{hs}, also the continuity requirements of a variable are protected during interleaving. The continuity requirements of the variables are declared via a signal evolution operator and a localization operator. Examples 2 and 3 in Section 2.4 show how these operators inhibit discontinuities.

2. **Hybrid χ:**

   Hybrid χ also has a signal emission operator like ACP^{srt}_{hs}. Hybrid χ specifications can also express assumptions about the environment in a signal emission proposition or a delay predicate. Like ACP^{srt}_{hs}, the assumptions about the environments must always be satisfied during execution of a Hybrid χ process. The semantics of Hybrid χ enforces that a process $(x := y \parallel y = 1)$ behaves the same as $(x := 1 \parallel y = 1)$.

   Now consider another process term.

$$\{x\} : x = 0 \gg \tau;\ (\{x\} : x = 4 \gg l_a \parallel \dot{x} = 1 \wedge x \leq 3)$$

   Let x be a continuous, non-jumping environment variable. The above process term will behave as follows:

   - Internal action τ is executed and x is assigned value 0.
   - $\{x\} : x = 4 \gg l_a$ cannot perform action a, as the predicate requires x to be assigned a value of 4. This is not consistent with the assumptions of the process term on the right.

2.5.5 Concluding Remarks

We observe that interaction of parallel processes in ϕ-calculus is different from that of other process algebras. In ϕ-calculus, a synchronization during a delay or in a communication cannot be modelled as it can be modelled in other algebras. Hybrid χ enforces synchronization during delays but cannot model agreement on variable updates when communication takes place. HyPA and ACP^{srt}_{hs} enforce both synchronization during delays and agreement on variable updates during communication. The parallel composition operator in ACP^{srt}_{hs} and Hybrid χ enforces further restrictions on action behaviour of parallel processes to protect the assumptions of the process about the environment. In ACP^{srt}_{hs}, the continuity requirements of a delayable process are also protected. This is achieved by discontinuity relations which makes the semantics of the parallel operator more complex.

2.6 A Case Study: A Train Gate Controller

We take a case study of a train gate controller from [BM05]. We specify the train gate controller in each process algebra and use the methods available in each process algebra to verify a safety condition. Our aim is to examine the expressiveness of process algebras and compare their strengths of algebraic reasoning.

An informal specification of the behaviour of the system is given as follows:

A train is coming towards the railway crossing gate with its speed between $48m/s$ and $52m/s$. As soon as it crosses the detector placed at 1000 m backward from the gate, an approach signal is sent to the controller. The train may now slow down to speed between 40m/s and 52 m/s. When it crosses the second detector placed at 100m forward from the gate, an *exit* signal is sent to the controller. A new train may come after the current one has passed the second detector, but only at a distance of 1500m or more. The gate is able to receive *lower* and *raise* signals from the controller at any time. When the gate receives a *lower* signal, it lowers from 90° to 0° at a constant rate of $-20°/s$. As soon as it receives a *raise* signal, it raises from 0° to 90° at the same rate in the opposite direction. The controller is able to receive *approach* and *exit* signals from the train detectors at any time. When the controller receives an *approach* signal, it takes less than 5 seconds before a *lower* signal is sent to the gate. When the controller receives an *exit* signal, it takes less than 5 seconds before a *raise* signal is sent to the gate. Because of fault tolerance considerations, an *approach* signal should always cause the gate to go down, and an *exit* signal should be ignored while the gate is going down.

It is assumed that initially there is no train at a distance smaller than 1400m from the gate, the gate is open, and the controller is idling. Moreover, it is assumed that each single train changes its speed only smoothly.

Outline of Our approach

The process algebras differ in their strengths of axiomatic reasoning. As described before, the process algebras HyPA and ACP^{srt}_{hs} have a large number axioms for deriving bisimilar forms of process terms. Hybrid χ offers a few axioms and ϕ-calculus proposes no axioms. Hence, we use different approaches to simplify the train gate controller specification in each process algebra. We briefly mention our approaches below:

1. The train gate controller system in HyPA is linearized completely using the axioms of HyPA. The safety condition is verified by the linearized specification.

2. For ϕ-calculus, there is no axiomatic reasoning available. We construct (part of) the state space of the train gate controller specification in ϕ-calculus and do reachability analysis to verify the safety condition.

3. To linearize Hybrid χ specification, we use the elimination theorem given in [SM06]. The recursive definition and recursive specification principle necessary to reason about recursive specifications have not been proven in Hybrid χ. We assume that these principles hold for Hybrid χ process terms. Unlike HyPA and ACP^{srt}_{hs}, there are very few axioms to study different forms of a Hybrid χ process. The linear form obtained from the elimination theorem is large and contains unreachable options that cannot be eliminated algebraically. The safety condition can be verified by doing reachability analysis on the linear form.

4. The train gate controller specification in ACP_{hs}^{srt} can completely be linearized if we assume the equation INT18 given in section 2.2.4. Without this equation, we cannot eliminate parallel composition from the train gate controller specification. The safety condition is verified by the linearized specification by calculating the minimum time required between events.

The algebraic specification of train gate controller system uses the following environment variables:

x – for the distance of train from gate
r – for the angle of gate with the ground
d – for possible delay of controller.
y – for speed of the train.

All the environment variables must evolve smoothly during delays and must not jump arbitrarily during actions. In ϕ-calculus and Hybrid χ , a separate variable y is used to represent the speed of the train. In Hybrid χ the dotted variables can jump arbitrarily during discrete actions and their trajectories can be discontinuous. Therefore, we declare a continuous environment variable y and set it equal to $\dot{x}$ during delays. In ϕ-calculus, the trajectories of variables are continuously differentiable functions of time. A ϕ-Calculus process environment consists of a variable valuation and a set of constraints on flow of variables. During assignments, only the variables that are assigned new values are updated and other variables in the valuation retain their previous values. The valuation does not include derivatives of variables. In order to make sure that $\dot{x}$ is not altered during environmental actions, we declare a variable y and add a flow constraint $\dot{x} = y$ to the environment.

Below we give the specification of the train gate controller in each process algebra. We simplify the specification in each process algebra and reason about the safety condition that the gate is fully closed whenever the train is within a certain distance from the gate.

2.6.1 HyPA

Specification

The train gate controller specification in HyPA is given by the following expression:

$$\partial_H(\mathit{Trains} \parallel \mathit{Gate} \parallel \mathit{Cntr})$$

where,

1. the recursion variables *Trains*, *Gate* and *Cntr* are defined by the following specifications:

$$\begin{array}{ll}
\text{Trains} & = [x, \dot{x} \mid x^+ \leq -1400 \wedge 48 \leq \dot{x}^+ \leq 52] \\
& \quad \gg T^{\text{far}} \\
T^{\text{far}} & = (x, \dot{x} \mid x \leq -1000 \wedge 48 \leq \dot{x} \leq 52) \\
& \quad \blacktriangleright [x^- = -1000] \gg s(\text{appr}) \odot T^{\text{near}} \\
T^{\text{near}} & = (x, \dot{x} \mid -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52) \\
& \quad \blacktriangleright [x^- = 0] \gg T^{\text{past}} \\
T^{\text{past}} & = (x, \dot{x} \mid 0 \leq x \leq 100 \wedge 40 \leq \dot{x} \leq 52) \\
& \quad \blacktriangleright [x^- = 100] \gg s(\text{exit}) \odot \\
& \quad [x, \dot{x} \mid x^+ \leq -1400 \wedge 48 \leq \dot{x}^+ \leq 52] \\
& \quad \gg T^{\text{far}}
\end{array}$$

$$
\begin{array}{ll}
\mathit{Gate} = & [r, \dot{r} \mid r^{+} = 90 \ \wedge \ \dot{r}^{+} = 0] \gg G^{\mathrm{op}} \\
G^{\mathrm{op}} = & (r, \dot{r} \mid r = 90 \ \wedge \ \dot{r} = 0) \\
 & \quad \blacktriangleright (r(\mathrm{lower}) \odot [\dot{r} \mid \dot{r}^{+} = -20] \gg G^{\mathrm{dn}} \\
 & \quad \oplus r(\mathrm{raise}) \odot G^{\mathrm{op}}) \\
G^{\mathrm{dn}} = & (r, \dot{r} \mid 0 \leq r \leq 90 \ \wedge \ \dot{r} = -20) \\
 & \quad \blacktriangleright ([r^{-} = 0] \gg [\dot{r} \mid \dot{r}^{+} = 0] \gg G^{\mathrm{cl}} \\
 & \quad \oplus r(\mathrm{raise}) \odot [\dot{r} \mid \dot{r}^{+} = 20] \gg G^{\mathrm{up}} \\
 & \quad \oplus r(\mathrm{lower}) \odot G^{\mathrm{dn}}) \\
G^{up} = & (r, \dot{r} \mid 0 \leq r \leq 90 \ \wedge \ \dot{r} = 20) \\
 & \quad \blacktriangleright ([r^{-} = 90] \gg [\dot{r} \mid \dot{r}^{+} = 0] \gg G^{\mathrm{op}} \\
 & \quad \oplus r(\mathrm{raise}) \odot G^{\mathrm{up}} \oplus r(\mathrm{lower}) \\
 & \quad \odot [\dot{r} \mid \dot{r}^{+} = -20] \gg G^{\mathrm{dn}}) \\
G^{\mathrm{cl}} = & (r, \dot{r} \mid r = 0 \ \wedge \ \dot{r} = 0) \\
 & \quad \blacktriangleright (r(\mathrm{lower}) \odot G^{\mathrm{cl}} \oplus r(\mathrm{raise}) \\
 & \quad \odot [\dot{r} \mid \dot{r}^{+} = 20] \gg G^{\mathrm{up}})
\end{array}
$$

$$
\begin{array}{ll}
\mathit{Cntr} = & C^{\mathrm{idle}} \\
C^{\mathrm{idle}} = & (\mathrm{true}) \blacktriangleright \\
 & \quad (r(\mathrm{appr}) \odot [d, \dot{d} \mid d^{+} = 0 \wedge \dot{d}^{+} = 1] \\
 & \quad \gg C^{\mathrm{dn}} \oplus r(\mathrm{exit}) \odot \\
 & \quad [d, \dot{d} \mid d^{+} = 0 \wedge \dot{d}^{+} = 1] \gg C^{\mathrm{up}}) \\
C^{dn} = & (d, \dot{d} \mid 0 \leq d \leq 5 \wedge \dot{d} = 1) \\
 & \quad \blacktriangleright (r(\mathrm{appr}) \odot C^{dn} \oplus \\
 & \quad r(\mathrm{exit}) \odot C^{\mathrm{dn}} \oplus s(\mathrm{lower}) \odot C^{\mathrm{idle}}) \\
C^{up} = & (d, \dot{d} \mid 0 \leq d \leq 5 \wedge \dot{d} = 1) \\
 & \quad \blacktriangleright (r(\mathrm{appr}) \odot [d \mid d^{+} = 0] \gg C^{dn} \oplus \\
 & \quad r(\mathrm{exit}) \odot C^{\mathrm{up}} \oplus s(\mathrm{raise}) \odot C^{\mathrm{idle}})
\end{array}
$$

2. the communication function γ is defined as follows:

$$
\begin{array}{l}
s(d) | r(d) = c(d) \\
d \in \{\mathrm{appr}, \mathrm{exit}, \mathrm{raise}, \mathrm{lower}\}
\end{array}
$$

3. the set H of actions that are to be blocked is given below:

$$
\begin{array}{ll}
H = & \{s(d) \mid d \in \{\mathrm{appr}, \mathrm{exitraise}, \mathrm{lower}\}\} \cup \\
 & \{r(d) \mid d \in \{\mathrm{appr}, \mathrm{exit}, \mathrm{raise}, \mathrm{lower}\}\}
\end{array}
$$

Note: In HyPA, generally the derivatives are not updated in reinitializations as done above. Instead, they are allowed to jump at the start of a flow clause.

Simplifying the Specification

A repeated application of axioms of HyPA simplifies the specification. The simplified specification is given below:

$$\partial_{\mathbf{H}}(\mathbf{Trains} \parallel \mathbf{Cntr} \parallel \mathbf{Gate})$$

$$= \left[\ V \;\middle|\; \begin{array}{l} x^+ \leq -1400 \\ 48 \leq x^+ \leq 52 \\ r^+ = 90 \\ \dot{r}^+ = 0 \end{array} \right] \gg \left(\begin{array}{c|l} x & x \leq -1000 \\ \dot{x} & 48 \leq \dot{x} \leq 52 \\ r & r = 90 \\ \dot{r} & \dot{r} = 0 \end{array} \right) \triangleright$$

$$[\ x^- = -1000\ ] \gg c_1(\text{appr}) \odot\ \partial_H(T^{near} \parallel\ [\ d, \dot{d} \mid d^+ = 0 \wedge \dot{d}^+ = 1\ ] \gg C^{dn} \parallel G^{op})$$

A train arriving from far reaches the detector placed at $1000m$ backwards from the gate and sends an approach signal to the controller. We now continue to expand the expression $\partial_H(T^{near} \parallel\ [\ d, \dot{d} \mid d^+ = 0 \wedge \dot{d}^+ = 1\ ] \gg C^{dn} \parallel G^{op})$.

$$\partial_{\mathbf{H}}(\mathbf{T^{near}} \parallel \mathbf{G^{op}} \parallel\ [\ \mathbf{d}, \dot{\mathbf{d}} \mid \mathbf{d}^+ = \mathbf{0} \wedge \dot{\mathbf{d}}^+ = \mathbf{1}\ ] \gg \mathbf{C^{dn}})$$

$$= \left[\ d, \dot{d} \;\middle|\; \begin{array}{l} d^+ = 0 \\ \dot{d}^+ = 1 \\ r^- = 90 \\ x^- = -1000 \end{array} \right] \gg c_2(\text{lower}) \odot \begin{array}{l} \partial_H(T^{near} \parallel C^{idle} \\ \quad \parallel [\ \dot{r} \mid \dot{r}^+ = -20\ ] \gg G^{dn}) \end{array}$$

$$\oplus \left[\ d, \dot{d} \;\middle|\; \begin{array}{l} d^+ = 0 \\ \dot{d}^+ = 1 \end{array} \right] \gg \left(\begin{array}{c|c} x & -1000 \leq x \leq 0 \\ \dot{x} & 40 \leq \dot{x} \leq 52 \\ r & r = 90 \\ \dot{r} & \dot{r} = 0 \\ d & d \leq 5 \\ \dot{d} & \dot{d} = 1 \end{array} \right) \triangleright c_2(\text{lower})$$

$$\odot \partial_H(T^{near} \parallel\ [\ \dot{r} \mid \dot{r}^+ = -20\ ] \gg G^{dn} \parallel C^{idle})$$

The controller either sends the *lower* signal to the gate immediately or delays till maximum $5s$ according to the flow predicate $d \leq 5 \wedge \dot{d} = 1$ before sending the *lower* signal. The specification then continues with process $\partial_H(T^{near} \parallel \ [\,\dot r \mid \dot r^+ = -20\,] \gg G^{dn} \parallel C^{idle})$.

$$\partial_{\mathbf{H}}(\mathbf{T}^{\mathbf{near}} \parallel \ [\,\dot{\mathbf{r}} \mid \dot{\mathbf{r}}^+ = -\mathbf{20}\,] \gg \mathbf{G}^{\mathbf{dn}} \parallel \mathbf{C}^{\mathbf{idle}})$$

$$= \left[\begin{array}{c|c} \dot r & \begin{array}{c} r^- = 90 \\ \dot r^+ = -20 \\ -1000 \le x^- \le -740 \end{array} \end{array}\right] \gg \left(\begin{array}{c|c} x & -1000 \le x \le 0 \\ \dot x & 40 \le \dot x \le 52 \\ r & 0 \le r \le 90 \\ \dot r & \dot r = -20 \end{array}\right) \rhd$$

$$\left[\begin{array}{c} r^- = 0 \\ -820 \le x^- \le -506 \end{array}\right] \gg [\,\dot r \mid \dot r = 0\,] \gg \left(\begin{array}{c|c} x & -1000 \le x \le 0 \\ \dot x & 40 \le \dot x \le 52 \\ r & r = 0 \\ \dot r & \dot r = 0 \end{array}\right) \rhd$$

$$\left[\begin{array}{c} r^- = 0 \\ x^- = 0 \end{array}\right] \gg \left(\begin{array}{c|c} x & 0 \le x \le 100 \\ \dot x & 40 \le \dot x \le 52 \\ r & r = 0 \\ \dot r & \dot r = 0 \end{array}\right) \rhd \ [\, x^- = 100 \,] \gg c_1(\text{exit})$$

$$\odot \partial_H(\left[\begin{array}{c|c} x & x^+ \le -1400 \\ \dot x & 48 \le \dot x^+ \le 52 \end{array}\right] \gg T^{far} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot d & \dot d^+ = 1 \end{array}\right] \gg C^{up} \parallel G^{cl})$$

The gate is lowering and a train is meanwhile approaching the gate. When the gate closes, the train is at least at a distance of 506 m from the gate. The train crosses the gate ($x = 0$), reaches the second detector placed at 100m forward from the gate and sends an *exit* signal to the controller.

We now expand the expression $\partial_H(\left[\begin{array}{c|c} x & x^+ \le -1400 \\ \dot x & 48 \le \dot x^+ \le 52 \end{array}\right] \gg T^{far} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot d & \dot d^+ = 1 \end{array}\right] \gg C^{up} \parallel G^{cl})$.

$$\partial_{\mathbf{H}}(\left[\begin{array}{c|c} x & x^+ \le -1400 \\ \dot x & 48 \le \dot x^+ \le 52 \end{array}\right] \gg \mathbf{T}^{\mathbf{far}} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot d & \dot d^+ = 1 \end{array}\right] \gg \mathbf{C}^{\mathbf{up}} \parallel \mathbf{G}^{\mathbf{cl}})$$

$$= \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot d & \dot d^+ = 1 \\ & r^- = 0 \\ & \dot r^- = 0 \end{array}\right] \gg c_2(raise) \odot \left[\begin{array}{c|c} \dot r & \dot r^+ = 20 \\ x & x^+ \le -1400 \\ \dot x & 48 \le \dot x^+ \le 52 \\ & r^- = 0 \end{array}\right] \gg$$

$$\left(\begin{array}{c|c} x & x \le -1000 \\ \dot x & 48 \le \dot x \le 52 \\ r & 0 \le r \le 90 \\ \dot r & \dot r = 20 \end{array}\right) \rhd \left[\begin{array}{c} x^- \le -1166 \\ r^- = 90 \end{array}\right] \gg \begin{array}{l} \partial_H(T^{far} \parallel \\ \quad [\,\dot r \mid \dot r^+ = 0\,] \gg G^{op} \parallel C^{idle}) \end{array}$$

(The expansion of $\partial_H(\left[\begin{array}{c|c} x & x^+ \le -1400 \\ \dot x & 48 \le \dot x^+ \le 52 \end{array}\right] \gg T^{far} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot d & \dot d^+ = 1 \end{array}\right] \gg C^{up} \parallel G^{cl})$ is continued.)

$$\oplus \left[\begin{array}{c|l} & r^- = 0 \wedge \dot{r}^- = 0 \\ x & x^+ \leq -1400 \\ \dot{x} & 48 \leq \dot{x}^+ \leq 52 \\ d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \end{array}\right] \gg \left(\begin{array}{c|c} x & x \leq -1000 \\ \dot{x} & 48 \leq \dot{x} \leq 52 \\ r & r = 0 \\ \dot{r} & \dot{r} = 0 \\ d & 0 \leq d \leq 5 \\ \dot{d} & \dot{d} = 1 \end{array}\right) \rhd$$

$$[\ x^- \leq -1140\ ] \gg c_2(raise) \odot [\ \dot{r} \mid \dot{r}^+ = 20\ ] \gg \left(\begin{array}{c|c} x & x \leq -1000 \\ \dot{x} & 48 \leq \dot{x} \leq 52 \\ r & 0 \leq r \leq 90 \\ \dot{r} & \dot{r} = 20 \end{array}\right) \rhd$$

$$\begin{array}{l} (\\ [\ r^- = 90\ ] \gg \partial_H(T^{far} \parallel C^{idle} \parallel G^{op}) \oplus [\ x^- = -1000\ ] \gg c_1(appr) \odot \\ \partial_H(T^{near} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \end{array}\right] \gg C^{dn} \parallel G^{up}) \\ ) \end{array}$$

The controller either sends the *raise* signal to the gate immediately or delays till maximum $5s$. If the *raise* signal is sent immediately, then the specification continues with expression $\partial_H(T^{far} \parallel [\ \dot{r} \mid \dot{r}^+ = 0\ ] \gg G^{op} \parallel C^{idle})$, whose expansion is similar to $\partial_H(\text{Trains} \parallel \text{Gate} \parallel \text{Cntr})$.

If the controller delays before sending the *raise* signal, then either the gate opens first or a new train arrives at the detector placed at 1000 m from the gate. If the gate opens first, the specification continues with the process $\partial_H(T^{far} \parallel C^{idle} \parallel G^{op})$ whose simplification is similar to $\partial_H(\text{Trains} \parallel \text{Gate} \parallel \text{Cntr})$. If a new train arrives first, the specification continues with the expression $\partial_H(T^{near} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \end{array}\right] \gg C^{dn} \parallel G^{up})$.

We now expand expression $\partial_H(T^{near} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \end{array}\right] \gg C^{dn} \parallel G^{up})$.

$$\partial_{\mathbf{H}}(\mathbf{T^{near}} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \end{array}\right] \gg \mathbf{C^{dn}} \parallel \mathbf{G^{up}})$$

$$= [\ r^- = 90\ ] \gg \partial_H(T^{near} \parallel \left[\begin{array}{c|c} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \end{array}\right] \gg C^{dn} \parallel [\ \dot{r} \mid \dot{r}^+ = 0\ ] \gg G^{op})$$

$$\oplus \left[\begin{array}{c|l} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \\ & 0 \leq r^- \leq 90 \\ & \dot{r}^- = 20 \end{array}\right] \gg c_2(lower) \odot \begin{array}{l} \partial_H(T^{near} \parallel \\ [\ \dot{r} \mid \dot{r}^+ = -20\ ] \gg G^{dn} \parallel C^{idle}) \end{array}$$

$$\oplus \left[\begin{array}{c|l} d & d^+ = 0 \\ \dot{d} & \dot{d}^+ = 1 \\ & \dot{r}^- = 20 \\ & -1000 \leq x^- \leq 0 \end{array}\right] \gg \left(\begin{array}{c|c} x & -1000 \leq x \leq 0 \\ \dot{x} & 48 \leq \dot{x} \leq 52 \\ r & 0 \leq r \leq 90 \\ \dot{r} & \dot{r} = 20 \\ d & 0 \leq d \leq 5 \\ \dot{d} & \dot{d} = 1 \end{array}\right) \rhd$$

$$\begin{array}{l} (\\ \ [\ x^- \leq -740\ ] \gg c_2(lower) \odot \partial_H(T^{near} \parallel C^{idle} \parallel [\ \dot{r} \mid \dot{r}^+ = -20\ ] \gg G^{dn}) \\ \oplus[r^- = 90] \gg \partial_H(T^{near} \parallel C^{dn} \parallel [\ \dot{r} \mid \dot{r}^+ = 0\ ] \gg G^{op}) \\ ) \end{array}$$

One of the two events can happen first: either the gate completely opens or the controller sends the *lower* signal to the gate. The specification now exhibits repeating patterns.

Verifying the safety condition

From the specification, it is obvious how the system behaves in different stages. While simplifying the specification, we take into account what can happen when two trains follow each other at minimum possible distance–i.e., a new train arrives at 1400 m backwards from the gate at the instant when the previous train crosses the second detector placed at 100 m forward from the gate.

The axioms for initially stateless bisimilarity in HyPA have been applied to those parts of the specification that are no longer within the scope of a parallel operator. These axioms make it possible to incorporate results obtained from real analysis of variable values into the process specification as it is being flattened. These results help in eliminating impossible options from the specification and hence greatly reducing its size. Also using these axioms, reinitialization clauses with empty sets of jumping variables, are added in the specification at different instances. These reinitialization clauses describe the values of variables such as the gate position or the distance of the train from the gate. Looking at these reinitialization clauses, it is easy to verify that the gate is closed when the train is at distance $506m$ or less from the gate. This was also confirmed by the simulator tool for HyPA specifications described in Section 2.7.

Inspite of the axiomatic strength of HyPA, the simplification of the specification by hand is a lengthy and error-prone procedure.

2.6.2 ϕ-Calculus:

Specification

The train gate controller is a parallel composition of the three processes starting in an empty environment.

$$\begin{aligned}&(\ \emptyset\ ,\\ &\nu\langle\text{appr,exit,raise,lower}\rangle(\mathit{Train} \parallel \mathit{Gate} \parallel \mathit{Cntr})\ )\end{aligned}$$

The channels "appr", "exit", "lower" and "raise" are made local to the process. It is also possible to define recursion variables of Train, Gate and Contr by using the replication operator of ϕ-calculus. We find the present approach of expressing recursion variables more readable.

The recursion variables $Train$, $Gate$ and $Cntr$ are specified below:

$$\begin{aligned}
&\mathit{Train}{=}\nu xy\ [\ (x,y) := (-1400,52)\ ]\ .T^{far}\\
&T^{far}\ =[\text{ reset } x,\dot{x},y \text{ with } \{x \mid x \leq -1000\},\\
&\qquad \{y \mid 48 \leq y \leq 52\},\{\dot{x} \mid \dot{x} = y\}]\\
&\qquad\quad .\ [\ x = -1000\ ]\ .\overline{appr}\,.T^{near}\\
&T^{near} = [\ \text{ reset } x{,}y \text{ with } \{x \mid -1000 \leq x \leq 0\},\\
&\qquad \{y \mid 40 \leq y \leq 52\}\ ]\ .\\
&\qquad [\ x = 0\ ]\ .T^{past}\\
&T^{past}\ = [\text{ reset } x \text{ with } \{x \mid 0 \leq x \leq 100\}\ ]\\
&\qquad .\ [\ x = 100\ ]\ .\\
&\qquad \overline{exit}\,.[(x,y) := (-1400,52)].T^{far}
\end{aligned}$$

$$\begin{aligned}
&\mathit{Gate}{=}\nu r\ [\ r := 90\ ]\ .G^{op}\\
&G^{op}\ = [\ \text{ reset } r{,}\dot{r} \text{ with } \{r \mid \dot{r} = 0\},\{r \mid r = 90\}\ ]\ .\\
&\qquad (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})\\
&G^{dn}\ = [\ \text{ reset } r{,}\dot{r} \text{ with } \{r \mid r \geq 0\},\{r \mid \dot{r} = -20\}\ ]\ .\\
&\qquad (\ [\ r = 0\ ]\ .G^{cl} + \text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{up})\\
&G^{up}\ = [\text{ reset } r{,}\dot{r} \text{ with } \{r \mid \dot{r} = 20\},\{r \mid r \leq 90\}\ ]\ .\\
&\qquad (\ [\ r = 90\ ]\ .G^{op} + raise\,.\,G^{up} + lower\,.\,G^{dn})\\
&G^{cl}\ \ = [\ \text{ reset } r{,}\dot{r} \text{ with } \{r \mid \dot{r} = 0\},\{r \mid r = 0\}\ ]\ .\\
&\qquad\ (\text{lower}\,.\,G^{cl} + \text{raise}\,.\,G^{up})
\end{aligned}$$

$$\begin{aligned}
&\mathit{Cntr}{=}\nu d C^{idle}\\
&C^{idle}{=}(appr\,.\ [\ d := 0\ ]\\
&\qquad .C^{dn} + exit\,.\ [\ d := 0\ ]\ .C^{up})\\
&C^{dn}\ = [\text{ reset } d{,}\dot{d} \text{ with } \{\dot{d} \mid \dot{d} = 1\},\{d \mid d \leq 5\}\ ]\\
&\qquad .(\overline{lower}\,.\ [\text{ reset } d,\ \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle}\\
&\qquad\quad +exit\,.\,C^{dn} + appr\,.\,C^{dn})\\
&C^{up}\ = [\text{ reset } d{,}\dot{d} \text{ with } \{\dot{d} \mid \dot{d} = 1\},\{d \mid d \leq 5\}\ ]\\
&\qquad\ .(\overline{raise}\,.\ [\text{ reset } d,\ \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle}\\
&\qquad\quad +exit\,.\,C^{up} + appr\,.\ [\ d := 0\ ]\ .C^{dn})
\end{aligned}$$

In [RS03], the derivatives of variables are defined by ordinary differential equations with unique solutions. We are aware of a later version of ϕ-calculus [RS], in which a broader class of flows are used in models. Also Hybrid PROMELA, the input language to the model checker SPHIN, permits differential inclusions for flows. Therefore we take the liberty to write the train gate controller model as above.

Some observations about the specification are as follows:

- The variables are given specific values in the ϕ-calculus environment. Other process algebras allow their variables to initialize or jump to an arbitrary value satisfying a condition. For example, in HyPA specification, as the train crosses the $100m$ detector, x is updated according to the condition $x^+ \leq -1400$. In ϕ-calculus, x is updated to a particular value -1400. Resetting x according to the condition $x^+ \leq -1400$ represents that a second train may take arbitrarily long time to arrive after the first train has passed. One way to model this in ϕ-calculus specification is to arbitrarily delay the assignment of -1400 to x. This can be achieved by using the delay action prefix as follows:

  – First introduce a dummy channel a in the specification.

  – Rewrite the recursion variable T^{past} as follows:

$$\begin{array}{rcl} T^{past} & = & [\text{ reset } x \text{ with } \{x \mid 0 \leq x \leq 100\}\ ] \\ & & .\ [\ x = 100\ ]\ .\overline{exit}\,. \\ & & [\ y := 0\ ]\ .\ [\text{ reset } y \text{ with } \{y \mid y = 0\}\ ] \\ & & \delta\,.\,a\,.\ [\ (x,y) := (-1400, 52)\ ]\ .T^{far} \end{array}$$

  – Put $!\bar{a}$ in parallel with the train gate controller specification. The train gate controller specification is recursive. Therefore we put infinitely many instances of the receive action $\bar{a}$ in parallel with the specification.

  The required process is as follows:

$$(\ \emptyset\ ,\ \nu\langle \text{appr,exit,raise,lower,a}\rangle(Train\ \mid\ Gate\ \mid\ Contr \mid !\bar{a})\ )$$

  The variable x varies with $\dot{x} = y$. We assign 0 to y and reset the flow constraint of y, in order to make the environment with valuation $x : 100$ delayable.

  The process term $\delta\,.\,a\,.\ [\ (x,y) := (-1400, 52)\ ]\ .T^{far}$ is arbitrarily delayable. It must discharge δ before the communication on channel a and the assignment $[\ (x,y) := (-1400, 52)\ ]$ can become urgent.

  This arrangement will have the same effect as initializing x according to the predicate $x^+ \leq -1400$. However the latter approach is much more intuitive.

- As soon as the controller has sent the *lower* or *raise* signal to the gate, the flow constraints $\{d \mid d \leq 5\}$ needs to be removed from the environment, to allow the environment to delay further. The approach we follow is to remove both $\{d \mid d \leq 5\}$ and $\{\dot{d} \mid \dot{d} = 1\}$ and replace it by a constraint $\{TRUE\}$.

Behaviour of the Specification

There are no axioms in ϕ-calculus. Therefore algebraic manipulation of the specification is not possible. The approach we adopt is to construct (part of) the state-space of the train

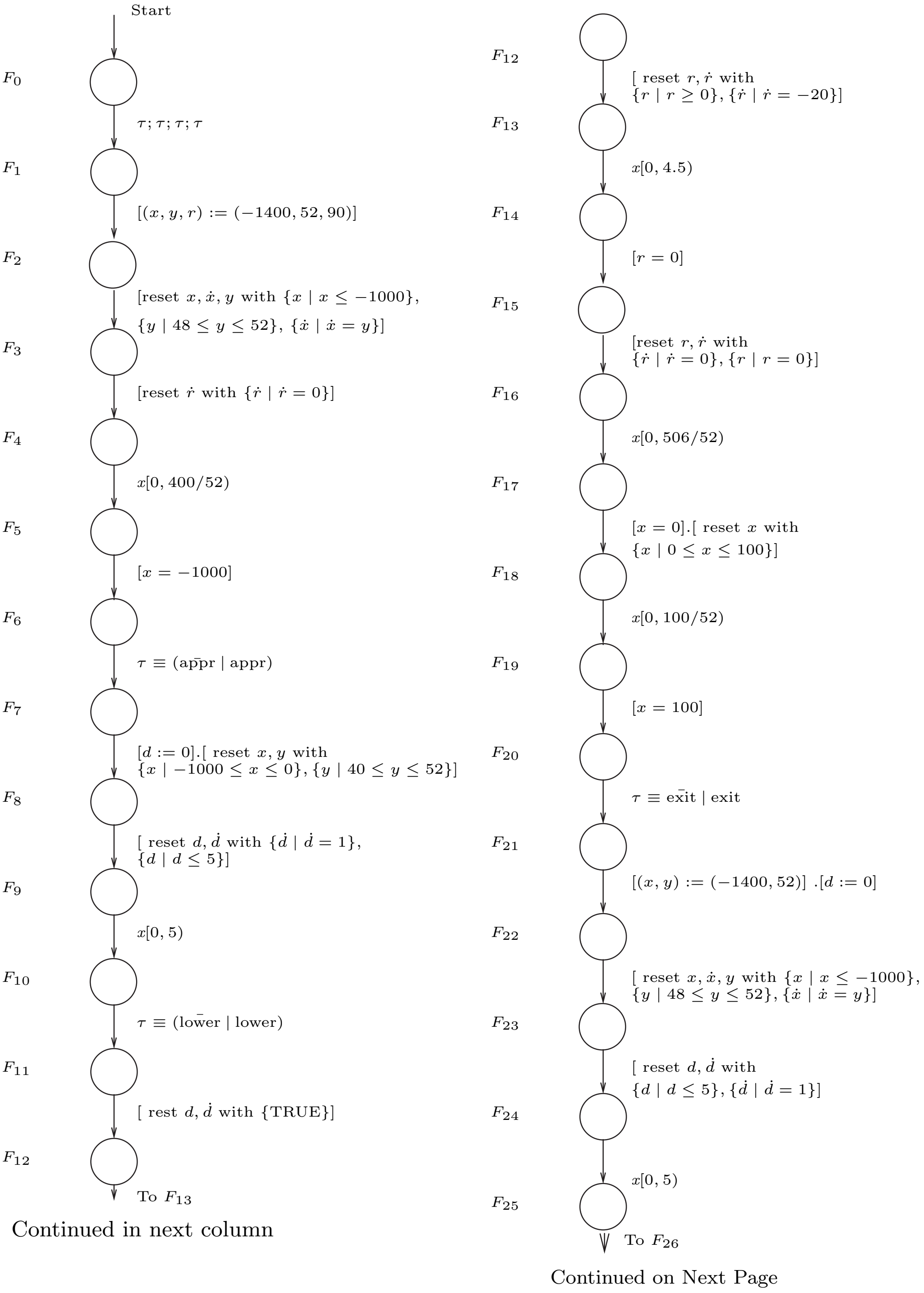
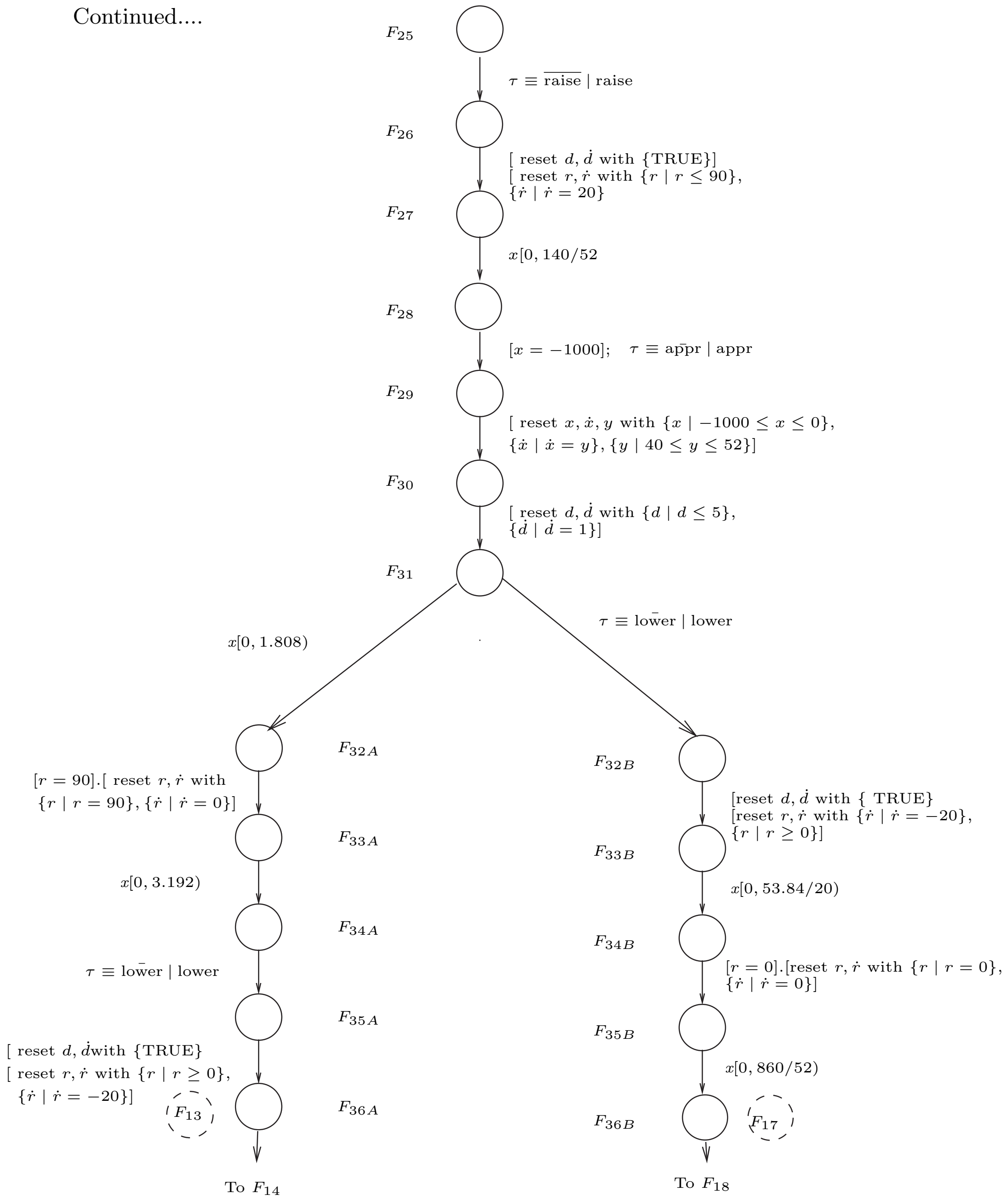

Start
F_0
$\tau; \tau; \tau; \tau$
F_1
$[(x, y, r) := (-1400, 52, 90)]$
F_2
[reset $x, \dot{x}, y$ with $\{x \mid x \leq -1000\}$,
$\{y \mid 48 \leq y \leq 52\}$, $\{\dot{x} \mid \dot{x} = y\}$]
F_3
[reset $\dot{r}$ with $\{\dot{r} \mid \dot{r} = 0\}$]
F_4
$x[0, 400/52)$
F_5
$[x = -1000]$
F_6
$\tau \equiv (\bar{\text{appr}} \mid \text{appr})$
F_7
$[d := 0]$.[reset x, y with
$\{x \mid -1000 \leq x \leq 0\}, \{y \mid 40 \leq y \leq 52\}$]
F_8
[reset $d, \dot{d}$ with $\{\dot{d} \mid \dot{d} = 1\}$,
$\{d \mid d \leq 5\}$]
F_9
$x[0, 5)$
F_{10}
$\tau \equiv (\bar{\text{lower}} \mid \text{lower})$
F_{11}
[rest $d, \dot{d}$ with {TRUE}]
F_{12}
To F_{13}
Continued in next column
F_{12}
[reset $r, \dot{r}$ with
$\{r \mid r \geq 0\}, \{\dot{r} \mid \dot{r} = -20\}$]
F_{13}
$x[0, 4.5)$
F_{14}
$[r = 0]$
F_{15}
[reset $r, \dot{r}$ with
$\{\dot{r} \mid \dot{r} = 0\}, \{r \mid r = 0\}$]
F_{16}
$x[0, 506/52)$
F_{17}
$[x = 0]$.[reset x with
$\{x \mid 0 \leq x \leq 100\}$]
F_{18}
$x[0, 100/52)$
F_{19}
$[x = 100]$
F_{20}
$\tau \equiv \bar{\text{exit}} \mid \text{exit}$
F_{21}
$[(x, y) := (-1400, 52)]$.$[d := 0]$
F_{22}
[reset $x, \dot{x}, y$ with $\{x \mid x \leq -1000\}$,
$\{y \mid 48 \leq y \leq 52\}, \{\dot{x} \mid \dot{x} = y\}$]
F_{23}
[reset $d, \dot{d}$ with
$\{d \mid d \leq 5\}, \{\dot{d} \mid \dot{d} = 1\}$]
F_{24}
$x[0, 5)$
F_{25}
To F_{26}
Continued on Next Page

Figure 2.3: State space of Train Gate Controller in ϕ-calculus

Continued....


F_{25}
$\tau \equiv \overline{\text{raise}} \mid \text{raise}$
F_{26}
[reset $d, \dot{d}$ with {TRUE}]
[reset $r, \dot{r}$ with $\{r \mid r \leq 90\}$,
$\{\dot{r} \mid \dot{r} = 20\}$
F_{27}
$x[0, 140/52$
F_{28}
$[x = -1000]$; $\tau \equiv \bar{\text{appr}} \mid \text{appr}$
F_{29}
[reset $x, \dot{x}, y$ with $\{x \mid -1000 \leq x \leq 0\}$,
$\{\dot{x} \mid \dot{x} = y\}, \{y \mid 40 \leq y \leq 52\}$]
F_{30}
[reset $d, \dot{d}$ with $\{d \mid d \leq 5\}$,
$\{\dot{d} \mid \dot{d} = 1\}$]
F_{31}
$x[0, 1.808)$
$\tau \equiv \bar{\text{lower}} \mid \text{lower}$
F_{32A}
$[r = 90]$.[reset $r, \dot{r}$ with
$\{r \mid r = 90\}, \{\dot{r} \mid \dot{r} = 0\}$]
F_{33A}
$x[0, 3.192)$
F_{34A}
$\tau \equiv \bar{\text{lower}} \mid \text{lower}$
F_{35A}
[reset $d, \dot{d}$with {TRUE}
[reset $r, \dot{r}$ with $\{r \mid r \geq 0\}$,
$\{\dot{r} \mid \dot{r} = -20\}$]
F_{13}
F_{36A}
To F_{14}
F_{32B}
[reset $d, \dot{d}$ with { TRUE}
[reset $r, \dot{r}$ with $\{\dot{r} \mid \dot{r} = -20\}$,
$\{r \mid r \geq 0\}$]
F_{33B}
$x[0, 53.84/20)$
F_{34B}
$[r = 0]$.[reset $r, \dot{r}$ with $\{r \mid r = 0\}$,
$\{\dot{r} \mid \dot{r} = 0\}$]
F_{35B}
$x[0, 860/52)$
F_{36B}
F_{17}
To F_{18}

gate controller in ϕ-calculus and do reachability analysis. The derivation of the state-space is a very lengthy procedure. It is given in the Appendix A.We give a schematic diagram of the state space in Figure 2.3.

The names of the states in Figure 2.3, correspond to the states in the state-space given in Appendix A.

Verifying the safety condition

We have used the semantics of ϕ-calculus to obtain a subset of the transition system of the train gate controller specification. While applying the transition rules, we have to make certain choices as we cannot cover all possible scenarios. For example there are infinitely many time transitions of duration less than or equal to five. We select a scenario with boundary conditions on variables. For example,

1. We choose the **maximum** possible delay of the controller, i.e. $5s$. This can be seen in the transitions from F_9 to F_{10}, F_{24} to F_{25}, and F_{31} to F_{34A} (summing up the delays $1.808 + 3.192$).
2. We choose the **maximum** train speed (i.e. $52m/s$). This can be seen in the transitions from F_4 to F_5 and F_{16} to F_{17}.
3. We also take the scenario when the second train arrives at distance -1400 from the gate (see transitions, F_{20} to F_{21}) as soon as the first one crosses the $100m$ detector (see transitions F_{19} to F_{20}).

We obtain the same result as in HYPA, that the gate is closed whenever the train is at a distance of 506 m from the gate or less.

2.6.3 Hybrid χ

In this section, we give the Hybrid χ specification of the train gate controller. We eliminate the parallel operator from the specification using an elimination theorem given in [SM06]. Finally, we comment on the results obtained.

Specification

The train gate controller process in Hybrid χ is specified as follows (the recursion variables T^{far}, G^{op} and C^{idle} are defined separately):

$$\begin{array}{l}
\langle\ \partial_{Aia}\ (\upsilon_{\{appr,exit,raise,lower\}} \\
\qquad\qquad (\ Train \parallel Gate \parallel Controller \\
\qquad\qquad)\ ), \\
\qquad\qquad \{x \mapsto -1400, y \mapsto -52, r \mapsto 90, d \mapsto 0, \mathrm{time} \mapsto 0\}, \\
\qquad\qquad (\\
\qquad\qquad\quad \{x, y, r, d\}, \emptyset, \emptyset, \\
\qquad\qquad\quad \{appr, exit, raise, lower\}, \\
\qquad\qquad\quad \{\ Train \mapsto T^{far}, \\
\qquad\qquad\quad\quad Gate \mapsto G^{op}, \\
\qquad\qquad\quad\quad Controller \mapsto C^{idle}\} \\
\qquad\qquad) \\
\rangle
\end{array}$$

Recall from Section 2.2.3, that a process in Hybrid χ is a triplet $\langle p, \nu, E\rangle$, where p is a process term, ν is a variable valuation and E is a Hybrid χ environment.

There are four environment variables in the specification. All are declared continuous. The derivative of x which represents the speed of the train, can have discontinuous trajectory. Also $\dot{x}$ can jump during communication actions. In order to avoid these discontinuities, a continuous variable y is declared and the delay predicate $y = \dot{x}$ is added to the train specification. The set of action labels Aia, includes send and receive action labels on channels "appr", "exit", "raise" and "lower". The encapsulation operator ∂_{Aia} blocks individual send and receive on these channels.

$$
\begin{array}{lcl}
T^{far} & = & x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
 & & \;[\!]\; x = -1000 \rightarrow \mathit{appr}!!\,;\; T^{near} \\
T^{near} & = & -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
 & & \;[\!]\; x = 0 \rightarrow \emptyset : \mathrm{true} \gg \mathit{pass}\,;\; T^{past} \\
T^{past} & = & 0 \leq x \leq 100 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
 & & \;[\!]\; x = 100 \rightarrow \mathit{exit}!!\,; \\
 & & \{x, y\} : x \leq -1400 \gg \tau\,;\; T^{far}
\end{array}
$$

$$
\begin{array}{lcl}
G^{op} & = & \dot{r} = 0 \wedge r = 90 \;[\!]\; [\mathit{raise}??]\,;\; G^{op} \\
 & & \;[\!]\; [\mathit{lower}??]\,;\; G^{dn} \\
G^{dn} & = & r \geq 0 \wedge \dot{r} = -20 \\
 & & \;[\!]\; r = 0 \rightarrow \emptyset : \mathrm{true} \gg \mathrm{ready\_}\mathit{dn}\,;\; G^{cl} \\
 & & \;[\!]\; [\mathit{lower}??]\,;\; G^{dn} \;[\!]\; [\mathit{raise}??]\,;\; G^{up} \\
G^{up} & = & r \leq 90 \wedge \dot{r} = 20 \\
 & & \;[\!]\; r = 90 \rightarrow \emptyset : \mathrm{true} \gg \mathrm{ready\_}\mathit{up}\,;\; G^{op} \\
 & & \;[\!]\; [\mathit{raise}??]\,;\; G^{up} \;[\!]\; [\mathit{lower}??]\,;\; G^{dn} \\
G^{cl} & = & \dot{r} = 0 \wedge r = 0 \;[\!]\; [\mathit{lower}??]\,;\; G^{cl} \\
 & & \;[\!]\; [\mathit{raise}??]\,;\; G^{up}
\end{array}
$$

$$
\begin{array}{lcl}
C^{idle} & = & [\mathit{appr}??]\,;\; \{d\} : d = 0 \gg \tau\,;\; C^{dn} \\
 & & \;[\!]\; [\mathit{exit}??]\,;\; \{d\} : d = 0 \gg \tau\,;\; C^{up} \\
C^{dn} & = & d \leq 5 \wedge \dot{d} = 1 \;[\!]\; [\mathit{lower}!!]\,;\; C^{idle} \\
 & & \;[\!]\; [\mathit{appr}??]\,;\; C^{dn} \;[\!]\; [\mathit{exit}??]\,;\; C^{dn} \\
C^{up} & = & d \leq 5 \wedge \dot{d} = 1 \;[\!]\; [\mathit{raise}!!]\,;\; C^{idle} \\
 & & \;[\!]\; [\mathit{appr}??]\,;\; \{d\} : d = 0 \gg \tau\,;\; C^{dn} \\
 & & \;[\!]\; [\mathit{exit}??]\,;\; C^{up}
\end{array}
$$

We note that in Hybrid χ , in addition to the send and receive actions, there are actions with labels *pass, ready_dn, ready_up.* We mentioned earlier (in section 2.2.3), that delay predicates in Hybrid χ represent infinite behaviour and strong time determinism of Hybrid χ plays an important role in the termination of delays. Delays are placed in alternative composition with guards or delayable action predicates. Actions with labels *pass, ready_dn, ready_up* are necessary in their respective places to terminate delays and resolve choices in alternative compositions. The $\emptyset$ in action predicates, for example in $\emptyset : \mathrm{true} \gg \mathrm{ready\_}\mathit{dn}$, indicates that no variables are allowed to jump.

Simplifying the specification

We simplify the train gate controller system using an elimination theorem given in [SM06]. The linear form obtained from the elimination algorithm is large and contains many unreachable options. These options cannot be eliminated algebraically from the linear form. During linearization, we use real analysis and the structural operational semantics of Hybrid χ to reason about the possible range of variable values at an instant. Using that, we do not expand unreachable options present in the linear form. For example, we do not expand the options with action pass in expressions G_2, G_3, G_4, G_{13} and G_{14} given below. From real analysis, we know that the train takes longer to reach the gate (i.e., when x is zero), than the process terms in these expressions can wait. Similarly, options with actions ready_{up} in G_{11} and $ca(appr, e_0, x_0)$ in G_{10} are not expanded as the process terms in G_{10}, G_{11} cannot do the delay required for the guards guarding the actions to become true.

We identify the process terms and their bisimilar forms obtained from the elimination theorem as $\mathbf{G_1}$, $\mathbf{G_2}$, $\mathbf{G_3}$, etc.

$$
\begin{array}{lll}
\mathbf{G_1}: & \partial_{Aia}(T^{far} \parallel G^{op} \parallel C^{idle}) & \mathrel{\underline{\leftrightarrow}} \dot{r} = 0 \wedge r = 90 \\
 & & \;[\!]\; x \leq -1000 \leq 0 \wedge 48 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
 & & \;[\!]\; [\; x = -1000 \rightarrow ca(appr, e_0, x_0)\;]\;; \\
 & & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \parallel G^{op} \parallel T^{near}) \\
 & & \;[\!]\; x = 1000 \rightarrow \delta \\
 & & \\
 & & (\text{where } \partial_{Aia}(appr!!;\; (C^{idle} \parallel G^{op} \parallel T^{near})) \mathrel{\underline{\leftrightarrow}} \delta)
\end{array}
$$

$$
\begin{array}{lll}
\mathbf{G_2}: & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \parallel & \\
 & G^{op} \parallel T^{near}) & \mathrel{\underline{\leftrightarrow}} -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
 & & \;[\!]\; \dot{r} = 0 \wedge r = 90 \\
 & & \;[\!]\; \{d\} : d = 0 \gg \tau; \\
 & & \partial_{Aia}(C^{dn} \parallel G^{op} \parallel T^{near}) \\
 & & \;[\!]\; x = 0 \rightarrow \emptyset : \text{true} \gg \text{pass}; \\
 & & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \parallel G^{op} \parallel T^{past})
\end{array}
$$

$$
\begin{array}{lll}
\mathbf{G_3}: & \partial_{Aia}(C^{dn} \parallel G^{op} \parallel T^{near}) & \mathrel{\underline{\leftrightarrow}} -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
 & & \;[\!]\; \dot{r} = 0 \wedge r = 90 \\
 & & \;[\!]\; \dot{d} = 1 \wedge d \leq 5 \\
 & & \;[\!]\; [ca(lower, e_0, x_0)]\;; \\
 & & \partial_{Aia}(C^{idle} \parallel G^{dn} \parallel T^{near}) \\
 & & \;[\!]\; x = 0 \rightarrow \emptyset : \text{true} \gg \text{pass}; \\
 & & \partial_{Aia}(C^{dn} \parallel G^{op} \parallel T^{past})
\end{array}
$$

$$\begin{array}{lll}
\mathbf{G_4}: & \partial_{Aia}(C^{idle} \parallel G^{dn} \parallel T^{near}) & \leftrightarrows -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
& & \;[]\; \dot{r} = -20 \wedge r \geq 0 \\
& & \;[]\; r = 0 \rightarrow (\emptyset : \mathrm{true} \gg \mathrm{ready\_}dn; \\
& & \partial_{Aia}(C^{idle} \parallel G^{cl} \parallel T^{near})) \\
& & \;[]\; x = 0 \rightarrow \emptyset : \mathrm{true} \gg \mathrm{pass}; \\
& & \partial_{Aia}(C^{idle} \parallel G^{dn} \parallel T^{past})
\end{array}$$

$$\begin{array}{lll}
\mathbf{G_5}: & \partial_{Aia}(C^{idle} \parallel G^{cl} \parallel T^{near}) & \leftrightarrows -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
& & \;[]\; r = 0 \wedge \dot{r} = 0 \\
& & \;[]\; x = 0 \rightarrow (\emptyset : \mathrm{true} \gg \mathrm{pass}; \\
& & \partial_{Aia}(C^{idle} \parallel G^{cl} \parallel T^{past}))
\end{array}$$

$$\begin{array}{lll}
\mathbf{G_6}: & \partial_{Aia}(C^{idle} \parallel G^{cl} \parallel T^{past}) & \leftrightarrows 0 \leq x \leq 100 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
& & \;[]\; r = 0 \wedge \dot{r} = 0 \\
& & \;[]\; \;[\; x = 100 \rightarrow ca(exit, e_0, x_0) \;]\; ; \\
& & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\ C^{up} \parallel G^{cl} \parallel \\
& & \{x, y\} : x \leq -1400 \gg \tau;\ T^{far}) \\
& & \;[]\; x = 100 \rightarrow \delta
\end{array}$$

$$\begin{array}{lll}
\mathbf{G_7}: & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\ C^{up} \parallel G^{cl} & \\
& \parallel \{x, y\} : x \leq -1400 \gg \tau;\ T^{far}) & \leftrightarrows \dot{r} = 0 \wedge r = 0 \\
& & \;[]\; \{d\} : d = 0 \gg \tau; \\
& & \partial_{Aia}(C^{up} \parallel G^{cl} \parallel \{x, y\} : x \leq -1400 \gg \tau;\ T^{far})) \\
& & \;[]\; \{x, y\} : x \leq -1400 \gg \tau; \\
& & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\ C^{up} \parallel G^{cl} \parallel T^{far})
\end{array}$$

$$\begin{array}{lll}
\mathbf{G_8}: & \partial_{Aia}(\{d\} : d = 0 \gg \tau;\ C^{up} & \\
& \parallel G^{cl} \parallel T^{far}) & \leftrightarrows \dot{r} = 0 \wedge r = 0 \\
& & \;[]\; x \leq -1400 \wedge 48 \leq \dot{x} \wedge y = \dot{x} \leq 52 \\
& & \;[]\; \{d\} : d = 0 \gg \tau; \\
& & \partial_{Aia}(C^{up} \parallel G^{cl} \parallel T^{far}) \\
& & \;[]\; x = -1000 \rightarrow \delta
\end{array}$$

$$
\begin{aligned}
&\mathbf{G_9}: \quad \partial_{Aia}(C^{up} \parallel G^{cl} \parallel \\
&\quad (\{x,y\} : x \leq -1400 \gg \tau\,; \\
&\quad T^{far})) \qquad \underline{\leftrightarrow}\, \dot{r} = 0 \wedge r = 0 \\
&\qquad [] \;\, d \leq 5 \wedge \dot{d} = 1 \\
&\qquad [] \;\, [\; ca(raise, e_0, x_0)\;] \;;\; \partial_{Aia}(C^{idle} \parallel G^{up} \\
&\qquad \parallel \{x,y\} : x \leq -1400 \gg \tau;\; T^{far})) \\
&\qquad [] \;\, \{x,y\} : x \leq -1400 \gg \tau\,; \\
&\qquad \partial_{Aia}(C^{up} \parallel G^{cl} \parallel T^{far})
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{G_{10}}: \quad \partial_{Aia}(C^{up} \parallel G^{cl} \parallel T^{far}) \qquad \underline{\leftrightarrow}\, x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
&\qquad [] \;\, d \leq 5 \wedge \dot{d} = 1 \\
&\qquad [] \;\, [x = -1000 \rightarrow ca(appr, e_0, x_0)]; \\
&\qquad \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \parallel G^{cl} \parallel T^{near}) \\
&\qquad [] \;\, r = 0 \wedge \dot{r} = 0 \\
&\qquad [] \;\, [\; ca(raise, e_0, x_0)\;] \;;\; \partial_{Aia}(C^{idle} \parallel G^{up} \\
&\qquad \parallel T^{far}) \\
&\qquad [] \;\, x = -1000 \rightarrow \delta
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{G_{11}}: \quad \partial_{Aia}(C^{idle} \parallel G^{up} \parallel \\
&\quad (\{x,y\} : x \leq -1400 \gg \tau\,; \\
&\quad T^{far})) \qquad \underline{\leftrightarrow}\, r \leq 90 \wedge \dot{r} = 20 \\
&\qquad [] \;\, \{x,y\} : x \leq -1400 \gg \tau\,; \\
&\qquad \partial_{Aia}(C^{idle} \parallel G^{up} \parallel T^{far}) \\
&\qquad [] \;\, r = 90 \rightarrow \emptyset : \text{true} \gg \text{ready}_{\text{up}}\,; \\
&\qquad \partial_{Aia}(C^{idle} \parallel G^{op} \parallel \\
&\qquad (\{x,y\} : x \leq -1400 \gg \tau;\; T^{far}))
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{G_{12}}: \quad \partial_{Aia}(C^{idle} \parallel G^{up} \parallel T^{far}) \qquad \underline{\leftrightarrow}\, x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
&\qquad [] \;\, [\; x = -1000 \rightarrow ca(appr, e_0, x_0)\;]\;; \\
&\qquad \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \parallel G^{up} \parallel T^{near}) \\
&\qquad [] \;\, r \leq 90 \wedge \dot{r} = 20 \\
&\qquad [] \;\, r = 90 \rightarrow (\emptyset : \text{true} \gg \text{ready\_up}; \\
&\qquad \partial_{Aia}(C^{idle} \parallel G^{op} \parallel T^{far}))
\end{aligned}
$$

See $\mathbf{G_1}$ for $\partial_{Aia}(C^{idle} \parallel G^{op} \parallel T^{far})$.

$$
\begin{aligned}
&\mathbf{G_{13}}: \quad \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \\
&\quad \parallel G^{up} \parallel T^{near}) \qquad \underline{\leftrightarrow}\, r \leq 90 \wedge \dot{r} = 20 \\
&\qquad [] \;\, -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
&\qquad [] \;\, \{d\} : d = 0 \gg \tau;\; \partial_{Aia}(C^{dn} \parallel G^{up} \parallel T^{near}) \\
&\qquad [] \;\, r = 90 \rightarrow \emptyset : \text{true} \gg \text{ready\_up}; \\
&\qquad \partial_{Aia}(\{d\} : d = 0 \gg \tau;\; C^{dn} \parallel G^{op} \parallel T^{near}) \\
&\qquad [] \;\, x = 0 \rightarrow \text{pass}; \\
&\qquad \partial_{Aia}(T^{past} \parallel G^{up} \parallel \{d\} : d = 0 \gg \tau;\; C^{dn})
\end{aligned}
$$

See $\mathbf{G_2}$ for $\partial_H(\{d\} : d = 0 \gg \tau;\ C^{dn} \parallel G^{op} \parallel T^{near})$.

$$
\begin{aligned}
\mathbf{G_{14}}:\quad \partial_{Aia}(C^{dn} \parallel G^{up} \parallel T^{near}) \ \leftrightarrows\ & \dot{r} = 20 \wedge r \leq 90 \\
& [\!]\ -1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge y = \dot{x} \\
& [\!]\ d \leq 5 \wedge \dot{d} = 1 \\
& [\!]\ r = 90 \rightarrow \emptyset : \text{true} \gg \text{ready\_}up; \\
& \partial_{Aia}(G^{op} \parallel C^{dn} \parallel T^{near}) \\
& [\!]\ \ [\ ca(lower, e_0, x_0)\ ]\ ; \\
& \partial_{Aia}(C^{idle} \parallel G^{dn} \parallel T^{near}) \\
& [\!]\ x = 0 \rightarrow \emptyset : \text{true} \gg \text{pass}; \\
& (T^{past} \parallel G^{up} \parallel C^{dn})
\end{aligned}
$$

See $\mathbf{G_3}$ $\partial_H(C^{dn} \parallel G^{op} \parallel T^{near})$.
See $\mathbf{G_4}$ for $\partial_H(C^{idle} \parallel G^{dn} \parallel T^{near})$.

Verification of safety condition

We can use reachability analysis to eliminate impossible options in the linearized specification and can reason about the semantics of the linearized process to conclude that the gate is indeed closed when the train is within 506 m from it. The Hybrid χ simulator confirms this result.

We find the elimination theorem useful in simplifying the train gate controller specification. As mentioned before, at different stages we make use of the operational semantics to resolve choices in alternative composition. A more comprehensive set of axioms of Hybrid χ together with the elimination theorem can make the whole procedure algebraic. A weaker notion of bisimulation than state-less bisimulation is needed to incorporate axioms for real analysis of systems, similar to the ones present in HyPA and ACP^{srt}_{hs}.

At the time of writing the report [Kha06], the elimination theorem of [SM06] was the only result available for eliminating a parallel operator from a Hybrid χ specification. Now more results are available as mentioned in [The06] and Chapter 4. The train gate controller specification in Hybrid χ was also linearized using the linearization tool of [The06]. The resulting specification was very large with about a hundred and seventy recursion variables and is not suitable for manual inspection. The algorithm mentioned in Chapter 4, proposes a technique for keeping the size of the resulting linear form small. This algorithm has not yet been implemented.

2.6.4 ACP^{srt}_{hs}

This section is structured as follows: First we give the specification of the train gate controller system in ACP^{srt}_{hs}; then we expand the specification using the axioms and lifting rules of ACP^{srt}_{hs}; and in the end we analyze the expanded specification for results about the safety condition.

Specification

The train gate controller system is specified as follows:

$$\partial_H(Trains \parallel Cntr \parallel Gate),$$

where

$$\begin{aligned} H &= \{\widetilde{\widetilde{s(d)}} \mid d \in \{\text{appr}, \text{exitraise}, \text{lower}\}\} \\ &\quad \cup\{\widetilde{\widetilde{r(d)}} \mid d \in \{\text{appr}, \text{exit}, \text{raise}, \text{lower}\}\} \end{aligned}$$

The communication function γ is given below:

$$\begin{aligned} \gamma(\widetilde{\widetilde{s_i(d)}}, \widetilde{\widetilde{r_i(d)}}) &= \widetilde{\widetilde{c_i(d)}},\ \text{where } d \in \{\text{appr}, \text{exit}, \text{raise}, \text{lower}\} \\ & \quad i \in \{1, 2\} \end{aligned}$$

The recursion variables $Train$, $Gate$ and $Cntr$ are defined as follows:

$$\begin{aligned} \text{Trains} &= (x \leq -1400) \wedge\!\!\blacktriangle\, T^{far} \\ T^{far} &= (x \leq -1000 \wedge 48 \leq \dot{x} \leq 52) \ulcorner\!\blacktriangledown_{\{x\}} \\ &\quad \sigma^*_{\mathsf{rel}}((x = -1000) :\rightarrow \\ &\quad\quad (x^{\bullet} = {}^{\bullet}x \wedge \dot{x}^{\bullet} = {}^{\bullet}\dot{x}) \ulcorner\!\blacktriangledown\, s_1(\widetilde{\widetilde{\text{appr}}}) \cdot T^{near}) \\ T^{near} &= (-1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52) \ulcorner\!\blacktriangledown_{\{x\}} \\ &\quad \sigma^*_{\mathsf{rel}}((x = 0) :\rightarrow \\ &\quad\quad (x^{\bullet} = {}^{\bullet}x \wedge \dot{x}^{\bullet} = {}^{\bullet}\dot{x}) \ulcorner\!\blacktriangledown\, \widetilde{\widetilde{\text{pass}}} \cdot T^{past}) \\ T^{past} &= (0 \leq x \leq 100 \wedge 40 \leq \dot{x} \leq 52) \ulcorner\!\blacktriangledown_{\{x\}} \\ &\quad \sigma^*_{\mathsf{rel}}((x = 100) :\rightarrow \\ &\quad\quad (x^{\bullet} \leq -1400) \ulcorner\!\blacktriangledown\, s_1(\widetilde{\widetilde{\text{exit}}})) \cdot T^{far} \end{aligned}$$

$$\begin{aligned} Gate &= (r = 90) \wedge\!\!\blacktriangle\, G^{op}, \\ G^{op} &= (r = 90 \wedge \dot{r} = 0) \ulcorner\!\blacktriangledown_{\{r\}} \\ &\quad (\sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{lower}}}) \cdot G^{dn}) \\ &\quad\quad + \sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{raise}}}) \cdot G^{op})) \\ G^{dn} &= (0 \leq r \leq 90 \wedge \dot{r} = -20) \ulcorner\!\blacktriangledown_{\{r\}} \\ &\quad (\sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{lower}}}) \cdot G^{dn}) \\ &\quad\quad + \sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{raise}}}) \cdot G^{up}) + \\ &\quad\quad \sigma^*_{\mathsf{rel}}((r = 0) :\rightarrow ((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, (\widetilde{\widetilde{ready_{dn}}}) \cdot G^{cl}))) \\ G^{up} &= (0 \leq r \leq 90 \wedge \dot{r} = 20) \ulcorner\!\blacktriangledown_{\{r\}} \\ &\quad (\sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{lower}}}) \cdot G^{dn}) \\ &\quad\quad + \sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{raise}}}) \cdot G^{up}) + \\ &\quad\quad \sigma^*_{\mathsf{rel}}((r = 90) :\rightarrow ((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, (\widetilde{\widetilde{ready_{up}}}) \cdot G^{op}))) \\ G^{cl} &= (r = 0 \wedge \dot{r} = 0) \ulcorner\!\blacktriangledown_{\{r\}} \\ &\quad (\sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{lower}}}) \cdot G^{cl}) \\ &\quad\quad + \sigma^*_{\mathsf{rel}}((r^{\bullet} = {}^{\bullet}r) \ulcorner\!\blacktriangledown\, r_2(\widetilde{\widetilde{\text{raise}}}) \cdot G^{up})) \end{aligned}$$

$$
\begin{aligned}
Contr &= (d=0) \wedge\!\!\blacktriangle\, C^{idle} \\[1ex]
C^{idle} &= (\dot{d}=0) \curvearrowright\!\!\blacktriangledown_{\{d\}} \\
&\quad (\sigma^{*}_{\mathsf{rel}}((d^{\bullet}=0) \ulcorner\!\!\blacktriangledown\, r_1(\widetilde{\widetilde{\widetilde{\mathrm{appr}}}}) \cdot C^{dn}) \\
&\quad\ +\sigma^{*}_{\mathsf{rel}}((d^{\bullet}=0) \ulcorner\!\!\blacktriangledown\, r_1(\widetilde{\widetilde{\widetilde{\mathrm{exit}}}}) \cdot C^{up})) \\[1ex]
C^{up} &= (0 \le d \le 5 \wedge \dot{d}=1) \curvearrowright\!\!\blacktriangledown_{\{d\}} \\
&\quad (\sigma^{*}_{\mathsf{rel}} \\
&\quad ((d^{\bullet}=0) \ulcorner\!\!\blacktriangledown\, r_1(\widetilde{\widetilde{\widetilde{\mathrm{appr}}}}) \cdot C^{\mathrm{dn}}) \\
&\quad\ \sigma^{*}_{\mathsf{rel}}((d^{\bullet}={}^{\bullet}d) \ulcorner\!\!\blacktriangledown\, r_1(\widetilde{\widetilde{\widetilde{\mathrm{exit}}}}) \cdot C^{up})+ \\
&\quad\ \sigma^{*}_{\mathsf{rel}}((d^{\bullet}=0) \ulcorner\!\!\blacktriangledown\, s_1(\widetilde{\widetilde{\widetilde{\mathrm{raise}}}}) \cdot C^{idle})) \\[1ex]
C^{dn} &= (0 \le d \le 5 \wedge \dot{d}=1) \curvearrowright\!\!\blacktriangledown_{\{d\}} \\
&\quad (\sigma^{*}_{\mathsf{rel}}((d^{\bullet}={}^{\bullet}d) \ulcorner\!\!\blacktriangledown\, r_1(\widetilde{\widetilde{\widetilde{\mathrm{appr}}}}) \cdot C^{\mathrm{dn}}) \\
&\quad\ \sigma^{*}_{\mathsf{rel}}((d^{\bullet}={}^{\bullet}d) \ulcorner\!\!\blacktriangledown\, r_1(\widetilde{\widetilde{\widetilde{\mathrm{exit}}}}) \cdot C^{dn})+ \\
&\quad\ \sigma^{*}_{\mathsf{rel}}((d^{\bullet}=0) \ulcorner\!\!\blacktriangledown\, s_1(\widetilde{\widetilde{\widetilde{\mathrm{lower}}}}) \cdot C^{idle}))
\end{aligned}
$$

The specification of ACP^{srt}_{hs} requires actions $\widetilde{\widetilde{\widetilde{pass}}}$, $\widetilde{\widetilde{\widetilde{ready}}}_{dn}$ and $\widetilde{\widetilde{\widetilde{ready}}}_{up}$ for mode switching. The specification is linearized manually using the axioms and lifting rules of ACP^{srt}_{hs}.

All variables x, r, d vary smoothly during delays. The specification of ACP^{srt}_{hs} requires actions $\widetilde{\widetilde{\widetilde{pass}}}$, $\widetilde{\widetilde{\widetilde{ready}}}_{dn}$ and $\widetilde{\widetilde{\widetilde{ready}}}_{up}$, because in ACP^{srt}_{hs}, an evolution operator offering one solution to variable trajectories during a delay, cannot be followed by another evolution operator offering a different solution. Consider for example G^{dn}. If the gate is going down with the rate -20, then when the gate is fully closed, in order to change the rate of going down to 0, an action $\widetilde{\widetilde{\widetilde{ready}}}_{dn}$ is required. Note in $(r^{\bullet} = {}^{\bullet}r) \ulcorner\!\!\blacktriangledown\, \widetilde{\widetilde{\widetilde{ready}}}_{dn}$, the derivative of r is not required to remain constant. Therefore, $\dot{r}$ can jump arbitrarily. The derivative $\dot{r}$ will jump such that the evolution proposition following action $ready_{dn}$ becomes true.

An action is required for mode switching in ACP^{srt}_{hs}.

Simplifying the specification

We apply axioms and lifting rules of $ACP^{srt}_{hs} + INT + REC$ in the simplification of $\partial_H(Trains \parallel Cntr \parallel Gate)$ including the equation (INT18) introduced in Section 2.2.4. It is given below:

$$\sigma^{r}_{\mathsf{rel}}(x) \mathbin{\|\!\lfloor} (\phi \curvearrowright\!\!\blacktriangledown_V (\textstyle\int_{v<r} \sigma^{v}_{\mathsf{rel}}(\nu_{rel}(y)) + \sigma^{r}_{\mathsf{rel}}(z))) = \phi \curvearrowright\!\!\blacktriangledown_V \sigma^{r}_{\mathsf{rel}}(x \mathbin{\|\!\lfloor} (\phi \curvearrowright\!\!\blacktriangledown_V z)) \qquad (INT18)$$

We use two phased derivation during the simplification of the specification. Two-phase derivation introduced in [BM05] means that the lifting rules and axioms of ACP^{srt}_{hs}, that are not interference-compatible bisimilar, can only be applied after eliminating parallelism, or in cases where parallel components do not share model variables (as is the case in the train gate controller system). An expansion of the train gate controller is also given in [BM05]. The expansion given below follows the pattern of the simplification in [BM05].

In [BM05], during simplification, parameterized recursion variables are used. The parameters passed are used to calculate the maximum and minimum bounds of time intervals between actions. Examples of parameters passed are exact delays of the controller in the C^{dn} or C^{up} mode. Parameterized processes are not defined currently in ACP^{srt}_{hs} but they can be defined. We have calculated parameter values that are passed between processes only after linearizing the train gate controller specification. The calculation of these parameters and their use in the specification below is informal. The initial linearized specification has less strict time bounds. After calculating the parameters, the time bounds of intervals between actions can be refined. Below, we give the linearized train gate controller specification with refined time bounds using parameters.

The train gate controller specification $\partial_H(Trains \parallel Cntr \parallel Gate)$ is simplified as follows:

Let X_0 denote the train gate controller system,

$$\partial_H(Trains \parallel Cntr \parallel Gate)$$

By axioms of $ACP^{srt}_{hs} + INT + REC$, X_0 can be rewritten as:

$$X_0 \quad = \quad (x \leq -1400 \wedge d = 0 \wedge r = 90) \mathbin{{}^\wedge\!\!\blacktriangle} X_1$$

where, X_1 is defined as:

$$\begin{array}{rcl} X_1 & = & (x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge \dot{d} = 0 \wedge r = 90 \wedge \dot{r} = 0) \curvearrowright_V \\ & & \partial_H(T^{far} \parallel C^{idle} \parallel G^{op}) \end{array}$$

Let ϕ_1 denote proposition $(x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge \dot{d} = 0 \wedge r = 90 \wedge \dot{r} = 0)$.

After simplification,

$$X_1 \quad = \quad \phi_1 \curvearrowright_V \partial_H((T^{far} \mid C^{idle}) \mathbin{\|\!\!\lfloor} G^{op})$$

For a set of variables V, C_V denotes the transition proposition $v^\bullet = {}^\bullet v \wedge \dot{v}^\bullet = {}^\bullet\dot{v}$, for every $v \in V$. In words C_V denotes that the values of all variables in V and the values of their derivatives remain unchanged during an action.

From the axioms and lifting rules of $ACP^{srt}_{hs} + INT + REC$, INT18 and two phased derivation, the following is derivable.

$$\begin{array}{rcl} X_1 & = & \phi_1 \curvearrowright_V (\int_{s \in [400/52, \infty)} \sigma^s_{\mathsf{rel}}(\partial_H((x = -1000 :\!\rightarrow C_{\{x\}} \mathbin{\ulcorner\!\blacktriangledown} s_1(\widetilde{\widetilde{appr}}) \cdot T^{near} \mid \\ & & \quad (d^\bullet = 0 \mathbin{\ulcorner\!\blacktriangledown} r_1(\widetilde{\widetilde{appr}}) \cdot C^{dn} + \\ & & \qquad\qquad d^\bullet = 0 \mathbin{\ulcorner\!\blacktriangledown} r_1(\widetilde{\widetilde{exit}}) \cdot C^{up})) \mathbin{\|\!\!\lfloor} G^{op}))) \\ & = & \phi_1 \curvearrowright_V (\int_{s \in [400/52, \infty)} \sigma^s_{\mathsf{rel}}(x = -1000 :\!\rightarrow \\ & & \qquad (C_{\{x,r\}} \wedge d^\bullet = 0) \mathbin{\ulcorner\!\blacktriangledown} c_1(\widetilde{\widetilde{appr}}) \cdot \partial_H(T^{near} \parallel C^{dn} \parallel G^{op}))) \end{array}$$

Let,

$$X_2 \quad = \quad \partial_H(T^{near} \parallel C^{dn} \parallel G^{op})$$

After simplification,

$$X_2 = \partial_H(T^{near} \mathbin{\|\!\!\lfloor} (C^{dn} \mid G^{op})) + \partial_H((C^{dn} \mid G^{op}) \mathbin{\|\!\!\lfloor} T^{near})$$

The recursion variable X_2 represents the scenario when the controller has just received an *approach* signal by an approaching train. The controller will now delay in the C^{dn} mode before sending the *lower* command to the gate. Using lifting rules, we derive that the action $\widetilde{\widetilde{pass}}$ of T^{near}, which is also not encapsulated by operator ∂_H, cannot be performed before action $c_2(\widetilde{\widetilde{\widetilde{lower}}})$. Let ϕ_2 denote proposition $(-1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge d \leq 5 \wedge \dot{d} = 1 \wedge r = 90 \wedge \dot{r} = 0)$. Then on simplifying X_2, we get:

$$\begin{array}{rcl} X_2 & = & \phi_2 \ulcorner\!\blacktriangledown_V \, (\int_{s \in [0,5]} \sigma^s_{\mathsf{rel}}((d^\bullet = 0 \wedge r^\bullet = {}^\bullet r \wedge C_{\{x\}}) \\ & & \ulcorner\!\blacktriangledown c_2(\widetilde{\widetilde{\widetilde{lower}}}) \cdot \partial_H(T^{near} \parallel C^{idle} \parallel G^{dn})^{s,r})) \end{array}$$

Let,

$$X_3^{t,R} \;=\; \partial_H(T^{near} \parallel C^{idle} \parallel G^{dn})^{t,R}$$

After simplification,

$$\begin{array}{rcl} \partial_H(T^{near} \parallel C^{idle} \parallel G^{dn}) & = & \partial_H(T^{near} \mathbin{\lfloor\!\lfloor} (C^{idle} \parallel G^{dn})) + \\ & & \partial_H(G^{dn} \mathbin{\lfloor\!\lfloor} (C^{idle} \parallel T^{near})) \end{array}$$

The recursion variable $X_3^{t,R}$ represents the scenario when the gate has received a *lower* command from the controller. The parameter t in $X_3^{t,R}$, gives the exact delay of controller in the C^{dn} mode. The parameter R in $X_3^{t,R}$ denotes the value of the angle of the gate at the instant when the controller issues a *lower* signal.

Again, using lifting rules, we derive that the action $\widetilde{\widetilde{\widetilde{pass}}}$ of T^{near} cannot be performed before action $\widetilde{\widetilde{\widetilde{ready_{dn}}}}$. Let ϕ_3 denote proposition $(-1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge \dot{d} = 0 \wedge r \geq 0 \wedge \dot{r} = -20)$. Then, after simplification:

$$\begin{array}{rcl} X_3^{t,R} & = & \partial_H(G^{dn} \mathbin{\lfloor\!\lfloor} (C^{idle} \parallel T^{near}) \\ & = & \phi_3 \ulcorner\!\blacktriangledown_V \, \sigma_{\mathsf{rel}}^{R/20}((r = 0) {:}\!\to \\ & & (r^\bullet = {}^\bullet r \wedge C_{\{d,x\}}) \ulcorner\!\blacktriangledown \widetilde{\widetilde{\widetilde{ready_{dn}}}} \cdot \\ & & \quad \partial_H(G^{cl} \parallel C^{idle} \parallel T^{near})^{t,R}) \end{array}$$

Let,

$$\begin{array}{rcl} X_4^{t,R} & = & \partial_H(G^{cl} \parallel C^{idle} \parallel T^{near})^{t,R} \\ & = & \partial_H(T^{near} \mathbin{\lfloor\!\lfloor} (G^{cl} \parallel C^{idle})) \end{array}$$

The variable $X_4^{t,R}$ denotes the scenario when the gate is fully closed and the train is now approaching the gate. The time passed since the train crossed the $-1000m$ detector, i.e. since performing action $c_1(\widetilde{\widetilde{\widetilde{appr}}})$ in recursion variable X_1, is the sum of the controller delay in C^{dn} mode, given by t in $X_4^{t,R}$, plus the time taken by the gate to close down, i.e. $R/20$. The train takes minimum time to reach the gate if it is coming at the maximum speed i.e. $52m/s$. The remaining time of the train to reach the gate is calculated by taking the total time required to travel $1000m$ **minus** the time passed since the train crossed the $1000m$ detector.

Let ϕ_4 denote proposition $(-1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge \dot{d} = 0 \wedge r = 0 \wedge \dot{r} = 0)$.

$$\begin{array}{rcl} X_4^{t,R} & = & \phi_4 \ulcorner\!\blacktriangledown_V \, (\int_{s \in [1000/52-(t+R/20),1000/40-(t+R/20)]} \sigma^s_{\mathsf{rel}}(\\ & & (x = 0) {:}\!\to (C_{\{x,r,d\}} \ulcorner\!\blacktriangledown \widetilde{\widetilde{\widetilde{pass}}} \cdot \partial_H(T^{past} \parallel C^{idle} \parallel G^{cl})))) \end{array}$$

Let,

$$X_5 \;=\; \partial_H(T^{past} \parallel C^{idle} \parallel G^{cl})$$

After simplification,

$$X_5 \quad = \quad \partial_H((T^{past} \mid C^{idle}) \mathbin{|\!\!\lfloor} G^{cl})$$

The recursion variable X_5 represents the scenario when the train has just crossed the gate and is approaching the second detector placed at 100 m from the gate. Let ϕ_5 denote $(0 \leq x \leq 100 \wedge 40 \leq \dot{x} \leq 52 \wedge \dot{d} = 0 \wedge r = 0 \wedge \dot{r} = 0)$. After simplification, X_5 can be rewritten as follows:

$$\begin{aligned} X_5 \quad = \quad & \phi_5 \curvearrowright_V (\textstyle\int_{s\in[100/52,100/40]}\sigma^s_{\mathsf{rel}}(\\ & \quad (x = 100) :\rightarrow (C_{\{r\}} \wedge x^\bullet \leq -1400 \wedge d^\bullet = 0) \Rsh c_1\widetilde{\widetilde{(exit)}}\cdot \\ & \quad \partial_H(T^{far} \parallel C^{up} \parallel G^{cl}))) \end{aligned}$$

Let,

$$X_6 \quad = \quad \partial_H(T^{far} \parallel C^{up} \parallel G^{cl})$$

After simplification,

$$X_6 \quad = \quad \partial_H((C^{up} \mid G^{cl}) \mathbin{|\!\!\lfloor} T^{far})$$

X_6 represents the scenario when the controller has received the *exit* signal from the train. The controller will now delay before sending the *raise* signal to the gate. Meanwhile, another train can appear at $-1400m$ from the gate. The minimum time taken by this train to reach the first detector is $400/52 = 7.69s$. Using lifting rules, we derive that the controller issues a *raise* signal to the gate *before* a new train reaches the first detector.

Let ϕ_6 denote $(x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge \dot{d} = 1 \wedge d \leq 5 \wedge r = 0 \wedge \dot{r} = 0)$. After simplification, X_6 is rewritten as:

$$\begin{aligned} X_6 \quad & = \quad \partial_H((C^{up} \mid G^{cl}) \mathbin{|\!\!\lfloor} T^{far}) \\ & = \quad \phi_6 \curvearrowright_V (\textstyle\int_{s\in[0,5]}\sigma^s_{\mathsf{rel}}((C_{\{x\}} \wedge d^\bullet = 0 \wedge r^\bullet = {}^\bullet r) \Rsh c_2\widetilde{\widetilde{(raise)}}\cdot \\ & \qquad \partial_H(T^{far} \parallel C^{idle} \parallel G^{up})^s)) \end{aligned}$$

Let,

$$X_7^t = \partial_H(T^{far} \parallel C^{idle} \parallel G^{up})^t$$

Simplifying $\partial_H(T^{far} \parallel C^{idle} \parallel G^{up})$, we get:

$$\begin{aligned} \partial_H(T^{far} \parallel C^{idle} \parallel G^{up}) \quad = \quad & \partial_H((T^{far} \mid C^{idle}) \mathbin{|\!\!\lfloor} G^{up}) + \\ & \partial_h(G^{up} \mathbin{|\!\!\lfloor} (T^{far} \parallel C^{idle})) \end{aligned}$$

The situation at X_7^t is that the controller has just sent the *raise* signal to the gate. The parameter t in X_7^t, is the exact delay of the controller in C^{up} mode. Also, t is the time elapsed since the first train crossed the second detector. The gate will take 90/20 time units to fully open and perform action $\widetilde{\widetilde{ready_{up}}}$. The recursion variable X_7^t models these two scenarios.

- **Scenario 1**: After receiving the *raise* signal, the gate takes 90/20 seconds to completely open and perform action $\widetilde{\widetilde{ready_{up}}}$.
- **Scenario 2**: Before the gate completely opens, a new train arrives at the first detector and sends an *approach* signal to the controller.

In scenario 2, the minimum time lapse between actions $c_2\widetilde{\widetilde{(raise)}}$ and $c_1\widetilde{\widetilde{(appr)}}$ is 400/52 *minus* the delay of controller in sending *raise* signal. This time lapse cannot be greater than 90/20, as by then the gate will be fully opened.

Let ϕ_7 denote $(x \leq -1000 \wedge 48 \leq \dot{x} \leq 52 \wedge \dot{d} = 0 \wedge d = 0 \wedge 0 \leq r \leq 90 \wedge \dot{r} = 20)$. Then, after simplification, X_7^t can be rewritten as:

$$\begin{array}{rcl} X_7^t & = & \phi_7 \ulcorner\!\!\blacktriangledown_V (\sigma_{\textsf{rel}}^{90/20}((r = 90) :\rightarrow (C_{\{x,d\}} \wedge r^\bullet = {}^\bullet r) \ulcorner\!\!\blacktriangledown \widetilde{\widetilde{ready}}_{up} \\ & & \quad \cdot\partial_H(T^{far} \parallel C^{idle} \parallel G^{op})^{t+90/20}) \\ & & +\int_{s\in[(400/52)-t,90/20)}\sigma_{\textsf{rel}}^s((x = -1000){:}\!\rightarrow \\ & & \quad (C_{\{x,r\}} \wedge d^\bullet = 0) \ulcorner\!\!\blacktriangledown c_1\widetilde{\widetilde{(appr)}} \cdot \partial_H(T^{near} \parallel C^{dn} \parallel G^{up})^s)) \end{array}$$

The behaviour of the process term $\partial_H(T^{far} \parallel C^{idle} \parallel G^{op}))$ is described by the recursion variable X_1. But in variable X_7^t, in the recursive call to $\partial_H(T^{far} \parallel C^{idle} \parallel G^{op})$, the minimum time before the new train can reach the first detector has to be readjusted.

Let

$$X_{11}^{t'} = \partial_H(T^{far} \parallel C^{idle} \parallel G^{op})^{t'}$$

The parameter t' in $X_{11}^{t'}$ denotes the time passed since the first train crossed the second detector. After the first train crosses the second detector, a new train can come at a distance of $1500m$ from the first train, or at a distance of $-1400m$ from the gate. The time bound in recursion variable X_1 is readjusted to cater for the time already passed since a new train may have appeared at $-1400m$ from the gate.

$$\begin{array}{rcl} X_{11}^t & = & \partial_H(T^{far} \parallel C^{idle} \parallel G^{op})^t \\ & = & \phi_1 \ulcorner\!\!\blacktriangledown_V (\int_{s\in[400/52-t,\infty)}\sigma_{\textsf{rel}}^s(x = -1000{:}\!\rightarrow \\ & & \qquad (C_{\{x,r\}} \wedge d^\bullet = 0) \ulcorner\!\!\blacktriangledown c_1\widetilde{\widetilde{(appr)}} \cdot \partial_H(T^{near} \parallel C^{dn} \parallel G^{op}))) \end{array}$$

Let,

$$X_8^t = \partial_H(T^{near} \parallel C^{dn} \parallel G^{up})^t$$

After simplification,

$$\begin{array}{rcl} \partial_H(T^{near} \parallel C^{dn} \parallel G^{up}) & = & \partial_H(G^{up} \mathbin{\lfloor\!\lfloor} (T^{near} \parallel C^{dn}))+ \\ & & \partial_H((C^{dn} \mid G^{up}) \mathbin{\lfloor\!\lfloor} T^{near}) \end{array}$$

The recursion variable X_8^t denotes the case when a new train has arrived before the gate was fully open. The parameter t in X_8^t is the time elapsed since the controller sent the *raise* signal to the gate. Again two situations can arise:

- **Scenario 1**: The gate performs the action $\widetilde{\widetilde{ready}}_{up}$ after $90/20 - t$ seconds.
- **Scenario 2**: Before the gate completely opens, the controller sends the *lower* signal to the gate.

Let ϕ_8 denote proposition $(-1000 \leq x \leq 0 \wedge 40 \leq \dot{x} \leq 52 \wedge d \leq 5 \wedge \dot{d} = 1 \wedge 0 \leq r \leq 90 \wedge \dot{r} = 20)$. Then, after simplification, X_8^t can be rewritten as:

$$\begin{array}{rcl} X_8^t & = & \phi_8 \ulcorner\!\!\blacktriangledown_V \int_{s\in[0,90/20-t)}\sigma_{\textsf{rel}}^s((C_{\{x\}} \wedge d^\bullet = 0 \wedge r^\bullet = {}^\bullet r) \\ & & \quad \ulcorner\!\!\blacktriangledown c_2\widetilde{\widetilde{(lower)}} \cdot \partial_H(T^{near} \parallel C^{idle} \parallel G^{dn})^{t+s,r}) \\ & & +\sigma_{\textsf{rel}}^{90/20-t}((r = 90) :\rightarrow (C_{\{x,d\}} \wedge r^\bullet = {}^\bullet r) \\ & & \ulcorner\!\!\blacktriangledown \widetilde{\widetilde{ready}}_{up} \quad \cdot \partial_H(T^{near} \parallel C^{dn} \parallel G^{op})) \end{array}$$

The recursion variables $X_3^{t,R}$ and X_2 define the behaviour of process terms $\partial_H(T^{near} \parallel C^{idle} \parallel G^{dn})$ and $\partial_H(T^{near} \parallel C^{dn} \parallel G^{op})$ respectively. The parameter t in $X_3^{t,R}$, denotes the delay of the controller in C^{dn} mode and R denotes the angle of the gate at the instant it received a *lower* signal. Note that in the recursive call to X_2 in X_8^t, the value of the variable d is different from its initial value 0. Hence, we do not need to pass a parameter to X_2, because as soon as d becomes 5, the recursion variable X_2 cannot delay further.

Verifying the safety condition

As can be seen from recursion variables $X_3^{t,R}$ and $X_4^{t,R}$, the train crosses the gate after the gate is fully closed. In order to calculate the position of the train when the gate closes, we take the worst possible case. We consider the train coming at the fastest speed, the controller delaying longest and the gate fully open. The maximum time lapse between actions $c_1(\widetilde{\widetilde{\mathrm{appr}}})$ in recursion variable X_1 and $\widetilde{\widetilde{\mathrm{ready}}}_{\mathrm{dn}}$ in $X_3^{t,R}$ is 9.5 s. At the maximum speed of $52m/s$, the train covers 494 m in 9.5 s. Hence, the gate is closed when the train is at a distance of at least 506m from the gate.

While expanding the specification, as in [BM05], we have calculated the timing of events and based on these calculations have checked the safety condition of the train gate controller. In this case study, like HyPA, using the lifting rules and axioms, predicates on the position of the gate and distance of the train from the gate can be incorporated in the specification while flattening it. Also, then it can be verified that the gate is closed whenever the train is within $506m$ of the gate.

Work needs to be done towards tool support in automatically applying algebraic reasoning to ACP_{hs}^{srt} specifications. Also further research on the elimination results of $ACP_{hs}^{srt} + INT$ is required.

2.6.5 Concluding Remarks

The train gate controller specification is simplified using axioms of HyPA and ACP_{hs}^{srt}. For simplification in ACP_{hs}^{srt}, we use Equation INT18 given in Section 2.2.4, in addition to the axioms of ACP_{hs}^{srt}. For ϕ-calculus, we do reachability analysis on the state space of train gate controller. In Hybrid χ, we simplify the specification using an elimination theorem given in [SM06] and further use reachability analysis on the simplified form obtained. All process algebras confirm the safety condition that the gate is closed whenever the train is at a distance of 506 m or less from the gate.

2.7 Available Tools

Tools play an important role in enhancing the usability of a formalism. ACP_{hs}^{srt} has no tools for simulation or verification of its specifications. A brief introduction of tools available for other process algebras is given below. We try to use the tools for modelling the train gate controller system given in Section 2.6. We could not use some tools successfully, i.e. the model checker for ϕ and linearization tool of HyPA, as explained further on.

2.7.1 HyPA

The tools available for HyPA specifications include a linearization tool and a simulation tool.

1. **HyPA Linearization Tool**

   Linearization is a procedure of rewriting a process specification into a simpler form which consists of only basic operators of the process algebra. In particular, a linearized specification does not contain a parallel operator. Many analysis tools and techniques can only be applied on linearized system specifications (see [Wou01, CR05, BM05]). More on linearization can be found in Chapter 4.

   A parallel operator can be eliminated manually from a HyPA specification by repeated application of axioms. However, as obvious from Section 2.6.1, manual linearization of even a small algebraic specification can be a long, tedious and error-prone task. Therefore tools for automatic linearization are very useful.

   In [BRC06], two linearization algorithms for HyPA specifications are given. One linearization algorithm makes use of the linearization tool of process algebra μCRL. HyPA specifications are first translated to pCRL, linearized by **mcrl** and the linear form obtained is translated back to HyPA. The drawbacks of this algorithm are that only a subset of HyPA processes can be linearized and the resulting linear form is very large.

   The second linearization algorithm uses the abstraction operator of HyPA to reduce the increase in the size of the linear form obtained. It can also handle a comparatively larger class of HyPA processes.

   We examined the text based tool implementing the second algorithm. A number of HyPA specifications that have been linearized using the tool are included with the distribution of the tool. We tried to linearize our train gate controller specification (see Section 2.6) using this tool. We were unable to linearize our specification due to a run time error in the tool. The work on the tool has been discontinued and we could not get user support to solve our problem.

2. **HyPA Simulator**

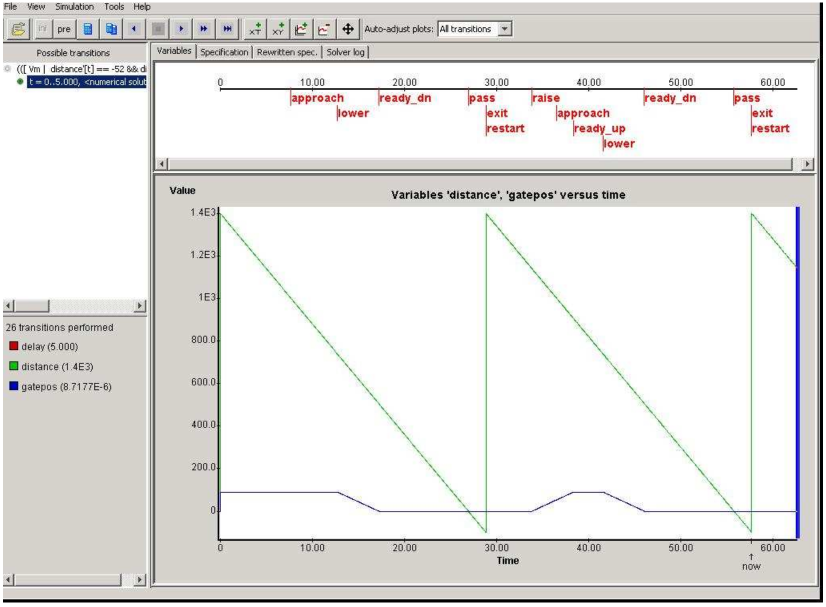
   A simulation tool for HyPA specifications is available, see [Sch05].

   The simulator uses the solving capabilities of software Mathematica [Mat]. The tool does not allow non-determinism in the evolution of variables. The flow of variables during a delay can be specified by ordinary differential equations and algebraic inequalities. Differential inclusions are not allowed. If the variables have not been initialized or the system of equations in flow clauses or reinitialization clauses is under specified, the simulator prompts the user to add more restrictions. There are two modes available for simulation: one *random* or *automatic*, and the other *step by step*. In random simulation choices are made by the simulator. A stop criterion for a random simulation can be given. In step-by-step mode, the simulator prompts the user to resolve an alternative choice or to choose the duration of a delay.

   We used the simulator to simulate the train gate controller specification. A snapshot of the simulation using this tool is given in Figure 2.4.

   In the center of the figure, a graph of the train distance (given in green) from the gate and the angle of the gate (given in blue) versus time is shown. At time 0, the train is at a distance $-1400m$ from the gate and the gate is open, i.e. at angle 90°. The distance of the train varies at a constant rate, i.e. $52m/s$ and is reset to -1400, when the train crosses $100m$ from the gate. The tool does not allow non-determinism in variable

Figure 2.4: Train Gate Controller in HyPA



Auto-adjust plots: All transitions

Possible transitions

(([Vm | distance'[t] == -52 && di

t = 0..5.000, <numerical solut

26 transitions performed

🟥 delay (5.000)

🟩 distance (1.4E3)

🟦 gatepos (8.7177E-6)

evolutions, therefore a constant velocity of $52m/s$ has been chosen for the train. The top center window shows the timing of discrete events. The top left window displays the next possible transitions. The user can select the next transitions from one of them. The bottom left window displays the current values of the variables delay, distance and gatepos (gate angle).

We found the simulator tool to be user friendly. The tool gives a number of options to the user (for example zooming in and out of the graph and undoing an action or time step) and provides a good visualization of options available, actions performed and variables' evolution. The simulator can help detecting errors in specifications. An error in the steam boiler specification given in [CR05] was detected by the simulator. By running its simulation, it was observed that water level can become negative initially (see section 7.1 in [Sch05],).

The disadvantage of the tool is that only a restricted flow (differential equations) of variables is allowed by it.

2.7.2 ϕ-Calculus

SPHIN is a model checker for hybrid system specifications in $\phi-$calculus. SPHIN is an extension of the model checker SPIN. Model checking is a verification technique by which a property of a specification is verified by taking into account all possible scenarios during the execution of the specification. On the other hand, techniques such as simulation and testing cover only a subset of the behaviours of a specification. By repeatedly simulating or testing a specification, one can increase one's confidence about its correctness but not verify whether a certain condition holds for a specification or not.

SPHIN is an extension of SPIN and the input language to SPHIN is PROMELA-Hybrid. PROMELA-Hybrid is a superset of PROMELA (the input language for SPIN). The PROMELA-

Hybrid language provides similar features as ϕ-calculus. Currently, there is no formal translation available between the constructs of the two languages. In a PROMELA-Hybrid specification, three intervals are associated with an analog variable. These represent a range of possible initial values, a range of possible derivative values and invariants on an analog variable. ϕ-Calculus specifications differ from PROMELA-Hybrid specifications in this respect that in PROMELA-Hybrid an environment variable can be initialized to any value in a given interval instead of being assigned a single value as in ϕ-calculus. By allowing a range of possible flows, we see that the SPHIN model checker allows more non-determinism than allowed by the simulators of HyPA and Hybrid χ (as is mentioned in the next section).

The distribution of SPHIN comes along with a number of hybrid system examples modelled in PROMELA-Hybrid. These include a train gate controller, Fischer's mutual exclusion algorithm, 3-robot bucket brigade and flocking agents.

The train gate controller specification distributed with the SPHIN package has been taken from [Hen96] and is different from our train gate controller case study given in Section 2.6. Using SPHIN, we tried to verify the safety condition (in the train gate controller specification of Section 2.6), that the gate is always closed whenever the train is less than 350 meters away from it. We were unable to verify that this condition holds in our train gate controller specification. This is contrary to the results obtained by linearizing the train gate controller specification in HyPA and ACP^{srt}_{hs}. In HyPA and ACP^{srt}_{hs}, by manually expanding the specification using axioms, we observe that the gate is always closed when the train is within 506 meters from it. Hence it appears that we erred while translating the train gate controller specification from ϕ-calculus to PROMELA-Hybrid or there is a bug in SPHIN. The PROMELA-hybrid code for our train gate controller specification is given in Appendix A.

SPHIN is a text-based tool. It gives messages about the evolution of variables, but in most cases, it is not easy to interpret the messages. A number of features that are available for the model checker SPIN are as yet not available for PROMELA-Hybrid specifications. These include guided simulation with counter-example generation and GUI support. A number of improvements have been suggested in [RSC06]. These suggestions include establishing a formal connection between $\phi-$calculus and PROMELA-Hybrid, adding convenience features such as guided simulation with counter-example generation, random/interactive simulation with analog state visualization and a GUI interface for SPHIN.

2.7.3 Hybrid χ

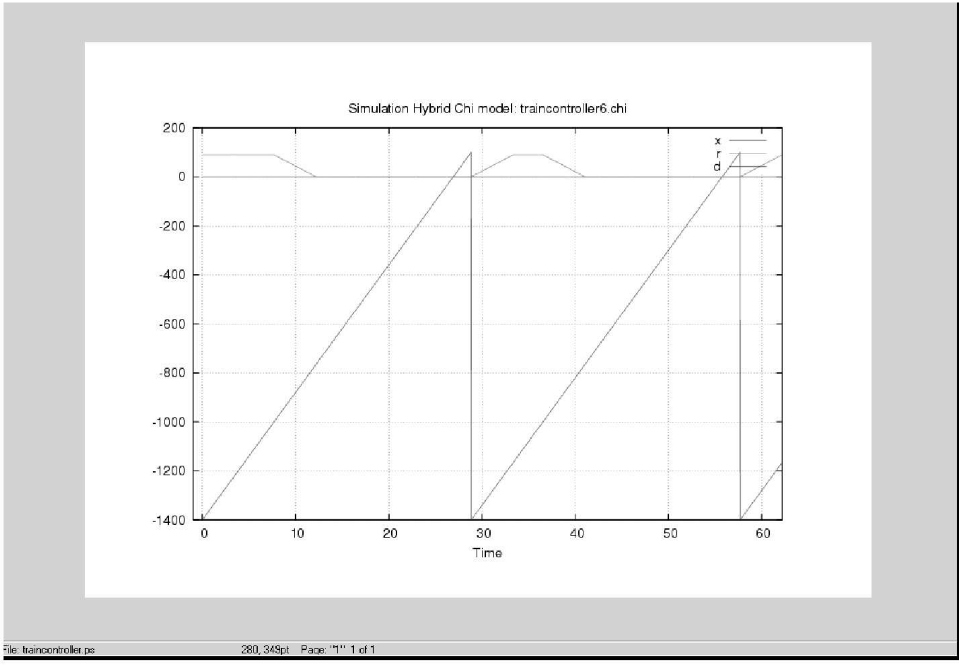
A list of tools for χ and Hybrid χ is available on the website,
http://se.wtb.tue.nl/sewiki/chi/tooling.

As χ is an active formalism, therefore this list keeps on evolving. At the time of writing of this thesis, the tool set available for Hybrid χ specifications consists of a translator of Hybrid χ specifications into hybrid automata, two Hybrid χ simulators and a linearization tool (not mentioned on the website) for hybrid specifications.

1. **Chi2HA translator**

   A subset of Hybrid χ language has been formally translated into a class of hybrid automata closely resembling I/O hybrid automata. The tool "Chi2HA" automatically translates Hybrid χ specifications into hybrid automata. A tool `PHaVER`, (Polyhedral hybrid automaton verifier, see [Fre05]) is then used to analyze the translations. For a

Figure 2.5: Train Gate Controller in Hybrid χ



discussion of the translation of specifications from Hybrid χ to hybrid automata and a description of the tool, please see [BJM+03] and [BRS+07].

2. **Hybrid χ Simulator**

   There are two simulators available for Hybrid χ specifications.

   - One is hybrid Chi Python simulator that uses the numeric solving capabilities of DDASRT and/or the symbolic solving capabilities of Maple.
   - The other is Chi-Simulink simulator that uses the numerical solving capabilities of Matlab Simulink.

   The abbreviation 'ddasrt' stands for "double precision dassle root finding."

   The simulators can simulate what can be solved by their respective solvers.

   The simulators simulate a Hybrid χ language that is strongly typed whereas types have not been mentioned in [BMR+06].

   We include the simulation result for Hybrid χ specification for train gate controller from hybrid chi python simulator. In Figure 2.5, the variable x represents the distance of the train from the gate (initialized to -1400), the variable r represents the angle of the gate (initialized to 90°) and the variable d represents a possible delay by the controller in forwarding a message to gate.

   The hybrid chi python simulator has a textual interactive user interface. It plots a graph of variable evolution during delays by using gnuplot. The simulator tool provides the user with a wide variety of options that include choosing the right solver, formatting the graph of variable evolution and choosing between interactive or automatic mode etc. In interactive mode, the simulator prompts the user for resolving alternative choices and for specifying the duration of delays. In automatic mode, the simulator makes these choices itself.

The Chi-Simulink simulator has a graphical interface and is based on the Simulink environment. See the Hybrid χ website [Chi] for more about the simulators.

3. **Hybrid χ linearization tool**

   In [The06], an algorithm and development of a tool is described for linearization of Hybrid χ specifications. The linearization algorithm given in [The06] does not use any special techniques to reduce the size of the output linearized form such as used by linearization algorithm of μCRL [Use02] and that of HyPA [BRC06].

   We used the linearization tool of [The06] to linearize the train gate controller system. The linear term obtained for the case study was very long with over a 170 recursion variables defined in the recursion definition. Many of these recursion variables were not reachable from the initial term.

   Chapter 4 of this thesis, describes another linearization algorithm for Hybrid χ specifications. This algorithm uses discrete counters in the linearization procedure. Using discrete counters, the increase in size of a linearized process term due to elimination of a parallel operator is reduced. This linearization algorithm will be implemented in $ASF + SDF$ [BKV01]. Currently the implementation of this algorithm is delayed in favor of the ongoing work on Hybrid χ 2.0 (see [BHR$^+$08]).

2.7.4 Concluding Remarks

As we analyze the tools available for each process algebra, we find that Hybrid χ is most active regarding tooling and user support. The results from linearization tools are not suitable for manual inspection (the HyPA linearization tool allows suppressing information not needed by the user). Due to lack of adequate user support, the linearization tool of HyPA and the SPHIN model checker could not be used to analyze the train gate controller specification.

2.8 Conclusion

In this chapter we have done a comparative study of four rather recent process algebras for hybrid systems. Our study includes HyPA [CR05], ϕ-calculus [RS03], Hybrid χ [BMR$^+$06] and ACP^{srt}_{hs} [BM05]. These process algebras are interesting in the sense that all of them originate from different backgrounds and different motivations guide their development. HyPA originates from ACP and aims to be a complete but simple process algebra for hybrid systems. Hybrid χ originates from the modelling language χ. The goal of χ and Hybrid χ is to be a modelling langauge of a wide class of manufacturing systems with a rigorous formal basis. ϕ-calculus is π-calculus extended with features to model dynamic behaviour of hybrid systems. ACP^{srt}_{hs} extends existing theories timed process algebra ACP^{srt} [BM02a] and Process Algebra with Propositional Signals ACP_{ps} [BB97] to model hybrid systems. While HyPA and ACP^{srt}_{hs} focus on axiomatizations and symbolic reasoning, ϕ-calculus and Hybrid χ focus on modelling and verification.

In our analysis, we observe that all process algebras allow compositional modelling. Complex systems can be constructed by composing simple processes by alternative, sequential and parallel compositions. From process algebraic point of view ϕ-calculus distinguishes itself from other process algebras in allowing modelling of reconfigurable systems. Modelling such systems in other process algebras is not that as easy as in ϕ-calculus. But ϕ-calculus also has its shortcomings. In ϕ-calculus, the interaction of dynamic behaviour of parallel processes cannot be modelled as in other process algebras. In a delay of a parallel composition between two processes P and Q, the process $P \parallel Q$ will delay according to the dynamics of P or Q, but not both. This can easily be remedied as explained in Section 2.5. In the train gate controller specification, the parallel components do not share environment variables. Otherwise, it would not have been possible to model the interaction between parallel components of the system. Two other restrictions in ϕ-calculus are: updating of environment variables by assignments (and not by predicates); and the restriction of flows to delay behaviour only expressible by differential equations with unique solutions. A model checker SPHIN model checks hybrid systems modelled in a language PROMELA-HYBRID–a language closely related to ϕ-calculus. SPHIN is proposed as a model checker for hybrid systems modelled in ϕ-calculus but a formal translation from ϕ-calculus to SPHIN has not yet been developed.

Hybrid χ is prominent among other process algebras in its tools, tool support and number of users. Two simulators and a linearization tool for hybrid χ specifications are available. A tool for translation of models specified in a subset of Hybrid χ language, to models in Hybrid I/O automata is available. The tool PHAVER that verifies safety properties for hybrid systems modelled in Hybrid I/O automata can then be used to verify the translated Hybrid χ models. Translations from Hybrid χ to a number of control theory formalisms for describing hybrid systems, see [BRS$^+$07], have also been defined. Hybrid χ lags behind in equational reasoning for its specifications from HyPA and ACP^{srt}_{hs}. (ϕ-calculus does not give any axioms for its specification.) The train gate controller specification in Hybrid χ is simplified using an elimination theorem, that allows eliminating the parallel operator from a subset of Hybrid χ processes. The linear form obtained by eliminating parallel operator contains many unreachable options which cannot be eliminated algebraically.

HyPA is a conservative extension of both ACP and control theory. HyPA distinguishes it self from other process algebra in its equational reasoning. The train gate controller specification is linearized using equational reasoning alone. Tools for simulation and linearization for HyPA models are available. HyPA semantics models the behaviour of delaying processes non time deterministically. Time determinism is well-known a property of timed systems. Time determinism states that a delay cannot resolve choices between delaying processes. HyPA cannot express time determinism. Hence, it is not suitable for modelling of systems where continuous physical behaviour has been calculated and actions are the only source of non-determinism.

The fourth process algebra included in our study is ACP^{srt}_{hs}. ACP^{srt}_{hs} is an extension of timed process algebra ACP^{srt} [BM02a] and Process Algebra with Propositional signals ACP_{ps} [BB97]. We observe that in ACP^{srt}_{hs} using two separate operators, the relative delay operator (σ_{rel}) and the signal evolution operator ($\curvearrowright$), for describing the delay of a process adds complexity to the semantics. Because while deriving a transition for a process, care must be taken that both the evolution proposition and the delay operator allow the desired delay under the given environment conditions (variables valuation). Following the convention of time determinism in timed process algebras, ACP^{srt}_{hs} chooses to only model time determinism or weak time determinism in delaying processes. A consequence of only allowing time determinism is that the definition of a variable abstraction operator in operational semantics becomes difficult and has not been achieved yet. [2] ACP_{hs}^{srt} has no tools to help in analysis of its models.

Recently, it was found that some of the axioms of ACP_{hs}^{srt} including associativity of choice (A2) and time determinism (SRT3) are not sound. These errors stem from putting (in our opinion) un-intuitive emphasis on duration of the delay of summands in a choice. Correcting these errors is discussed in chapter 3. We discover that not all closed process terms of $ACP_{hs}^{srt} + INT$ can be linearized to a basic term. Because counterparts of some axioms concerning the left parallel merge for integration are missing in $ACP_{hs}^{srt} + INT$.

Regarding future work, the further development of numerical simulators should be given priority, if hybrid process algebras are to be used as modelling languages in a more industrial setting. But, with more emphasis on formal semantics than is usual in the control science community.

[2] One idea is to give a time stamped semantics to ACP_{hs}^{srt} (see [BR04]) and then introduce a variable abstraction operator in it.

Chapter 3

Basic Timed Process Algebra with Non-existence

Chapter 2 presents a comparison of four process algebras for specification of hybrid systems. Process algebras studied include Hybrid Process Algebra (HyPA) [CR05] , ϕ-Calculus [RS03], Hybrid χ [BMR^{+}06] and Process Algebra for Hybrid Systems (ACP^{srt}_{hs}) [BM05]. The work presented here is related to ACP^{srt}_{hs} [BM05].

Recently, a number of errors have been found in Process Algebra for Hybrid Systems. It turns out that in ACP^{srt}_{hs}, Choice is not associative (Axiom A2), Time determinism (Axiom SRT3) does not hold and a number of other less important axioms are also not sound. One of the most basic components of *Process Algebra for Hybrid Systems* is called *Basic Process Algebra with standard relative timing (srt) and Non existence* ($\perp$) abbreviated as $BPA^{srt}_{\perp}$. One of the errors in ACP^{srt}_{hs}, namely unsoundness of Time determinism axiom, can be traced down to this most basic component $BPA^{srt}_{\perp}$, therefore we think that fixing $BPA^{srt}_{\perp}$ is essential in rectifying the errors in Process Algebra for Hybrid Systems. Accordingly, in this chapter we present two proposals for correcting the process algebra $BPA^{\mathrm{srt}}_{\perp}$.

In this chapter, two proposals for $BPA^{srt}_{\perp}$ are presented aimed at correcting the error in time determinism axiom. The two proposals given in this chapter differ from each other in the sense that in the first proposal, time determinism holds for the process terms of the sub-algebra BPA^{srt} but not for the non-existence process. In the second proposal, time determinism holds for all process terms of $BPA^{srt}_{\perp}$ including the Non-existence process. Both proposals are conservative ground-extensions of BPA^{srt} [BM02a] and $BPA_{\perp}$ [BB97].

The chapter is structured as follows: In Section 3.1, we give a hierarchical structure of the Process Algebra for Hybrid Systems. In Section 3.2, we point out the appearance of errors in the hierarchy of ACP^{srt}_{hs} and give examples violating the non-associativity of the choice and the time determinism axiom (SRT3). In the preliminaries section, we introduce the reader to Basic Timed Process Algebra with Non-existence ($BPA^{srt}_{\perp}$). In Section 3.4, we discuss what is wrong with the current presentation of the algebra $BPA^{srt}_{\perp}$ as it is put forward in [BM05]. We present our first proposal for a corrected $BPA^{srt}_{\perp}$ in Section 3.5. In this proposal, we leave the semantics intact and change the SRT3 axiom to fit the semantics. Before presenting our second proposal in Section 3.6, we discuss a number of possible attempts at changing the semantics and leaving the time determinism axiom intact for all process terms in the theory. In Section 3.6.1, we review an instance from literature, where a timed process algebra has been combined with the Non-existence process $\perp$. In our second proposal of $BPA^{srt}_{\perp}$, we modify the semantics so that the axiom SRT3 holds for all process terms in the algebra. Section 3.7 discusses the possibilities of extending our proposals for $BPA_{\perp}^{srt}$ to a hybrid process algebra.

3.1 Hierarchy of ACP_{hs}^{srt}

The equational theory of Process Algebra for Hybrid Systems [BM05] is a blend of various process algebraic theories. Figure 3.1 shows a hierarchical structure of the theories ACP_{hs}^{srt} and BPA_{hs}^{srt}. BPA_{hs}^{srt} is Basic Process Algebra for Hybrid Systems—i.e. ACP_{hs}^{srt} without concurrency and communication. The discussion in this chapter does not involve concurrency and is therefore confined to Basic Process Algebra for Hybrid Systems.

BPA_{hs}^{srt} is constructed from some basic theories as follows:

Basic Process Algebra with standard relative timing (BPA^{srt}) [BM02a] is extended with the Non-existence process ($\perp$) from [BB97]. The resulting theory $BPA_{\perp}^{srt}$ is combined with Basic Process Algebra with propositional signals BPA_{ps} [BB97]. The succeeding algebra BPA_{ps}^{srt} is extended with two new operators needed to specify hybrid behaviour of processes. Then we have built the Basic Process Algebra for Hybrid Systems BPA_{hs}^{srt}. Integration and recursion further increase the expressiveness of BPA_{hs}^{srt}.

3.2 Location of Errors

There are two main errors in BPA_{hs}^{srt}. One is that the choice is not associative. The other is that the axiom of time determinism (SRT3) does not hold. If we look at the hierarchical structure of BPA_{hs}^{srt} (see Figure 3.1), then the unsoundness of axiom SRT3 already appears in $BPA_{\perp}^{srt}$, i.e. there are two $BPA_{\perp}^{srt}$ process terms that are equivalent according to axiom SRT3 but are not bisimilar. On the other hand, the associativity of the choice only breaks down at the level of BPA_{ps}^{srt}. In the theories below BPA_{ps}^{srt}, we cannot find an example violating the associativity of choice. The unsoundness of time determinism axiom and non-associativity of choice propagate in theories above $BPA_{\perp}^{srt}$ and BPA_{ps}^{srt}, where we find more erroneous axioms.

The following examples exhibit violation of axiom SRT3 and non-associativity of choice. An introduction to the semantics of BPA_{hs}^{srt} is given in Appendix B. In the examples below we refer to the transition rules and the definitions of bisimulation given in this appendix. In BPA_{hs}^{srt}, when two processes x and y are ic-bisimilar, then we denote it by $x \leftrightarroweq y$. The definition of *IC-bisimulation* is given in the Appendix B.

Axiom SRT3: $\sigma_{\mathsf{rel}}^{r}(x) + \sigma_{\mathsf{rel}}^{r}(y) = \sigma_{\mathsf{rel}}^{r}(x+y) \qquad r \geq 0$

Example 4 *(Counter-Example to SRT3)*

$$\sigma_{\mathsf{rel}}^{t}(\tilde{\tilde{a}}) + \sigma_{\mathsf{rel}}^{t}(\perp) \not\leftrightarroweq \sigma_{\mathsf{rel}}^{t}(\tilde{\tilde{a}} + \perp) \qquad t > 0$$

According to the semantics of BPA_{hs}^{srt} (Rule HS-12) the left-hand side of this inequation can perform a time-step,

$$\langle \sigma_{\mathsf{rel}}^{t}(\tilde{\tilde{a}}) + \sigma_{\mathsf{rel}}^{t}(\perp), \alpha \rangle \overset{t,\rho}{\longmapsto} \langle \tilde{\tilde{a}}, \alpha' \rangle,$$

because one of the arguments ($\sigma_{\mathsf{rel}}^{t}(\tilde{\tilde{a}})$) can perform that time-step (Rule HS-6), while the other $\sigma_{\mathsf{rel}}^{t}(\perp)$ is consistent (Rule HS-37) but cannot perform this time-step (in particular, Rule HS-6 is not applicable because $\alpha' \notin [\mathbf{s}(\perp)]$. On the other hand, the right-hand side of this inequation

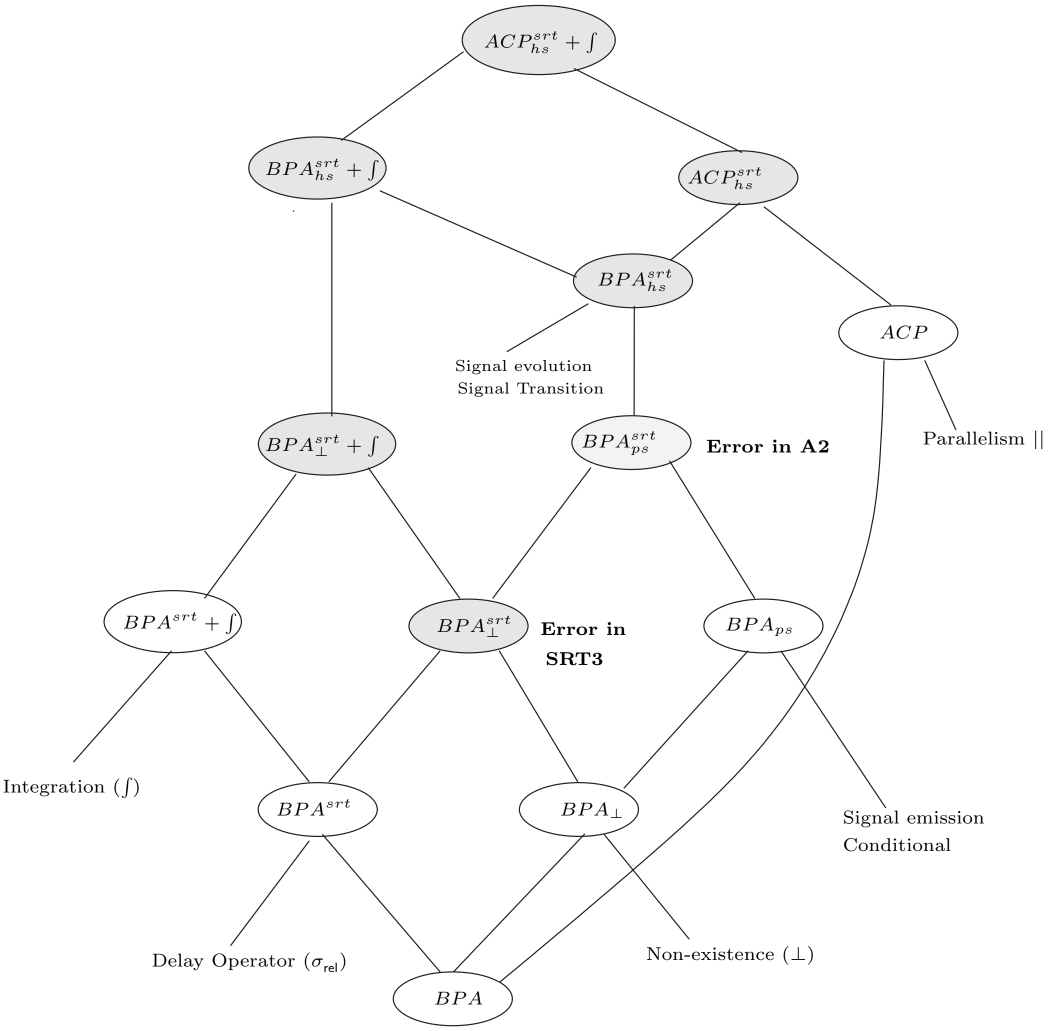


Figure 3.1: Hierarchical Structure of ACP_{hs}^{srt}

cannot perform a time-step of duration t, in particular because $\alpha' \notin [\mathbf{s}(\tilde{\tilde{a}} + \bot)]$, and hence Rule HS-6 is not applicable. We conclude that the two processes are not ic-bisimilar (nor bisimilar).

Example 5 *(Non-Associativity of Choice)*

Let p, q, r be process terms, where,

$$\begin{array}{rcl} p & = & \sigma^t_{\mathsf{rel}}((l = 0) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{a}}) \\ q & = & \sigma^t_{\mathsf{rel}}((l = 1) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{b}}) \\ r & = & \sigma^t_{\mathsf{rel}}(\tilde{\tilde{c}}) \end{array}$$

$$(p + q) + r \not\underline{\leftrightarrow} p + (q + r)$$

Note, that for each of the three subprocesses, $\sigma^t_{\mathsf{rel}}((l = 0) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{a}})$, $\sigma^t_{\mathsf{rel}}((l = 1) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{b}})$ and $\sigma^t_{\mathsf{rel}}(\tilde{\tilde{c}})$, some time-step of duration t is possible, but the evolutions ρ that are visible during this time-step will end in different valuations of l for the first two. This observation shows that Rules HS-12 and HS-13 are not applicable to any combination of two of these three processes. Hence Rule HS-14 must be applied, which synchronizes the evolutions of the two alternatives. Applying rule HS-14 to $(\sigma^t_{\mathsf{rel}}((l = 0) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{a}}) + \sigma^t_{\mathsf{rel}}((l = 1) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{b}}))$, which occurs in the left-hand side of our target inequality, is not possible because the end-valuations of the two processes are different, and hence there is no common ρ on which to synchronize. In other words, $(\sigma^t_{\mathsf{rel}}((l = 0) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{a}}) + \sigma^t_{\mathsf{rel}}((l = 1) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{b}}))$ cannot delay for a duration t, and using rule HS-13, we conclude that the left-hand side of the inequality can delay for a duration t as process term r and become process $\tilde{\tilde{c}}$.

Regarding the right-hand side of the inequality, we find that rule HS-14 can be applied to $q + r$, resulting in a time-step with an evolution that ends in the valuation $(l = 1)$ and process term $(l = 1) \mathrel{\wedge\!\!\blacktriangle} \tilde{\tilde{b}} + \tilde{\tilde{c}}$. Subsequently, HS-14 is not applicable to the right-hand side as a whole, because there is no common ρ on which p and $q + r$ can synchronize, and rules HS-12 and HS-13 are not applicable because both p and $q + r$ can delay individually.

Hence, the left-hand side can delay with duration t and become $\tilde{\tilde{c}}$ while the right-hand side cannot delay.

3.3 Preliminaries

3.3.1 *BPA*

$BPA^{srt}_{\bot}$ is a combination of Basic Process Algebra (BPA) [BW90] extended with timing and the non-existence process constant $\bot$.

BPA can express sequential processes, i.e. processes that perform activities one after another. The set of closed process terms of BPA contains atomic actions from a set of actions 'A' representing independent activities; the deadlock process constant 'δ' representing absence of any activity; a binary operator sequential composition '.' to specify a process followed by another process; and a binary operator alternative composition '+' to specify a choice between two processes.

The set Q of all closed terms of BPA, with $q \in Q$ is given in Table 3.1.

The axioms satisfied by all processes of BPA are given in Table 3.2. BPA is not sufficient

Table 3.1: BPA-Syntax Summary ($a \in A$)

$$q ::= a \mid \delta \mid q \cdot q \mid q + q$$

Table 3.2: BPA-Axioms

$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5
$x + \delta = x$	A6
$\delta \cdot x = \delta$	A7

to specify processes for which time plays an important role. For example controllers, communication protocols etc. (see [Mee01, Ver94, Hil94, GvW01, KO94]). In order to specify time related properties of such processes, BPA must be extended.

3.3.2 BPA^{srt}

There a number of ways in which timing can be added to BPA. The decisions to be made include whether the time domain is discrete or continuous and whether the time is recorded beginning at the start of a process or a record of time elapsed between events is kept.

Basic Process Algebra for Hybrid Systems is an extension of Basic Process Algebra with standard relative timing BPA^{srt} [BM02a]. The word *standard* indicates that the time domain consists of real numbers—i.e. the time domain is dense.

In BPA^{srt}, the actions are replaced by immediate actions. Immediate actions ($\tilde{\tilde{a}}, \tilde{\tilde{b}}$) are denoted by an action label $a, b \in A$ with a double tilde $\tilde{\tilde{}}$ on it. An action $\tilde{\tilde{a}}$ performs an action immediately and terminates in the current instance of time. The deadlock process δ of BPA is replaced by immediate deadlock process $\tilde{\tilde{\delta}}$. The process $\tilde{\tilde{\delta}}$ cannot perform an action nor can it flow to a later moment in time. The relative delay operator σ_{rel} adds a delay of non-negative duration before a process. A process $\sigma^0_{\mathsf{rel}}(p)$ behaves the same as process p. The relative undelayable timeout operator ν_{rel} (which we call the *now* operator), blocks the delay behaviour of a process. A process $\nu_{rel}(p)$ performs an action immediately if p can perform an action immediately otherwise $\nu_{rel}(p)$ behaves as a deadlock process.

The set P of BPA^{srt} process terms, with $p \in P$ is given in Table 3.3.

Table 3.3: BPA^{srt}- Syntax summary ($a \in A, r > 0$)

p	$::=$	$\tilde{\tilde{a}}$	undelayable action
	$\vert$	$\tilde{\tilde{\delta}}$	undelayable deadlock constant
	$\vert$	$\sigma^0_{\mathsf{rel}}(p)$	relative delay of zero duration
	$\vert$	$\sigma^r_{\mathsf{rel}}(p)$	relative delay of duration r
	$\vert$	$p + p$	alternative composition
	$\vert$	$p \cdot p$	sequential composition
	$\vert$	$\nu_{rel}(p)$	relative undelayable timeout operator/Now operator

The axioms of BPA^{srt} are the axioms of BPA (given in Table 3.2) extended with the following axioms (see Table 3.4):

Axioms A6 and A7 are replaced by axioms A6SR and A7SR which contain the immediate deadlock constant $\tilde{\tilde{\delta}}$ instead of deadlock δ. Axiom SRT1 expresses that adding a delay of zero time units does not alter the behaviour of a process. Axiom SRT2 expresses that consecutive delays can be added. Axiom SRT3 which is called the axiom of time factorization, expresses that a delay cannot make choices. We call it the *axiom of Time Determinism*, as *Time Determinism* is a more well-known term. Since $SRT3$ is central to this chapter, we explain it in more detail in Section 3.4. Axiom SRT4 reflects that time is counted relative to the last action performed. Axiom SRU1 and SRU2 reflect the fact that the relative undelayable timeout operator ν_{rel} when applied to a process does not change its action behaviour but blocks its initial delay. The now operator distributes over choice (SRU3) and it only effects the initial behaviour of a process (SRU4).

Table 3.4: Additional axioms for BPA^{srt} $(a \in A, u, v \geq 0, r > 0)$

$x + \tilde{\tilde{\delta}} = x$	$A6SR$
$\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$	$A7SR$
$\sigma^{0}_{\mathsf{rel}}(x) = x$	$SRT1$
$\sigma^{u}_{\mathsf{rel}}(\sigma^{v}_{\mathsf{rel}}(x)) = \sigma^{u+v}_{\mathsf{rel}}(x)$	$SRT2$
$\sigma^{u}_{\mathsf{rel}}(x) + \sigma^{u}_{\mathsf{rel}}(y) = \sigma^{u}_{\mathsf{rel}}(x + y)$	$SRT3$
$\sigma^{u}_{\mathsf{rel}}(x) \cdot y = \sigma^{u}_{\mathsf{rel}}(x \cdot y)$	$SRT4$
$\nu_{rel}(\tilde{\tilde{a}}) = \tilde{\tilde{a}}$	$SRU1$
$\nu_{rel}(\sigma^{r}_{\mathsf{rel}}(x)) = \tilde{\tilde{\delta}}$	$SRU2$
$\nu_{rel}(x + y) = \nu_{rel}(x) + \nu_{rel}(y)$	$SRU3$
$\nu_{rel}(x \cdot y) = \nu_{rel}(x) \cdot y$	$SRU4$

3.3.3 $BPA^{srt}_{\perp}$

In Process Algebra with Propositional Signals [BB97], a Basic Process Algebra with Non-existence ($BPA_{\perp}$) is introduced. $BPA_{\perp}$ is BPA extended with the Non-existence process ($\perp$). In [BB97], the states in a transition system are labelled by propositions. The propositions labelling a state are supposed to hold in that state and are called *signals emitted by the state.* A false proposition can never hold. Hence a process emitting the false signal cannot exist. Here comes the need to introduce a process constant called the non-existence process denoting a process that emits a false signal.

The signature of $BPA_{\perp}$ is the signature of BPA extended with the Non-existence process. The set $Q_{\perp}$ of $BPA_{\perp}$ process terms, with $q_{\perp} \in Q_{\perp}$ is given in Table 3.5.

Table 3.5: $BPA_{\perp}$- Syntax summary $(a \in A)$

$$q_{\perp} \quad ::= \quad a \quad | \, \delta \quad | \perp \quad | \, q_{\perp} + \quad q_{\perp} \quad | \, q_{\perp} \cdot \quad q_{\perp}$$

The behaviour of the Non-existence process ($\perp$) is described by the axioms given in Table 3.6.

Table 3.6: $BPA_{\perp}$-Additional axioms $(a \in A)$

$\perp + x = \perp$	$NE1$
$\perp \cdot x = \perp$	$NE2$
$a \cdot \perp = \delta$	$NE3$

The root state of a transition system of a Non-existence process ($\perp$) is called an *inconsistent* state. Axiom NE2 expresses that an inconsistent state can never be exited and axiom NE3 reflects that it is not possible to enter such a state from a consistent one.

The axioms of $BPA_{\perp}$ are the axioms of BPA extended with the axioms $NE1, NE2$ and $NE3$ defining the Non-existence process.

Now as depicted in Figure 3.1, the Basic Timed Process Algebra with Non-existence

$BPA^{srt}_{\perp}$ is a combination of BPA^{srt} and $BPA_{\perp}$. The signature of $BPA^{srt}_{\perp}$ is the signature of BPA^{srt} (see Table 3.3), extended with the Non-existence process constant.

The set $P_{\perp}$ of $BPA^{srt}_{\perp}$ process terms, with $p_{\perp} \in P_{\perp}$ is given in Table 3.7.

Table 3.7: $BRA^{srt}_{\perp}$- Syntax summary $(a \in A, r > 0)$

$$p_{\perp} \;::=\; \tilde{\tilde{a}} \;\mid \tilde{\tilde{\delta}} \;\mid \perp \;\mid \sigma^{0}_{\mathsf{rel}}(p_{\perp}) \;\mid \sigma^{r}_{\mathsf{rel}}(p_{\perp}) \;\mid p_{\perp} + \; p_{\perp} \;\mid p_{\perp} \cdot \; p_{\perp} \;\mid \nu_{rel}(p_{\perp})$$

The axioms $BPA^{srt}_{\perp}$ include the axioms for BPA^{srt}, extended with the axioms $NE1, NE2$ and $NE3SR$. The axiom $NE3SR$ is a modification of axiom $NE3$ where an action a is replaced by an undelayable action $\tilde{\tilde{a}}$.

$$\tilde{\tilde{a}} \cdot \perp = \tilde{\tilde{\delta}} \qquad\qquad NE3SR$$

The axioms of $BPA^{srt}_{\perp}$ also include a new axiom representing the effect of the now operator ν_{rel} on non-existence.

$$\nu_{rel}(\perp) = \perp \qquad\qquad NESRU$$

A complete set of axioms of $BPA^{srt}_{\perp}$, taken from [BM05], is given in Table 3.8.

Table 3.8: Axioms of $BPA^{srt}_{\perp}$ as in [BM05] $(a \in A_{\delta}, u, v \geq 0, r > 0)$

$x + y = y + x$	$A1$	$\sigma^{0}_{\mathsf{rel}}(x) = x$	$SRT1$
$(x + y) + z = x + (y + z)$	$A2$	$\sigma^{u}_{\mathsf{rel}}(\sigma^{v}_{\mathsf{rel}}(x)) = \sigma^{u+v}_{\mathsf{rel}}(x)$	$SRT2$
$x + x = x$	$A3$	$\sigma^{u}_{\mathsf{rel}}(x) + \sigma^{u}_{\mathsf{rel}}(y) = \sigma^{u}_{\mathsf{rel}}(x + y)$	$SRT3$
$(x + y) \cdot z = x \cdot z + y \cdot z$	$A4$	$\sigma^{u}_{\mathsf{rel}}(x) \cdot y = \sigma^{u}_{\mathsf{rel}}(x \cdot y)$	$SRT4$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$A5$		
$x + \tilde{\tilde{\delta}} = x$	$A6SR$	$\nu_{rel}(\tilde{\tilde{a}}) = \tilde{\tilde{a}}$	$SRU1$
$\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$	$A7SR$	$\nu_{rel}(\sigma^{r}_{\mathsf{rel}}(x)) = \tilde{\tilde{\delta}}$	$SRU2$
		$\nu_{rel}(x + y) = \nu_{rel}(x) + \nu_{rel}(y)$	$SRU3$
$x + \perp = \perp$	$NE1$	$\nu_{rel}(x \cdot y) = \nu_{rel}(x) \cdot y$	$SRU4$
$\perp \cdot x = \perp$	$NE2$		
$\tilde{\tilde{a}} \cdot \perp = \tilde{\tilde{\delta}}$	$NE3SR$	$\nu_{rel}(\perp) = \perp$	$NESRU$

In $BPA^{srt}_{\perp}$ as presented in Process Algebra for Hybrid Systems [BM05], the axiom of time determinism SRT3 is included in the set of axioms. But as we have shown in Section 3.2, SRT3 does not hold in the semantics of [BM05]. In our first proposal of $BPA^{srt}_{\perp}$, (see Section 3.5), we replace SRT3 by a conditional axiom that exhibits time determinism in the absence of Non-existence process. This conditional axiom also holds in the semantics of [BM05] for all $BPA^{srt}_{\perp}$ process terms. Altering the axiomatization of $BPA^{srt}_{\perp}$ to fit the semantics is a straightforward solution, but due to the importance of SRT3, we also set our selves to the task of finding a semantics for $BPA^{srt}_{\perp}$, where axiom SRT3 holds. Our search results in our second proposal for $BPA^{srt}_{\perp}$, which is given in Section 3.6.

3.4 Combining Inconsistency with Time

When Basic Process Algebra with standard relative timing (BPA^{srt}) is extended with Non-existence, one is faced with the issue of how to treat inconsistent states in a timed transition system. An inconsistent state is the root state of the non-existence process constant. In this section, we discuss the design choices made in Process Algebra for Hybrid Systems [BM05] regarding the non-existence process and the effects of these choices. We also explain time determinism, the property represented by Axiom SRT3 and its significance for hybrid systems.

The semantics of a timed process algebra allows processes to evolve by performing action steps as well as time steps. In the semantics of $BPA_{\perp}$, where only action steps are present, the following axiom holds:

$$a \cdot \perp = \delta \qquad\qquad (NE3)$$

This axiom reflects that a process cannot enter into an inconsistent state after performing an action.

This view is also adopted by BPA_{hs}^{srt} [BM05].

In BPA_{hs}^{srt}, a timed counterpart of axiom $NE3$ holds:

$$\tilde{\tilde{a}} \cdot \perp = \tilde{\tilde{\delta}} \qquad\qquad (NE3SR)$$

An introduction to the semantics of BPA_{hs}^{srt} is given in the Appendix B.

In addition to action steps, time steps are also included in the semantics of BPA_{hs}^{srt}. There, like in the case of an action step, a process cannot enter into an inconsistent state after doing a time step. It is mentioned on Page 222 of [BM05], that:

> The process $\sigma_{\mathsf{rel}}^{r}(\perp)$, $(r > 0)$ is considered to be capable of idling (waiting), but only till arbitrarily close to the point of time that is reached after a period of time r. Thus, just like after performing an action, it is impossible to go on as $\perp$ after idling (waiting) for a period of time.

This characteristic of BPA_{hs}^{srt} is reflected in its transition rule HS-6 given below:

Let x be a process term, α, α' be any variable valuations, $r > 0$ and ρ be a state evolution, describing evolution of variables in the interval $[0, r]$.

$$\frac{\alpha' \in [\mathsf{s}(x)]}{\langle \sigma_{\mathsf{rel}}^{r}(x), \alpha \rangle \xmapsto{r,\rho} \langle x, \alpha' \rangle} \quad \text{HS-6}$$

(See Appendix B for the complete set of transition rules of BPA_{hs}^{srt}.)

The rule states that a process $\sigma_{\mathsf{rel}}^{r}(x)$ can wait for r time units according to any state evolution ρ and become x. The only condition is that the valuation at the end of the delay must satisfy the signal emitted by x. The signal emitted by the non-existence process ($\perp$) is false, which cannot be satisfied by any valuation. Therefore, Rule HS-6 cannot be used to derive a time step of duration r for process $\sigma_{\mathsf{rel}}^{r}(\perp)$.

Since there are no other rules applicable (see Appendix B), a time step of duration r for process $\sigma_{\mathsf{rel}}^{r}(\perp)$ cannot be derived. In BPA_{hs}^{srt}, the following predicate reflects this fact:

For any valuation α,

$$\langle \sigma_{\mathsf{rel}}^{r}(\perp), \alpha \rangle \not\xmapsto{r}$$

Rather surprisingly, a consequence of this choice in the semantics of BPA_{hs}^{srt} and its interaction with the rules for alternative composition is that the axiom of time determinism

(SRT3) does not hold. The axiom of time determinism does not hold when one of the process terms x or y is the Non-existence process or a process bisimilar to it. In Section 3.5, we show that SRT3 holds for all BPA^{srt} processes in the current semantics of [BM05].

Axiom SRT3 represents the property of time determinism, which is explained below:

> Time Determinism is an important property of a large class of timed systems. According to this property, choices between processes cannot be resolved while waiting.

Consider for example a computer application waiting for a key stroke from the user. The behaviour of the application in the future depends on which key is pressed. No decision can be made while waiting.

Axiom SRT3 reflects that as long as all operands of a choice are idling, the decision of proceeding as one operand or the other is postponed.

$$\sigma^u_{\mathsf{rel}}(x) + \sigma^u_{\mathsf{rel}}(y) = \sigma^u_{\mathsf{rel}}(x + y) \qquad u \geq 0(SRT3)$$

Time Determinism is a widely accepted property of timed systems. In many timed process algebras, counterparts to axiom SRT3 can be found. See, for example, "Algebra of Timed Processes" [NS94], Timed CCS [MT90] and Timed CSP [RR88].

In the field of hybrid systems, an interesting debate surrounding time determinism exists. In a hybrid system, two processes may allow different evolutions of variables during a delay. In such a case, the debate is whether a passage of time must resolve a choice, can resolve a choice between processes with different variable evolutions or is not allowed to resolve a choice. It has been discussed in detail in chapter 2. The reader is referred to Section 2.3 of chapter 2 for a recall.

Process Algebra for Hybrid Systems [BM05] adopts a uniform approach towards inconsistent states with regards to action and time steps. In the semantics of BPA^{srt}_{hs} an inconsistent state is unreachable by action or time steps. But a consequence of this choice together with the design of alternative composition is that the axiom $SRT3$ does not hold with the non-existence process. In Section 3.5, we present an algebra $BPA^{srt}_{\perp}$, where we abandon axiom $SRT3$ for the non-existence process constant. Altering the axiomatization of $BPA^{srt}_{\perp}$ to fit the semantics is a straightforward solution, but due to the importance of $SRT3$, we also set our selves to the task of finding a semantics for $BPA^{srt}_{\perp}$, where axiom $SRT3$ holds. In Section 3.6, we argue that in order to preserve $SRT3$ with Non-existence, the semantics of $BPA^{srt}_{\perp}$ needs to be modified. In that section, we also present the possibilities of modifications and our second proposal for $BPA^{srt}_{\perp}$. We prove that both our first and second proposals are conservative ground-extensions of BPA^{srt} and $BPA_{\perp}$.

3.5 $BPA^{srt}_{\perp}$ with conditional Time Determinism

In this Section, we present a proposal for $BPA^{srt}_{\perp}$, in which general time determinism (i.e. time determinism in all cases including the non-existence process) is **replaced** by conditional time determinism. The section is outlined as follows: first we introduce the conditional axiom that replaces the axiom of time determinism ($SRT3$) in this proposal. Then we give the semantics of this proposal in Section 3.5.2. A bisimulation is defined for this semantics and we show that bisimulation is a congruence relation. We prove that our first proposal for $BPA_{\perp}^{srt}$ is a conservative ground-extension of $BPA_{\perp}$ and BPA^{srt}. In Section 3.5.3, the axioms that are sound in this proposal are presented. At the end of this section, we prove that for all $BPA_{\perp}^{srt}$ processes, the semantics of this proposal is equivalent to the semantics of BPA_{hs}^{srt} [BM05].

3.5.1 Axioms replacing SRT3

In our first proposal for $BPA_{\perp}^{srt}$, the axiom SRT3 is replaced by two axioms: one expressing time determinism in the absence of the non-existence process (axiom SRTD) and the other expressing time non-determinism in the presence of Non-existence (axiom SRTD⊥).

Time determinism holds for all processes that are not bisimilar to the non-existence process in our first proposal for $BPA_{\perp}^{srt}$. This includes all BPA^{srt} processes. The conditional axiom of time determinism, which we call SRTD, is given below:

$$\sigma^{u}_{\mathsf{rel}}(x) + \sigma^{u}_{\mathsf{rel}}(y) = \sigma^{u}_{\mathsf{rel}}(x + y),$$

where $u \geq 0$, x, y are $BPA_{\perp}^{srt}$ processes and none of them is bisimilar to the non-existence process.

Later on in Section 3.5.2, we introduce a predicate `consistent` on process terms which holds for only those processes that are **not** bisimilar to the non-existence process.

Then Axiom SRTD can be written as:

$$\begin{array}{ll}\sigma^{u}_{\mathsf{rel}}(x) + \sigma^{u}_{\mathsf{rel}}(y) = \sigma^{u}_{\mathsf{rel}}(x + y) & \qquad (SRTD), \\ \text{where } \langle \texttt{consistent}\ x\rangle \wedge \langle \texttt{consistent}\ y\rangle & \end{array}$$

The axiom SRTD reflects that a choice between two processes that do not enter into an inconsistent state at the end of their common delay is postponed till the end of their common delay.

The set of consistent process terms in $BPA_{\perp}^{srt}$ is infinite. The set of consistent process terms in $BPA_{\perp}^{srt}$ includes all closed BPA^{srt} process terms and more. For example, the process term $\tilde{\tilde{a}} \cdot \perp$ is consistent but not a $BPA_{\perp}^{srt}$ process term. In order to replace the conditional axiom SRTD by a finite number of axioms that cover all the cases where the condition of consistency is fulfilled, we need an auxiliary operator.

The other, axiom SRTD⊥, reflects time non-determinism incase one of the processes in SRTD is a non-existence process. Axiom $SRTD\perp$ is given below:

$$\sigma^{u+r}_{\mathsf{rel}}(x) + \sigma^{r}_{\mathsf{rel}}(\perp) = \sigma^{u+r}_{\mathsf{rel}}(x) \qquad (SRTD\perp)$$

where $r > 0$,$u \geq 0$ and x is a $BPA_{\perp}^{srt}$ process term.

The axiom SRTD⊥ expresses that a delay resolves a choice between two delaying processes, when one of them enters into inconsistency earlier than the other. In that case, the process entering the inconsistency earlier is **dropped** from the choice.

Next we introduce the semantics for this proposal of $BPA_{\perp}^{srt}$.

3.5.2 Semantics

For $BPA_{\perp}^{srt}$ process terms, the semantics of this proposal is the same as that of BPA_{hs}^{srt}. However, we have simplified it in order not to burden ourselves with unnecessary notations. In $BPA_{\perp}^{srt}$, there are no environment variables whose values need to be tracked, therefore in the semantics given below, variable valuations are not included in transitions and time steps do not contain variable trajectories.

The semantics consists of four relations. They are Action Relations; Time Relations; Termination Predicates; and Consistency Predicates.

The relations are defined below:

1. Action Relations:

   $\rightarrow \subseteq P \times A \times P$

   For $(x, a, x') \in \rightarrow$, we write:

   $$\langle x \rangle \xrightarrow{a} \langle x' \rangle$$

2. Time Relations:

   $\mapsto \subseteq P \times \mathbb{R}^{>} \times P$

   For $(x, r, x') \in \mapsto$, we write:

   $$\langle x \rangle \overset{r}{\longmapsto} \langle x' \rangle$$

3. Termination Predicates:

   $\rightarrow_{\surd} \subseteq P \times A$

   For $(x, a) \in \rightarrow_{\surd}$, we write:

   $$\langle x \rangle \xrightarrow{a} \surd$$

4. Consistency:

   $\texttt{Consistent} \subseteq P$

   For $(x) \in \texttt{Consistent}$, we write:

   $$\langle \texttt{consistent}\ x \rangle$$

A predicate

$$\langle x \rangle \overset{r}{\not\longmapsto}$$

stands for the predicate $\nexists x' \in P : \langle x \rangle \overset{r}{\longmapsto} \langle x' \rangle$.

1. An action step $\langle x \rangle \xrightarrow{a} \langle x' \rangle$ represents that $\langle x \rangle$ can perform action a and proceed as term x';

2. A time step $\langle x \rangle \overset{r}{\longmapsto} \langle x' \rangle$ represents that $\langle x \rangle$ can idle for r time units and proceed as term x';

3. A termination predicate $\langle x \rangle \xrightarrow{a} \surd$ represents that $\langle x \rangle$ can perform action a and terminate;

4. A predicate $\langle \texttt{consistent}\ x \rangle$ indicates that the process term x is consistent. The consistency predicate does not hold for the non-existence process constant and all process terms bisimilar to it. For example, $\perp, \sigma^{0}_{\mathsf{rel}}(\perp), \perp + x, \perp \cdot x$, etc. It holds for all process terms that are not bisimilar to the non-existence process. For example, $\tilde{\tilde{a}}, \tilde{\tilde{\delta}}, etc$. This predicate is needed to distinguish between $\tilde{\tilde{\delta}}$ and $\perp$.

A set of transition rules for the signature of $BPA^{srt}_{\perp}$ is given in Table 3.9.

Table 3.9: Semantics of Proposal 1 for $BPA_{\perp}^{\mathrm{srt}}$ $(a \in A, r, u > 0)$

$$\frac{}{\langle \mathtt{consistent}\ \tilde{\tilde{\delta}} \rangle}\ \text{P1-}\underline{1}$$

$$\frac{}{\langle \mathtt{consistent}\ \tilde{\tilde{a}} \rangle}\ \text{P1-}\underline{2} \qquad \frac{}{\langle \tilde{\tilde{a}} \rangle \xrightarrow{a} \surd}\ \text{P1-}\underline{3}$$

$$\frac{\langle \mathtt{consistent}\ x \rangle}{\langle \mathtt{consistent}\ \sigma_{\mathsf{rel}}^{0}(x) \rangle}\ \text{P1-}\underline{4} \qquad \frac{\langle x \rangle \xrightarrow{a} \surd}{\langle \sigma_{\mathsf{rel}}^{0}(x) \rangle \xrightarrow{a} \surd}\ \text{P1-}\underline{5}$$

$$\frac{\langle x \rangle \xrightarrow{a} \langle x' \rangle}{\langle \sigma_{\mathsf{rel}}^{0}(x) \rangle \xrightarrow{a} \langle x' \rangle}\ \text{P1-}\underline{6} \qquad \frac{\langle x \rangle \xmapsto{r} \langle x' \rangle}{\langle \sigma_{\mathsf{rel}}^{0}(x) \rangle \xmapsto{r} \langle x' \rangle}\ \text{P1-}\underline{7}$$

$$\frac{}{\langle \mathtt{consistent}\ \sigma_{\mathsf{rel}}^{r}(x) \rangle}\ \text{P1-}\underline{8} \qquad \frac{}{\langle \sigma_{\mathsf{rel}}^{r+u}(x) \rangle \xmapsto{u} \langle \sigma_{\mathsf{rel}}^{r}(x) \rangle}\ \text{P1-}\underline{9}$$

$$\frac{\langle \mathtt{consistent}\ x \rangle}{\langle \sigma_{\mathsf{rel}}^{r}(x) \rangle \xmapsto{r} \langle x \rangle}\ \text{P1-}\underline{10} \qquad \frac{\langle x \rangle \xmapsto{u} \langle x' \rangle}{\langle \sigma_{\mathsf{rel}}^{r}(x) \rangle \xmapsto{r+u} \langle x' \rangle}\ \text{P1-}\underline{11}$$

$$\frac{\langle \mathtt{consistent}\ x \rangle}{\langle \mathtt{consistent}\ x \cdot y \rangle}\ \text{P1-}\underline{12} \qquad \frac{\langle x \rangle \xrightarrow{a} \langle x' \rangle}{\langle x \cdot y \rangle \xrightarrow{a} \langle x' \cdot y \rangle}\ \text{P1-}\underline{13}$$

$$\frac{\langle x \rangle \xrightarrow{a} \surd, \langle \mathtt{consistent}\ y \rangle}{\langle x \cdot y \rangle \xrightarrow{a} \langle y \rangle}\ \text{P1-}\underline{14} \qquad \frac{\langle x \rangle \xmapsto{r} \langle x' \rangle}{\langle x \cdot y \rangle \xmapsto{r} \langle x' \cdot y \rangle}\ \text{P1-}\underline{15}$$

$$\frac{\begin{array}{c}\langle \mathtt{consistent}\ x \rangle, \\ \langle \mathtt{consistent}\ y \rangle\end{array}}{\langle \mathtt{consistent}\ x + y \rangle}\ \text{P1-}\underline{16} \qquad \frac{\langle x \rangle \xrightarrow{a} \surd, \langle \mathtt{consistent}\ y \rangle}{\langle x + y \rangle \xrightarrow{a} \surd}\ \text{P1-}\underline{17}$$

$$\frac{\langle y \rangle \xrightarrow{a} \surd, \langle \mathtt{consistent}\ x \rangle}{\langle x + y \rangle \xrightarrow{a} \surd}\ \text{P1-}\underline{18} \qquad \frac{\langle x \rangle \xrightarrow{a} \langle x' \rangle, \langle \mathtt{consistent}\ y \rangle}{\langle x + y \rangle \xrightarrow{a} \langle x' \rangle}\ \text{P1-}\underline{19}$$

$$\frac{\langle y \rangle \xrightarrow{a} \langle y' \rangle, \langle \mathtt{consistent}\ x \rangle}{\langle x + y \rangle \xrightarrow{a} \langle y' \rangle}\ \text{P1-}\underline{20} \qquad \frac{\begin{array}{c}\langle x \rangle \xmapsto{r} \langle x' \rangle, \\ \langle y \rangle \xmapsto{r} \langle y' \rangle\end{array}}{\langle x + y \rangle \xmapsto{r} \langle x' + y' \rangle}\ \text{P1-}\underline{21}$$

Continued on Next Page…

Table 3.9 – Continued ($a \in A, r, u > 0$)

$$\frac{\langle x \rangle \overset{r}{\longmapsto} \langle x' \rangle, \langle y \rangle \overset{r}{\not\longmapsto}, \; \langle \texttt{consistent } y \rangle}{\langle x + y \rangle \overset{r}{\longmapsto} \langle x' \rangle} \quad \text{P1-}\underline{22} \qquad \frac{\langle y \rangle \overset{r}{\longmapsto} \langle y' \rangle, \langle x \rangle \overset{r}{\not\longmapsto}, \; \langle \texttt{consistent } x \rangle}{\langle x + y \rangle \overset{r}{\longmapsto} \langle y' \rangle} \quad \text{P1-}\underline{23}$$

$$\frac{\langle \texttt{consistent } x \rangle}{\langle \texttt{consistent } \nu_{rel}(x) \rangle} \quad \text{P1-}\underline{24} \qquad \frac{\langle x \rangle \xrightarrow{a} \surd}{\langle \nu_{\text{rel}}(x) \rangle \xrightarrow{a} \surd} \quad \text{P1-}\underline{25}$$

$$\frac{\langle x \rangle \xrightarrow{a} \langle x' \rangle}{\langle \nu_{\text{rel}}(x) \rangle \xrightarrow{a} \langle x' \rangle} \quad \text{P1-}\underline{26}$$

Next, we define a bisimulation on $BPA_{\perp}^{srt}$ process terms. Later on, we use this definition and prove that process terms that are derivably equal by the axioms given in Table 3.10 are in fact bisimilar.

Definition 2 *(Bisimulation)*

A relation $R \subseteq P \times P$ on pairs of closed process terms of $BPA_{\perp}^{srt}$ is called a bisimulation relation if and only if the following conditions hold:

For all $a \in A, r > 0, x, y, z \in P$,

1.
$$((x,y) \in R \wedge \langle x \rangle \xrightarrow{a} \langle z \rangle) \implies \exists z' \in P : \langle y \rangle \xrightarrow{a} \langle z' \rangle \textit{ and } (z, z') \in R$$

2.
$$((x,y) \in R \wedge \langle y \rangle \xrightarrow{a} \langle z \rangle) \implies \exists z' \in P : \langle x \rangle \xrightarrow{a} \langle z' \rangle \textit{ and } (z', z) \in R$$

3.
$$((x,y) \in R \wedge \langle x \rangle \overset{r}{\longmapsto} \langle z \rangle) \implies \exists z' \in P : \langle y \rangle \overset{r}{\longmapsto} \langle z' \rangle \textit{ and } (z, z') \in R$$

4.
$$((x,y) \in R \wedge \langle y \rangle \overset{r}{\longmapsto} \langle z \rangle) \implies \exists z' \in P : \langle x \rangle \overset{r}{\longmapsto} \langle z' \rangle \textit{ and } (z', z) \in R$$

5.
$$(x,y) \in R \implies (\langle x \rangle \xrightarrow{a} \surd \iff \langle y \rangle \xrightarrow{a} \surd)$$

6.
$$(x,y) \in R \implies (\langle \textit{consistent } x \rangle \iff \langle \textit{consistent } y \rangle)$$

Two process terms x and y are called bisimilar written $\langle x \rangle \leftrightarroweq \langle y \rangle$ if there exists a bisimulation relation R such that $(x,y) \in R$.

Theorem 1 *Bisimulation is a congruence for the signature of $BPA_{\perp}^{srt}$.*

Proof We use the theorem given in [Ver95], to prove that bisimulation is a congruence for the signature of $BPA_{\perp}^{srt}$. The theorem used is given below:

> Let $T = (\Sigma, D)$ be a well-founded, stratifiable term deduction system in panth format then strong bisimulation is a congruence for all function symbols occurring in Σ.

In our case, Σ is the signature of $BPA_{\perp}^{srt}$ and the set of deduction rules D is the set of rules given in Table 3.9. It is trivial to show that our term deduction system is well-founded and in PANTH format.

Below we give a function and that is a strict stratification for our term deduction system:

The function S when applied to a given transition returns the size of the source process term and when applied to a predicate returns the size of the process term on which the predicate is applied. For a process term p, the size of the process term is denoted by $|p|$.

For the given semantics, S is defined as follows:

$$\begin{array}{l} S(\langle \texttt{consistent}\ \ x\rangle) = |x| \\ S(\langle x\rangle \xrightarrow{a} \surd) = |x| \\ S(\langle x\rangle \xrightarrow{a} \langle x'\rangle) = |x| \\ S(\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle) = |x| \end{array}$$

The size of a process term is defined as follows:

$$\begin{array}{lcl} |\tilde{\tilde{a}}| & = & 1 \\ |\tilde{\tilde{\delta}}| & = & 1 \\ |\perp| & = & 1 \\ |\sigma_{\mathsf{rel}}^{0}(x)| & = & |x| + 1 \\ |\sigma_{\mathsf{rel}}^{r}(x)| & = & |x| + 1 \\ |x + y| & = & |x| + |y| \\ |x \cdot y| & = & |x| + |y| \\ |\nu_{\mathrm{rel}}(x)| & = & |x| + 1 \end{array}$$

⊠

3.5.3 Axioms

Table 3.10 contains the set of axioms that we present for this proposal. The rules for deriving $\langle \texttt{consistent}\ \ x\rangle$ are given in Table 3.9.

In Theorem 2, we prove that process terms that are derivably equal by the axioms given in Table 3.10 are bisimilar.

Theorem 2 *For all closed terms t_1, t_2 of $BPA_{\perp}^{srt}$, we have,*

$$\textit{Proposal 1} \models t_1 = t_2 \implies t_1 \leftrightarroweq t_2$$

Proof The soundness proofs of the axioms are given in Appendix H ⊠

$BPA_{\perp}^{srt}$ is a combination of theories $BPA_{\perp}$ and BPA^{srt} (see Figure 3.1). Our first proposal

Table 3.10: Proposal 1 $BPA_{\perp}^{srt}$- Axioms ($a \in A_{\delta}, u, v, v' \geq 0, r > 0$)

$x + y = y + x$	$A1$
$(x + y) + z = x + (y + z)$	$A2$
$x + x = x$	$A3$
$(x + y) \cdot z = x \cdot z + y \cdot z$	$A4$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$A5$
$x + \tilde{\tilde{\delta}} = x$	$A6SR$
$\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$	$A7SR$
$x + \perp = \perp$	$NE1$
$\perp \cdot x = \perp$	$NE2$
$\tilde{\tilde{a}} \cdot \perp = \tilde{\tilde{\delta}}$	$NE3SR$
$\sigma_{\mathsf{rel}}^{0}(x) = x$	$SRT1$
$\sigma_{\mathsf{rel}}^{u}(\sigma_{\mathsf{rel}}^{v}(x)) = \sigma_{\mathsf{rel}}^{u+v}(x)$	$SRT2$
$\sigma_{\mathsf{rel}}^{u}(x) + \sigma_{\mathsf{rel}}^{u}(y) = \sigma_{\mathsf{rel}}^{u}(x + y)$ if $\langle \texttt{consistent } x \rangle \wedge \langle \texttt{consistent } y \rangle$	$SRTD$
$\sigma_{\mathsf{rel}}^{r+u}(x) + \sigma_{\mathsf{rel}}^{r}(\perp) = \sigma_{\mathsf{rel}}^{r+u}(x)$	$SRTD\perp$
$\sigma_{\mathsf{rel}}^{u}(x) \cdot y = \sigma_{\mathsf{rel}}^{u}(x \cdot y)$	$SRT4$
$\nu_{rel}(\tilde{\tilde{a}}) = \tilde{\tilde{a}}$	$SRU1$
$\nu_{rel}(\sigma_{\mathsf{rel}}^{r}(x)) = \tilde{\tilde{\delta}}$	$SRU2$
$\nu_{rel}(x + y) = \nu_{rel}(x) + \nu_{rel}(y)$	$SRU3$
$\nu_{rel}(x \cdot y) = \nu_{rel}(x) \cdot y$	$SRU4$
$\nu_{rel}(\perp) = \perp$	$NESRU$

for $BPA_{\perp}^{srt}$ is a conservative ground-extension of $BPA_{\perp}$ and BPA^{srt}. By this we mean that Proposal 1 for $BPA_{\perp}^{srt}$ can express all process terms that can be expressed in $BPA_{\perp}$ or BPA^{srt}; for all closed terms of $BPA_{\perp}$ and BPA^{srt}, the axioms of $BPA_{\perp}$ and that of BPA^{srt} are preserved in Proposal 1 ; and finally, Proposal 1 does not introduce any new equalities among the closed process terms of $BPA_{\perp}$ or those of BPA^{srt}. Theorem 4, given below, asserts these observations.

Axiom $SRTD$ of Proposal 1 replaces the time determinism axiom $SRT3$ of BPA^{srt}. In Theorem 3, we claim that Axiom $SRTD$ covers all closed instances of Axiom $SRT3$ in BPA^{srt}. The proof of Theorem 4 uses this fact that replacing $SRT3$ by the conditional axiom $SRTD$ still covers all closed instances of BPA^{srt}.

Theorem 3 *The conditional Time Determinism axiom $SRTD$ of Table 3.10 covers all closed instances of axiom $SRT3$ for BPA^{srt} process terms.*

Proof The proof consists of the observation that for all closed BPA^{srt} process terms x, the predicate $\langle \texttt{consistent}\ x \rangle$ holds. ☒

Theorem 4 *(Conservative Ground-Extension)*

1. *Proposal 1 for $BPA_{\perp}^{srt}$ is a conservative ground-extension of $BPA_{\perp}$.*
2. *Proposal 1 for $BPA_{\perp}^{srt}$ is a conservative ground-extension of BPA_{srt}.*

Proof

1. $BPA_{\perp}$

   (a) If a and δ in the signature of $BPA_{\perp}$ (see Table 3.5) are mapped to $\tilde{\tilde{a}}$ and $\tilde{\tilde{\delta}}$, then the signature of $BPA_{\perp}^{srt}$ (see Table 3.7) extends the signature of $BPA_{\perp}$ and the axioms of Proposal 1 (see Table 3.10) include the axioms of $BPA_{\perp}$.

   Hence, Proposal 1 for $BPA_{\perp}^{srt}$ is an extension of $BPA_{\perp}$.

   (b) All other axioms in Table 3.10, i.e. Axioms $SRT1$, $SRT2$, $SRTD$, $SRTD\perp$, $SRT4$, $SRU1 - SRU4$, $NESRU$, reason about process terms that are not included in the signature of $BPA_{\perp}$.

   Hence, Proposal 1 for $BPA_{\perp}^{srt}$ is a conservative extension of $BPA_{\perp}$.

2. BPA^{srt}

   (a) The signature of $BPA_{\perp}^{srt}$ (see Table 3.7) extends the signature of BPA^{srt} (see Table 3.3).

   Axioms of BPA^{srt} (Table 3.4) , i.e. $A1 - A5$, $A6SR$, $A7SR$, $SRT1$, $SRT2$, $SRT4$, $SRU1 - SRU4$ are included in the axioms of Proposal 1 (Table 3.10). Only Axiom $SRT3$ of BPA^{srt} is not present in Table 3.10. Theorem 3 proves that all closed instances of SRT3 for BPA^{srt} process terms are covered by axiom$SRTD$.

   Hence, Proposal 1 for $BPA_{\perp}^{srt}$ is a ground-extension of BPA_{srt}.

(b) All other axioms in Table 3.10, i.e. Axioms $NE1, NE2, NE3SR$, $SRTD\bot, NESRU$, reason about process terms that are not included in the signature of BPA^{srt}. Hence, Proposal 1 for $BPA^{srt}_{\bot}$ is a conservative ground-extension of BPA^{srt}.

⊠

3.5.4 Proposal 1 versus BPA^{srt}_{hs}

In this section, we discuss the relationship between the semantics of the first proposal and the semantics of Process Algebra for Hybrid Systems [BM05]. We show that for all closed process terms of $BPA^{srt}_{\bot}$, the semantics of the first proposal is equivalent to the semantics of Basic Process Algebra for Hybrid Systems (reviewed in Section B).

When two BPA^{srt}_{hs} processes x and y are ic-bisimilar, then we denote it by $x \underline{\leftrightarrows} y$. The definition of *IC-bisimulation* is given in the Appendix B.

Theorem 5 *Let x and y be closed process terms of $BPA^{srt}_{\bot}$. Then the following holds:*

$$(\textit{Proposal 1}\ )\quad x \leftrightarrows y \iff \quad (BPA^{srt}_{hs}\ )\quad x \underline{\leftrightarrows} y$$

The set of closed process terms of $BPA^{srt}_{\bot}$ are neutral with respect to valuations. In a $BPA^{srt}_{\bot}$ process term (see the set of process terms $P_{\bot}$ in Table 3.7), there are no conditionals, signal emissions, signal evolutions or signal transitions. These operators of [BM05] allow a BPA^{srt}_{hs} process term to behave differently in different valuations. If these operators are absent from a BPA^{srt}_{hs} process term, then it cannot differentiate between valuations. Hence, Proposal 1 being equivalent to the semantics of BPA^{srt}_{hs} for $BPA^{srt}_{\bot}$ terms means that if two terms x and y are bisimilar in Proposal 1, then they are bisimilar for all valuations in the semantics of BPA^{srt}_{hs}. Not only that, but x and y are ic-bisimilar, since the target process term of a transition is again a $BPA^{srt}_{\bot}$ term–i.e. the target process term is again independent of valuations.

We prove the above theorem by proving the following:
Let x be a closed $BPA^{srt}_{\bot}$ process term.

> For every action step, time step, termination predicate and consistency predicate that is derivable for x in the semantics of Proposal 1, there exists a corresponding action step, time step, termination predicate and signal relation for x that is derivable in the semantics of BPA^{srt}_{hs}, and vice versa.

These conditions are formally stated in Theorem 6:

Theorem 6 *Let x, x' be closed process terms of $BPA^{srt}_{\bot}$, a be an action, r be a delay duration ($r > 0$), then the following holds:*

1. *(Proposal 1)* $\langle \texttt{consistent}\ x\rangle \iff (BPA^{srt}_{hs})\quad \forall\alpha : \alpha \in [\mathsf{s}(x)]$
2. *(Proposal 1)* $\langle x\rangle \xrightarrow{a} \surd \iff (BPA^{srt}_{hs})\quad \forall\alpha,\alpha' : \langle x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle$
3. *(Proposal 1)* $\langle x\rangle \xrightarrow{a} \langle x'\rangle \iff (BPA^{srt}_{hs})\quad \forall\alpha,\alpha' : \langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle$

4. (Proposal 1) $\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle \quad \Longleftrightarrow \quad (BPA_{hs}^{srt}) \quad \forall\rho : \langle x, \alpha_0^\rho\rangle \overset{r,\rho}{\longmapsto} \langle x', \alpha_r^\rho\rangle$

where, for some $t > 0$, α_t^ρ *denotes a valuation that matches with the values assigned to variables by state evolution* ρ *at time* t*.*

A consistency predicate in Proposal 1 is related to the set of signal relations with all valuations in BPA_{hs}^{srt}. An action and a termination step in Proposal 1 are related to the sets of action steps and termination steps in BPA_{hs}^{srt} with all possible pairs of source and target valuations and so on. Note that the non-derivability of a transition or predicate in Proposal 1 corresponds to non-derivability of a transition or predicate for all valuations in BPA_{hs}^{srt}.
Proof The proof of Theorem 6 is by structural induction on all closed process terms of $BPA_{\perp}^{srt}$. It is given in Appendix E. ⊠

Theorem 5 and Theorem 6 show that our Proposal 1 for $BPA_{\perp}^{srt}$ is equivalent to $BPA_{\perp}^{srt}$ presented in [BM05]. By Theorem 2, we conclude that $BPA_{\perp}^{srt}$ in [BM05] will be sound once the axiom SRT3 is replaced by the axiom $SRTD$ exhibiting conditional time determinism. The new axiom $SRTD\perp$ added to the axioms in [BM05] would reflect that a passage of time makes choices in the presence of the non-existence process constant in the semantics of [BM05].

3.5.5 Concluding Remarks

We have presented a proposal for $BPA_{\perp}^{srt}$, where we replace general time determinism by conditional time determinism. The conditional axiom $SRTD$ reflects that a passage of time does not resolve choices between process terms that are consistent now and cannot enter into an inconsistent state at the end of the delay. The behaviour of a choice between processes one of which can enter into an inconsistent state after a delay is expressed in the new axiom $SRTD\perp$. In the latter case, a passage of time makes choices in favor of the process term that stays consistent over time. The semantics of Proposal 1 coincides with that of [BM05] for $BPA_{\perp}^{srt}$ process terms. Removing the details like valuations and variable trajectories results in a concise and transparent semantics for $BPA_{\perp}^{srt}$.

3.6 Time Determinism in $BPA_{\perp}^{srt}$

In the previous chapter, we've shown that the time determinism problem of ACP_{hs}^{srt} can be solved partially, by letting time determinism apply only to consistent target processes. However, we feel there is also a need of a variant of $BPA_{\perp}^{srt}$ where the axiom of time determinism holds for all process terms including the Non-existence process. Following are our motivations for searching for a time deterministic $BPA_{\perp}^{srt}$:

1. First, in some schools of process algebra, time determinism is considered to be an essential property of all timed systems.
2. Secondly, it is clear that time determinism in all cases was the intention of the authors of Process Algebra for Hybrid Systems. On Page 222 of [BM05], the following equation is given as a derivable equation:

$$\sigma_{\mathsf{rel}}^{p+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp) = \sigma_{\mathsf{rel}}^{r}(\perp) \tag{3.1}$$

where $r > 0, p \geq 0$. This equation can only be derived if the time determinism axiom holds for the non-existence process.

3. Thirdly, we found an instance in literature where a timed process algebra is combined with the non-existence process. It is Discrete Time Process Algebra, abbreviated as PA^{drt}_{psc} defined in [BMU01]. The axiom of time determinism holds in PA^{drt}_{psc} for all processes including $\perp$.

To achieve this objective, we tried a number of approaches at finding a suitable semantics for $BPA^{srt}_{\perp}$ before we could reach a relatively satisfactory solution. The axiom of time determinism SRT3 (repeated below),

$$\sigma^u_{\mathsf{rel}}(x) + \sigma^u_{\mathsf{rel}}(y) = \sigma^u_{\mathsf{rel}}(x + y) \qquad u \geq 0 \qquad (SRT3)$$

reasons about two operators, the relative time delay operator and the alternative composition. We try to modify the semantics of these operators so that the axiom of time determinism holds in all cases.

This section is structured as follows: In Section 3.6.1, we give a brief account of choices in PA^{drt}_{psc} that enable time determinism to hold. In Sections 3.6.2, 3.6.3 and 3.6.4, three different semantics for $BPA^{srt}_{\perp}$ are given. Each semantics implements an idea for preserving time determinism with the Non-existence process. The final attempt namely "Testing for Future Inconsistency " constitutes the semantics of our second proposal of $BPA^{srt}_{\perp}$.

3.6.1 Time Determinism in PA^{psc}_{drt}

Discrete Time Process Algebra PA^{psc}_{drt} is introduced in [BMU01]. It is an extension of Process Algebra with discrete relative timing (see [BM02b]) and Process Algebra with Propositional Signals [BB97]. According to our knowledge, the process algebra PA^{psc}_{drt} is the first combination of a timed process algebra with Non-existence process.

The time domain in PA^{psc}_{drt} is discrete i.e. it is divided into slices or units. In [BMU01], the authors give ample attention to the process $\sigma(\perp)$, which is non-existence process with a unit delay operator.

In PA^{psc}_{drt}, the axiom of time determinism is expressed as follows:

$$\sigma(x) + \sigma(y) = \sigma(x + y) \qquad \text{(DRT1)}$$

where, the operator $\sigma : P \rightarrow P$ adds a delay of a unit time to a process.

There are several factors in Discrete Time Process Algebra which ensure that time determinism holds with Non-existence process. We discuss them one by one below:

1. In the semantics of PA^{drt}_{psc}, a process term with the relative delay operator σ can unconditionally do a time step. Hence, the process term $\sigma(\perp)$ can delay for a unit time and become $\perp$. The following transition can be derived:

$$\sigma(\perp) \xrightarrow{\sigma} \perp \qquad (3.2)$$

(In the semantics of PA^{psc}_{drt}, transition labels contain an extra symbol called valuation. We ignore this symbol in our discussion.)

2. Another factor contributing towards soundness of Time Determinism axiom is that a time step allows a process term to move into the next time slice and not further in time. For example,
$$\sigma(\sigma(a)) \xrightarrow{\sigma} \sigma(a)$$
is allowed. But no rule allows a time step to cross over multiple time slices. Therefore the transition,
$$\sigma(\sigma(a)) \xrightarrow{\sigma\sigma} a$$
is not allowed.

   This property combined with the semantics of alternative composition (described next) contributes to time determinism in PA^{drt}_{psc}.

3. Finally, the alternative composition is defined as follows:

   Consider an alternative composition $p + q$. It can delay as follows:

   > If both p and q can do a unit delay to the next time slice, then $p + q$ can delay for a unit time such that the choice is retained after the delay.

   For example
$$\sigma(\sigma(a)) + \sigma(\bot) \xrightarrow{\sigma} \sigma(a) + \bot$$

   > An alternative composition can proceed as one operand only if the other operand cannot perform the same time step. The root signal of the passive operand must be satisfied at the start.

   Now the process term $\sigma(a) + \bot$ cannot delay further. Because the root signal of the right operand $\bot$ is false and it cannot be satisfied.

In PA^{psc}_{drt}, like $BPA_\bot$, it is not possible to reach an inconsistent state by performing an action. In PA^{psc}_{drt}, a counterpart of axiom NE3 of $BPA_\bot$ holds:

$$\underline{\underline{a}} \cdot \bot = \underline{\underline{\delta}} \qquad (3.3)$$

(An undelayable action a and undelayable deadlock constants are denoted by $\underline{\underline{a}}$ and $\underline{\underline{\delta}}$ respectively in PA^{psc}_{drt}.)

In contrast to an action step, a process can delay for a unit time and then enter into an inconsistent state as shown in Transition 3.2. Hence, there is non-uniformity between action steps and time steps with regards to an inconsistent state. The authors of [BMU01] explain this non-uniformity as follows:

Suppose, $\sigma(\bot)$ is not allowed to delay and is put equal to deadlock.

$$\sigma(\bot) = \underline{\underline{\delta}} \qquad (3.4)$$

Then by using axioms of Basic Discrete Timed Process Algebra (given in Appendix D), Axiom NE1 and Equation 3.4, the following can be derived:

$$\begin{array}{rcll} \sigma(x) & = & \sigma(x) + \underline{\underline{\delta}} & \text{By DRT4A} \\ & = & \sigma(x) + \sigma(\bot) & \text{by Equation 3.4} \\ & = & \sigma(x + \bot) & \text{By axiom DRT1} \\ & = & \sigma(\bot) & \text{By Axiom NE1} \\ & = & \underline{\underline{\delta}} & \text{By Equation 3.4} \end{array}$$

Hence, allowing Equation 3.4 in PA^{psc}_{drt}, leads to undesirable results.

We see that certain choices in the semantics, help preserve the axiom of time determinism with the non-existence process in PA^{psc}_{drt}. Keeping in view this process algebra, we look for a time deterministic $BPA^{srt}_{\perp}$. The time domain in process algebra $BPA^{srt}_{\perp}$ is continuous which offers different challenges than the discrete time domain present in PA^{psc}_{drt}. Also, recall from Section 3.4, that in the semantics of [BM05], uniformity between action steps and time steps with regards to inconsistent states was intended. The combination of two goals, i.e. preserving axiom SRT3 unconditionally and a uniform approach towards inconsistency with regards to both action and time steps, further complicates our task. In the next sections, we describe the attempts undertaken to construct a desired semantics for $BPA^{srt}_{\perp}$.

3.6.2 Modifying Alternative Composition

In this section, we present a semantics for $BPA^{srt}_{\perp}$ where the definition of alternative composition is modified so that the axiom of Time Determinism (SRT3) holds.

The semantics of $BPA^{srt}_{\perp}$ presented here has the same transition relations as defined for $BPA^{srt}_{\perp}$ with conditional time determinism (See Section 3.5).

They are: The Action Relation ($\rightarrow$); the Time Relation ($\mapsto$); the Termination Predicates ($\rightarrow_{\surd}$); and the Consistency Predicates (`Consistent`).

The transition rules for this semantics are in Table 3.11.

Important features of this semantics are given below:

1. An inconsistent state cannot be reached after a time step.

   Consider Rule AC- 9:
   $$\frac{\langle \texttt{consistent}\ x\rangle}{\langle \sigma^{r}_{\mathsf{rel}}(x)\rangle \xmapsto{r} \langle x\rangle}$$

   Hence, the following predicate holds:

   $$\sigma^{r}_{\mathsf{rel}}(\perp) \not\xmapsto{r}$$

2. An alternative composition $p + q$ is allowed to delay in one of the following ways:

   (a) If both p and q can delay for a non-zero duration, then they delay together for a duration less than or equal to their common duration. At the end of this time step, the choice between operands is unresolved.

   For example, a process term $\sigma^{5}_{\mathsf{rel}}(\tilde{\tilde{a}}) + \sigma^{3}_{\mathsf{rel}}(\tilde{\tilde{b}})$ can delay as follows:

   $$\sigma^{5}_{\mathsf{rel}}(\tilde{\tilde{a}}) + \sigma^{3}_{\mathsf{rel}}(\tilde{\tilde{b}}) \xmapsto{3} \sigma^{2}_{\mathsf{rel}}(\tilde{\tilde{a}}) + \tilde{\tilde{b}} \tag{3.5}$$

   (b) The term $p + q$ can delay as p only if q cannot do a time step of the same duration as p. The transition system of q has a consistent root. Also q cannot do time steps of any smaller durations.

   Similarly, $p + q$ can delay as q, if p satisfies the conditions mentioned above.

   For example, a process term $\sigma^{2}_{\mathsf{rel}}(\tilde{\tilde{a}}) + \tilde{\tilde{b}}$ can delay as follows:

   $$\sigma^{2}_{\mathsf{rel}}(\tilde{\tilde{a}}) + \tilde{\tilde{b}} \xmapsto{2} \tilde{\tilde{a}} \tag{3.6}$$

(c) A time step for $p+q$ can also be a finite sequence of time steps, each of which has been obtained from one of the two ways described above.
Hence, the process term $\sigma^5_{\mathsf{rel}}(\tilde{\tilde{a}}) + \sigma^3_{\mathsf{rel}}(\tilde{\tilde{b}})$ can also delay as follows:

$$\sigma^5_{\mathsf{rel}}(\tilde{\tilde{a}}) + \sigma^3_{\mathsf{rel}}(\tilde{\tilde{b}}) \xmapsto{5} \tilde{\tilde{a}} \tag{3.7}$$

Rules AC-19, AC-20 and AC-21 define the delay behaviour of an alternative composition.

Rule AC-26 allows a time step that is a sequence of two time steps. Applying this rule a finite number of times allows one to derive a time step by appending multiple time steps.

Table 3.11: $BPA^{\mathrm{srt}}_{\perp}$-Modifying Choice ($a \in A, r, u > 0$)

$\dfrac{}{\langle \texttt{consistent}\ \tilde{\tilde{\delta}}\rangle}$	AC-$\underline{1}$	$\dfrac{}{\langle \texttt{consistent}\ \tilde{\tilde{a}}\rangle}$	AC-$\underline{2}$
$\dfrac{}{\langle \tilde{\tilde{a}}\rangle \xrightarrow{a} \surd}$	AC-$\underline{3}$	$\dfrac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle \sigma^0_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle x'\rangle}$	AC-$\underline{4}$
$\dfrac{\langle x\rangle \xrightarrow{a} \surd}{\langle \sigma^0_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \surd}$	AC-$\underline{5}$	$\dfrac{\langle x\rangle \xmapsto{r} \langle x'\rangle}{\langle \sigma^0_{\mathsf{rel}}(x)\rangle \xmapsto{r} \langle x'\rangle}$	AC-$\underline{6}$
$\dfrac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ \sigma^0_{\mathsf{rel}}(x)\rangle}$	AC-$\underline{7}$	$\dfrac{}{\langle \sigma^{r+u}_{\mathsf{rel}}(x)\rangle \xmapsto{u} \langle \sigma^r_{\mathsf{rel}}(x)\rangle}$	AC-$\underline{8}$
$\dfrac{\langle \texttt{consistent}\ x\rangle}{\langle \sigma^r_{\mathsf{rel}}(x)\rangle \xmapsto{r} \langle x\rangle}$	AC-$\underline{9}$	$\dfrac{}{\langle \texttt{consistent}\ \sigma^r_{\mathsf{rel}}(x)\rangle}$	AC-$\underline{10}$
$\dfrac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle x\cdot y\rangle \xrightarrow{a} \langle x'\cdot y\rangle}$	AC-$\underline{11}$	$\dfrac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x\cdot y\rangle \xrightarrow{a} \langle y\rangle}$	AC-$\underline{12}$
$\dfrac{\langle x\rangle \xmapsto{r} \langle x'\rangle}{\langle x\cdot y\rangle \xmapsto{r} \langle x'\cdot y\rangle}$	AC-$\underline{13}$	$\dfrac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ x\cdot y\rangle}$	AC-$\underline{14}$
$\dfrac{\langle x\rangle \xrightarrow{a} \langle x'\rangle, \langle \texttt{consistent}\ y\rangle}{\langle x+y\rangle \xrightarrow{a} \langle x'\rangle}$	AC-$\underline{15}$	$\dfrac{\langle y\rangle \xrightarrow{a} \langle y'\rangle, \langle \texttt{consistent}\ x\rangle}{\langle x+y\rangle \xrightarrow{a} \langle y'\rangle}$	AC-$\underline{16}$
$\dfrac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x+y\rangle \xrightarrow{a} \surd}$	AC-$\underline{17}$	$\dfrac{\langle y\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ x\rangle}{\langle x+y\rangle \xrightarrow{a} \surd}$	AC-$\underline{18}$
$\dfrac{\begin{array}{c}\langle x\rangle \xmapsto{r} \langle x'\rangle,\\ \langle y\rangle \xmapsto{r} \langle y'\rangle\end{array}}{\langle x+y\rangle \xmapsto{r} \langle x'+y'\rangle}$	AC-$\underline{19}$	$\dfrac{\begin{array}{l}\langle x\rangle \xmapsto{r} \langle x'\rangle,\\ \langle \texttt{consistent}\ y\rangle, \langle y\rangle \not\xmapsto{r},\\ (\forall s < r\ \ \langle y\rangle \not\xmapsto{s})\end{array}}{\langle x+y\rangle \xmapsto{r} \langle x'\rangle}$	AC-$\underline{20}$

Continued on Next Page…

Table 3.11 – Continued ($a \in A, r, u > 0$)

$\dfrac{\langle y\rangle \overset{r}{\longmapsto} \langle y'\rangle,\ \langle \texttt{consistent}\ x\rangle, \langle x\rangle \not\overset{r}{\longmapsto},\ (\forall s < r\ \ \langle x\rangle \not\overset{s}{\longmapsto})}{\langle x+y\rangle \overset{r}{\longmapsto} \langle y'\rangle}$	AC-$\underline{21}$	$\dfrac{\langle \texttt{consistent}\ x\rangle,\ \langle \texttt{consistent}\ y\rangle}{\langle \texttt{consistent}\ x+y\rangle}$	AC-$\underline{22}$
$\dfrac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle \nu_{\mathrm{rel}}(x)\rangle \xrightarrow{a} \langle x'\rangle}$	AC-$\underline{23}$	$\dfrac{\langle x\rangle \xrightarrow{a} \surd}{\langle \nu_{\mathrm{rel}}(x)\rangle \xrightarrow{a} \surd}$	AC-$\underline{24}$
$\dfrac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ \nu_{rel}(x)\rangle}$	AC-$\underline{25}$	$\dfrac{\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle, \langle x'\rangle \overset{s}{\longmapsto} \langle x''\rangle}{\langle x\rangle \overset{r+s}{\longmapsto} \langle x''\rangle}$	AC-$\underline{26}$

In the semantics of $BPA^{srt}_{\perp}$ presented in Table 3.11, the axiom of time determinism holds. In order to prove the soundness of Axiom SRT3, we need a notion of bisimulation. The semantics in this section uses exactly the same relations as used in the semantics of first proposal for $BPA^{srt}_{\perp}$. Therefore, we use Definition 2 of Section 3.5.

Theorem 7 *Axiom SRT3 is sound in the semantics of Table 3.11.*

Proof The proof is given in Appendix F. ⊠

Consider the following equation derivable from Axiom SRT3:

$$\sigma^{u+r}_{\mathsf{rel}}(x) + \sigma^{r}_{\mathsf{rel}}(\perp) = \sigma^{r}_{\mathsf{rel}}(\perp) \tag{3.8}$$

where $u \geq 0$, $r > 0$.

In the semantics of $BPA^{srt}_{\perp}$ presented in this section, Equation 3.8 holds. The time steps derivable in this semantics for the left and right hand sides of Equation 3.8 are given below:

1. $\sigma^{r}_{\mathsf{rel}}(\perp)$:

   An inconsistent state is not reachable by a time step. Rule AC-9 cannot be applied to $\sigma^{r}_{\mathsf{rel}}(\perp)$. Since no other rules are applicable, hence we infer:

   $$\langle \sigma^{r}_{\mathsf{rel}}(\perp)\rangle \not\overset{r}{\longmapsto}$$

   For all $s, t > 0$, such that $r = s + t$, Rule AC-8 can derive the following time step for $\sigma^{r}_{\mathsf{rel}}(\perp)$:

   $$\langle \sigma^{s+t}_{\mathsf{rel}}(\perp)\rangle \overset{s}{\longmapsto} \langle \sigma^{t}_{\mathsf{rel}}(\perp)\rangle$$

   After each such time step, the resulting process term $\sigma^{t}_{\mathsf{rel}}(\perp)$ is again delayable. The time domain in $BPA^{srt}_{\perp}$ is dense. I.e. between any two real numbers, there exists an infinite number of real numbers. Hence, for any $t > 0$, there exist infinitely many numbers that are greater than zero and less than t. The process $\sigma^{r}_{\mathsf{rel}}(\perp)$ can do an arbitrary number of time steps getting closer and closer to $\sigma^{0}_{\mathsf{rel}}(\perp)$ but not reaching it.

2. $\sigma_{\mathsf{rel}}^{u+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp)$:

   A time step for process term $\sigma_{\mathsf{rel}}^{u+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp)$ can be derived by Rule AC-19 and Rule AC-26. Applying Rule AC-19 on $\sigma_{\mathsf{rel}}^{u+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp)$, we can derive the following time steps:

   For all $s, t > 0$, such that $r = s + t$:

   $$\langle \sigma_{\mathsf{rel}}^{u+s+t}(x) + \sigma_{\mathsf{rel}}^{s+t}(\perp) \rangle \overset{s}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{u+t}(x) + \sigma_{\mathsf{rel}}^{t}(\perp) \rangle$$

   Taking closure of successive time steps obtained by Rule AC-19 using Rule AC-26 results in a time step that has a target process term of the form $\sigma_{\mathsf{rel}}^{u+t}(x) + \sigma_{\mathsf{rel}}^{t}(\perp)$ for some $t > 0$.

   Rule AC-20 (similarly Rule AC-21) is not applicable on $\sigma_{\mathsf{rel}}^{u+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp)$ as the condition that the right (left) alternative cannot delay at all is not satisfied. Also Rule AC-20 (similarly Rule AC-21) is not applicable on the target process terms of any transitions that have been derived by applying Rule AC-19 and Rule AC-26 on $\sigma_{\mathsf{rel}}^{u+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp)$.

We conclude that the left hand side and right hand side of Equation 3.8 are bisimilar in this semantics.

Contrary to the approach here, in Proposal 1 of $BPA_{\perp}^{srt}$ (see Section 3.5), a delay is allowed to resolve a choice between two delaying processes, when one of them enters into inconsistency earlier than the other. In that case, the process entering the inconsistency earlier is dropped from the choice. Axiom $SRTD\perp$ of Proposal 1, which is repeated below, reflects this fact:

$$\sigma_{\mathsf{rel}}^{u+r}(x) + \sigma_{\mathsf{rel}}^{r}(\perp) = \sigma_{\mathsf{rel}}^{r+u}(x) \qquad (SRTD\perp)$$

Considering the semantics of Table 3.11 a suitable semantics for a time deterministic $BPA_{\perp}^{srt}$, we investigate into further extending this semantics with other operators of BPA_{hs}^{srt} that have been defined in [BM05]. We find a problem in extending it with integration.

Integration represents an alternative composition over an infinite set of alternatives. It is briefly explained in Section 3.7.1. The set of transition rules for integration and axioms holding in BPA^{srt} [BM02a] are given in Appendix C.

The semantic rules for integration in BPA^{srt} are defined along the same lines as alternative composition. Following this approach, in the above semantics of $BPA_{\perp}^{srt}$, a rule for deriving a time step for an integral $\int_{u \in U} F(u)$ would allow it to delay for a duration that is less than or equal to a duration common among all delaying process terms in the set $\{F(p) \mid p \in U\}$. For example, for the above semantics, the following seems to be a good candidate of a transition rule allowing a process term $\int_{u \in U} F(u)$ to delay:

We call it Rule <u>**AC**</u>**-27**:

$$\frac{\begin{array}{l} \{F(q) \overset{r}{\longmapsto} F_1(q) \mid q \in U_1\} \\ \vdots \\ \{F(q) \overset{r}{\longmapsto} F_n(q) \mid q \in U_n\} \\ \{F(q) \overset{r}{\not\longmapsto}, \langle \texttt{consistent}\ \ F(q) \rangle, \\ \forall s < r, \quad F(q) \overset{s}{\not\longmapsto} \mid q \in U_{n+1}\} \end{array}}{\int_{u \in U} F(u) \overset{r}{\longmapsto} \int_{u \in U_1} F_1(u) + \ldots + \int_{u \in U_n} F_n(u)} \quad \begin{array}{l} \{U_1, \ldots U_n\} \text{ is a partition of} \\ U \backslash U_{n+1}, U_{n+1} \subset U \end{array}$$

The above Rule indicates that a process term $\int_{u \in U} F(u)$ can delay for a certain duration, if for a nonempty subset U' of U, all process terms $F(q)$, with $q \in U'$ can delay for that duration. The set U' is partitioned into n sets $\{U_1 \ldots U_n\}$. For each set U_i, $F(q)$ with $q \in U_i$ may evolve into a different process term after the delay. Whereas all process terms $F(q)$, with $q \in U \backslash U'$ have consistent roots and cannot perform a delay of that duration or any smaller delay.

Now consider the process term $\int_{u>0} \sigma^u_{\mathsf{rel}}(\tilde{\tilde{a}})$. In the set $\{\sigma^u_{\mathsf{rel}}(\tilde{\tilde{a}}) \mid u > 0\}$, all process terms are delayable, but we cannot determine a delay duration that is common among all members of the set as there does not exist a smallest real number greater than zero. Hence, when the semantics of $BPA^{srt}_{\perp}$ under discussion is extended with integration, then a time step for the process $\int_{u>0} \sigma^u_{\mathsf{rel}}(\tilde{\tilde{x}})$ cannot be derived. In fact, when we extend this semantics with integration, we find that the following equation holds:

$$\int_{u>0} \sigma^u_{\mathsf{rel}}(x) = \tilde{\tilde{\delta}} \tag{3.9}$$

for any process term x.

By modifying alternative composition as proposed above in $BPA^{srt}_{\perp}$, we can save the axiom of time determinism, but then we cannot add integration to this modified $BPA^{srt}_{\perp}$ as it has been added to BPA^{srt}.

Hence, we decide to look for other approaches for preserving time determinism with Non-existence process in $BPA^{srt}_{\perp}$.

3.6.3 Modifying the Relative Delay Operator σ^r_{rel}

Now we focus on changing the semantics of the other operator in the axiom of time determinism, i.e. the relative delay operator.

In the semantics presented here, we modify the semantics of the delay operator σ^r_{rel}, so that an inconsistent state is reachable by a time step. The semantics uses the same relations as the semantics for $BPA^{srt}_{\perp}$ with conditional time determinism (see Section 3.5), i.e. the Action Relation ($\rightarrow$), the Time Relation ($\mapsto$), the Termination Predicates ($\rightarrow_{\surd}$) and the Consistency Predicates (`Consistent`).

The transition rules for this semantics are given in Table 3.12.

We discuss the important features of this semantics below:

1. The semantics of the delay operator σ^r_{rel}, with $r > 0$, has been modified. Now a process term $\sigma^r_{\mathsf{rel}}(x)$ can delay unconditionally for r time units. Hence, the process term $\sigma^t_{\mathsf{rel}}(\perp)$ can delay for t seconds and become $\perp$. The following transition is derivable:

$$\sigma^r_{\mathsf{rel}}(\perp) \overset{r}{\mapsto} \perp$$

2. In order to preserve Axiom SRT3, the delay behaviour of an alternative composition is defined as follows:

   Consider an alternative composition, $p + q$. It can delay in one of the following ways:

   (a) If both p and q can delay for a given duration, then they delay together and the choice is retained at the end of their common delay.

   (b) The term $p + q$ can delay as p only if q cannot do a time step of the same duration as p; the transition system of q has a consistent root; also if q can do time steps of a **smaller** duration then all such time steps of q must end in processes with consistent states.

(c) Similarly, $p+q$ can behave also as q if p fulfills the conditions mentioned above for q. I.e. p cannot do a time step of the same duration as q; the transition system of p has a consistent root; also if p can do time steps of a **smaller** durations then all such time steps of p must end in processes with consistent states.

Rules RI-20, RI-21 and RI-22 define the delay behaviour of an alternative composition.

Table 3.12: $BPA^{\mathrm{srt}}_{\perp}$-Modifying Relative Delay $\sigma^{r}_{\mathsf{rel}}$ ($a \in A, r, u > 0$)

$\frac{}{\langle \texttt{consistent}\ \tilde{\tilde{\delta}}\rangle}$	RI-$\underline{1}$	$\frac{}{\langle \texttt{consistent}\ \tilde{\tilde{a}}\rangle}$	RI-$\underline{2}$
$\frac{}{\langle \tilde{\tilde{a}}\rangle \xrightarrow{a} \surd}$	RI-$\underline{3}$	$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle \sigma^{0}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle x'\rangle}$	RI-$\underline{4}$
$\frac{\langle x\rangle \xrightarrow{a} \surd}{\langle \sigma^{0}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \surd}$	RI-$\underline{5}$	$\frac{\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle}{\langle \sigma^{0}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle x'\rangle}$	RI-$\underline{6}$
$\frac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ \sigma^{0}_{\mathsf{rel}}(x)\rangle}$	RI-$\underline{7}$	$\frac{}{\langle \sigma^{r+u}_{\mathsf{rel}}(x)\rangle \overset{u}{\longmapsto} \langle \sigma^{r}_{\mathsf{rel}}(x)\rangle}$	RI-$\underline{8}$
$\frac{}{\langle \sigma^{r}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle x\rangle}$	RI-$\underline{9}$	$\frac{\langle x\rangle \overset{u}{\longmapsto} \langle x'\rangle}{\langle \sigma^{r}_{\mathsf{rel}}(x)\rangle \overset{r+u}{\longmapsto} \langle x'\rangle}$	RI-$\underline{10}$
$\frac{}{\langle \texttt{consistent}\ \sigma^{r}_{\mathsf{rel}}(x)\rangle}$	RI-$\underline{11}$	$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle x \cdot y\rangle \xrightarrow{a} \langle x' \cdot y\rangle}$	RI-$\underline{12}$
$\frac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x \cdot y\rangle \xrightarrow{a} \langle y\rangle}$	RI-$\underline{13}$	$\frac{\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle}{\langle x \cdot y\rangle \overset{r}{\longmapsto} \langle x' \cdot y\rangle}$	RI-$\underline{14}$
$\frac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ x \cdot y\rangle}$	RI-$\underline{15}$	$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle, \langle \texttt{consistent}\ y\rangle}{\langle x + y\rangle \xrightarrow{a} \langle x'\rangle}$	RI-$\underline{16}$
$\frac{\langle x\rangle \xrightarrow{a} \langle y'\rangle, \langle \texttt{consistent}\ x\rangle}{\langle x + y\rangle \xrightarrow{a} \langle y'\rangle}$	RI-$\underline{17}$	$\frac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x + y\rangle \xrightarrow{a} \surd}$	RI-$\underline{18}$
$\frac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x + y\rangle \xrightarrow{a} \surd}$	RI-$\underline{19}$	$\frac{\begin{array}{c}\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle, \\ \langle y\rangle \overset{r}{\longmapsto} \langle y'\rangle\end{array}}{\langle x + y\rangle \overset{r}{\longmapsto} \langle x' + y'\rangle}$	RI-$\underline{20}$

Continued on Next Page...

Table 3.12 – Continued ($a \in A, r, u > 0$)

$$\frac{\begin{array}{l}\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle,\\ \langle \texttt{consistent}\ y\rangle, \langle y\rangle \overset{r}{\not\longmapsto},\\ (\forall y', \forall s < r\ \ \langle y\rangle \overset{s}{\longmapsto} \langle y'\rangle\\ \ \Longrightarrow\ \langle \texttt{consistent}\ y'\rangle)\end{array}}{\langle x + y\rangle \overset{r}{\longmapsto} \langle x'\rangle}\ \text{RI-}\underline{21} \qquad \frac{\begin{array}{l}\langle y\rangle \overset{r}{\longmapsto} \langle y'\rangle,\\ \langle \texttt{consistent}\ x\rangle, \langle x\rangle \overset{r}{\not\longmapsto},\\ (\forall x', \forall s < r\ \ \langle x\rangle \overset{s}{\longmapsto} \langle x'\rangle\\ \ \Longrightarrow\ \langle \texttt{consistent}\ x'\rangle)\end{array}}{\langle x + y\rangle \overset{r}{\longmapsto} \langle y'\rangle}\ \text{RI-}\underline{22}$$

$$\frac{\begin{array}{c}\langle \texttt{consistent}\ x\rangle,\\ \langle \texttt{consistent}\ y\rangle\end{array}}{\langle \texttt{consistent}\ x + y\rangle}\ \text{RI-}\underline{23} \qquad \frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle \nu_{\mathrm{rel}}(x)\rangle \xrightarrow{a} \langle x'\rangle}\ \text{RI-}\underline{24}$$

$$\frac{\langle x\rangle \xrightarrow{a} \surd}{\langle \nu_{\mathrm{rel}}(x)\rangle \xrightarrow{a} \surd}\ \text{RI-}\underline{25} \qquad \frac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ \nu_{rel}(x)\rangle}\ \text{RI-}\underline{26}$$

The transition rules defining alternative composition have a different format then the standard tyft format. In Rules RI-21 and RI-22, a universal quantifier on process terms is required to include all possible target terms of time transitions of all possible smaller durations. The rules come under the Mousavi-Reniers [MR07] UNTyft format.

The axiom of time determinism holds in the above semantics. The notion of bisimulation used here is the same as defined for Proposal 1 for $BPA_{\perp}^{srt}$ in Section 3.5.

Theorem 8 *Axiom SRT3 holds in the semantics of Table 3.12.*

Proof The proof is given in the Appendix G. ☒

This appears to be a suitable semantics for a time deterministic $BPA_{\perp}^{srt}$. The only drawback is that making an inconsistent state reachable by a time step is not uniform with the approach adopted for action steps. Compare Rule RI-13 in Table 3.12 with Rule RI-9. From Rule RI-13 it is not possible to enter into an inconsistent state after performing an action. We find an answer to this dilemma in the next section.

3.6.4 Testing for Future Inconsistency

In this section, we introduce a semantics for $BPA_{\perp}^{srt}$, that adds a new predicate relation to the semantics for $BPA_{\perp}^{srt}$ with conditional time determinism (see Section 3.5). We call this approach *Testing for future inconsistency.* The new predicate relation that is added to the semantics checks whether a process term can enter into an inconsistent state after a delay.

This semantics uses five relations. They are: The Action Relation ($\rightarrow$); the Time Relation ($\mapsto$); the Termination Predicate ($\rightarrow_{\surd}$); the Consistency Predicates (`Consistent`); and the Future Inconsistency Predicates ($\mapsto_{\perp}$).

The relation "Future Inconsistency" is defined on pairs of process terms and durations. It is denoted by $\mapsto_{\perp}$.

$$\mapsto_{\perp} \subseteq P \times \mathbb{R}^{>}$$

For $(x, r) \in \mapsto_{\perp}$, we write:

$$\langle x \rangle \overset{r}{\mapsto}_{\perp}$$

A future inconsistency predicate $\langle x \rangle \overset{r}{\mapsto}_{\perp}$ represents that if allowed to delay, the transition system of $\langle x \rangle$ would enter into an inconsistent state after r units of time.

For example, the following predicate holds:

$$\langle \sigma^{r}_{\mathsf{rel}}(\perp) \rangle \overset{r}{\mapsto}_{\perp}$$

The motivation for adding Future Inconsistency Predicates in the semantics comes from the previous section. Consider the rules in Table 3.12. When x is not bisimilar to the Nonexistence process constant, then the transition $\sigma^{r}_{\mathsf{rel}}(\perp) \overset{r}{\mapsto} \perp$ is different from $\sigma^{r}_{\mathsf{rel}}(x) \overset{r}{\mapsto} x$. For example, the transition $\sigma^{r}_{\mathsf{rel}}(\perp) \overset{r}{\mapsto} \perp$ has a different effect on the definition of alternative composition. We might as well keep an inconsistent state unreachable after a time step and produce the same effect on alternative composition by adding a new predicate relation in the semantics. We call this Predicate Relation the *Future Inconsistency* predicate relation.

The two semantics of $BPA^{srt}_{\perp}$, one presented in this section and the other "Modifying the Relative Delay Operator σ_{rel}" (see Section 3.6.3) are claimed without proof to be isomorphic, with two process terms being bisimilar in one semantics if and only if they are bisimilar in the other.

This is the semantics we adopt for a time deterministic $BPA^{srt}_{\perp}$. The transition rules for this semantics are given in Table 3.13.

Some important features of this semantics regarding the operators of time determinism axiom are described below:

1. In this approach inconsistent states are unreachable. I.e., the following predicate holds:

$$\sigma^{r}_{\mathsf{rel}}(\perp) \overset{r}{\not\mapsto}$$

2. The delay behaviour of an alternative composition is defined as follows: Consider an alternative composition $p + q$. It can delay as follows:

   (a) If both p and q can delay for a given duration, then they delay together and the choice is retained at the end of their common delay.

   (b) The term $p + q$ can delay as p only if q cannot do a time step of the same duration as p. The transition system of q has a consistent root. Also q cannot does not have an inconsistency predicate of duration less than or equal to that of the delay of p.

   (c) Also allow $p + q$ to delay as q, if p satisfies the conditions mentioned above.

   Rules P2-24, P2-25 and P2-26 of Table 3.13 define the delay behaviour of an alternative composition.

As we see furtheron, the axiom of time determinism holds in this semantics.

Table 3.13: Semantics of Proposal 2 for $BPA_{\perp}^{\mathrm{srt}}(a \in A, r, u > 0)$

$$\frac{}{\langle \texttt{consistent}\ \tilde{\tilde{\delta}}\rangle}\quad \text{P2-}\underline{1}$$

$$\frac{}{\langle \texttt{consistent}\ \tilde{\tilde{a}}\rangle}\quad \text{P2-}\underline{2} \qquad \frac{}{\langle \tilde{\tilde{a}}\rangle \xrightarrow{a} \surd}\quad \text{P2-}\underline{3}$$

$$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \xrightarrow{a} \langle x'\rangle}\quad \text{P2-}\underline{4} \qquad \frac{\langle x\rangle \xrightarrow{a} \surd}{\langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \xrightarrow{a} \surd}\quad \text{P2-}\underline{5}$$

$$\frac{\langle x\rangle \xmapsto{r} \langle x'\rangle}{\langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \xmapsto{r} \langle x'\rangle}\quad \text{P2-}\underline{6} \qquad \frac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ \sigma_{\mathsf{rel}}^{0}(x)\rangle}\quad \text{P2-}\underline{7}$$

$$\frac{\langle x\rangle \xmapsto{r} \perp}{\langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \xmapsto{r} \perp}\quad \text{P2-}\underline{8}$$

$$\frac{}{\langle \sigma_{\mathsf{rel}}^{r+u}(x)\rangle \xmapsto{u} \langle \sigma_{\mathsf{rel}}^{r}(x)\rangle}\quad \text{P2-}\underline{9} \qquad \frac{\langle \texttt{consistent}\ x\rangle}{\langle \sigma_{\mathsf{rel}}^{r}(x)\rangle \xmapsto{r} \langle x\rangle}\quad \text{P2-}\underline{10}$$

$$\frac{\langle x\rangle \xmapsto{u} \langle x'\rangle}{\langle \sigma_{\mathsf{rel}}^{r}(x)\rangle \xmapsto{r+u} \langle x'\rangle}\quad \text{P2-}\underline{11} \qquad \frac{}{\langle \texttt{consistent}\ \sigma_{\mathsf{rel}}^{r}(x)\rangle}\quad \text{P2-}\underline{12}$$

$$\frac{\neg\langle \texttt{consistent}\ x\rangle}{\langle \sigma_{\mathsf{rel}}^{r}(x)\rangle \xmapsto{r} \perp}\quad \text{P2-}\underline{13} \qquad \frac{\langle x\rangle \xmapsto{u} \perp}{\langle \sigma_{\mathsf{rel}}^{r}(x)\rangle \xmapsto{r+u} \perp}\quad \text{P2-}\underline{14}$$

$$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle x\cdot y\rangle \xrightarrow{a} \langle x'\cdot y\rangle}\quad \text{P2-}\underline{15} \qquad \frac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x\cdot y\rangle \xrightarrow{a} \langle y\rangle}\quad \text{P2-}\underline{16}$$

$$\frac{\langle x\rangle \xmapsto{r} \langle x'\rangle}{\langle x\cdot y\rangle \xmapsto{r} \langle x'\cdot y\rangle}\quad \text{P2-}\underline{17} \qquad \frac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ x\cdot y\rangle}\quad \text{P2-}\underline{18}$$

$$\frac{\langle x\rangle \xmapsto{r} \perp}{\langle x\cdot y\rangle \xmapsto{r} \perp}\quad \text{P2-}\underline{19}$$

$$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle, \langle \texttt{consistent}\ y\rangle}{\langle x+y\rangle \xrightarrow{a} \langle x'\rangle}\quad \text{P2-}\underline{20} \qquad \frac{\langle y\rangle \xrightarrow{a} \langle y'\rangle, \langle \texttt{consistent}\ x\rangle}{\langle x+y\rangle \xrightarrow{a} \langle y'\rangle}\quad \text{P2-}\underline{21}$$

$$\frac{\langle x\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ y\rangle}{\langle x+y\rangle \xrightarrow{a} \surd}\quad \text{P2-}\underline{22} \qquad \frac{\langle y\rangle \xrightarrow{a} \surd, \langle \texttt{consistent}\ x\rangle}{\langle x+y\rangle \xrightarrow{a} \surd}\quad \text{P2-}\underline{23}$$

Continued on Next Page…

Table 3.13 – Continued ($a \in A, r, u > 0$)

$$\frac{\begin{array}{c}\langle x\rangle \overset{r}{\mapsto} \langle x'\rangle,\\ \langle y\rangle \overset{r}{\mapsto} \langle y'\rangle\end{array}}{\langle x+y\rangle \overset{r}{\mapsto} \langle x'+y'\rangle}\ \text{P2-}\underline{24}$$

$$\frac{\begin{array}{l}\langle x\rangle \overset{r}{\mapsto} \langle x'\rangle, \langle \texttt{consistent}\ y\rangle,\\ \langle y\rangle \overset{r}{\not\mapsto}, \forall s \leq r(\langle y\rangle \overset{s}{\not\mapsto}_{\perp})\end{array}}{\langle x+y\rangle \overset{r}{\mapsto} \langle x'\rangle}\ \text{P2-}\underline{25} \qquad \frac{\begin{array}{l}\langle y\rangle \overset{r}{\mapsto} \langle y'\rangle, \langle \texttt{consistent}\ x\rangle,\\ \langle x\rangle \overset{r}{\not\mapsto}, \forall s \leq r(\langle x\rangle \overset{s}{\not\mapsto}_{\perp})\end{array}}{\langle x+y\rangle \overset{r}{\mapsto} \langle y'\rangle}\ \text{P2-}\underline{26}$$

$$\frac{\begin{array}{c}\langle \texttt{consistent}\ x\rangle,\\ \langle \texttt{consistent}\ y\rangle\end{array}}{\langle \texttt{consistent}\ x+y\rangle}\ \text{P2-}\underline{27}$$

$$\frac{\begin{array}{l}\langle x\rangle \overset{r}{\mapsto}_{\perp}, \langle \texttt{consistent}\ y\rangle,\\ \forall s < r(\langle y\rangle \overset{s}{\not\mapsto}_{\perp})\end{array}}{\langle x+y\rangle \overset{r}{\mapsto}_{\perp}}\ \text{P2-}\underline{28} \qquad \frac{\begin{array}{l}\langle y\rangle \overset{r}{\mapsto}_{\perp}, \langle \texttt{consistent}\ x\rangle,\\ \forall s < r(\langle x\rangle \overset{s}{\not\mapsto}_{\perp})\end{array}}{\langle x+y\rangle \overset{r}{\mapsto}_{\perp}}\ \text{P2-}\underline{29}$$

$$\frac{\langle x\rangle \xrightarrow{a} \langle x'\rangle}{\langle \nu_{\mathrm{rel}}(x)\rangle \xrightarrow{a} \langle x'\rangle}\ \text{P2-}\underline{30} \qquad \frac{\langle x\rangle \xrightarrow{a} \surd}{\langle \nu_{\mathrm{rel}}(x)\rangle \xrightarrow{a} \surd}\ \text{P2-}\underline{31}$$

$$\frac{\langle \texttt{consistent}\ x\rangle}{\langle \texttt{consistent}\ \nu_{\mathrm{rel}}(x)\rangle}\ \text{P2-}\underline{32}$$

Next we define a bisimulation on $BPA_{\perp}^{srt}$ process terms that relates two process terms when they have exactly the same behaviour in the semantics presented above. Later on, when we present the set of axioms, we use this definition and prove that process terms that are set equal by the axioms are in fact semantically bisimilar. The definition of bisimulation for the semantics presented in this section is obtained by adding a comparison of future inconsistency predicates in the bisimulation definition of $BPA_{\perp}^{srt}$ with conditional time determinism (see Definition 2).

It is defined as follows:

Definition 3 *A relation $R \subseteq P \times P$ is called a bisimulation relation if and only if the following conditions hold:*

For all $a \in A, r > 0, x, y, z \in P$,

1.

$$((x,y) \in R \wedge \langle x\rangle \xrightarrow{a} \langle z\rangle) \implies \exists z' \in P : \langle y\rangle \xrightarrow{a} \langle z'\rangle \textit{ and } (z,z') \in R$$

2.

$$((x,y) \in R \wedge \langle y\rangle \xrightarrow{a} \langle z\rangle) \implies \exists z' \in P : \langle x\rangle \xrightarrow{a} \langle z'\rangle \textit{ and } (z',z) \in R$$

3.

$$((x,y) \in R \wedge \langle x \rangle \overset{r}{\longmapsto} \langle z \rangle) \implies \exists z' \in P : \langle y \rangle \overset{r}{\longmapsto} \langle z' \rangle \text{ and } (z,z') \in R$$

4.

$$((x,y) \in R \wedge \langle y \rangle \overset{r}{\longmapsto} \langle z \rangle) \implies \exists z' \in P : \langle x \rangle \overset{r}{\longmapsto} \langle z' \rangle \text{ and } (z',z) \in R$$

5.

$$(x,y) \in R \implies (\langle x \rangle \xrightarrow{a} \surd \iff \langle y \rangle \xrightarrow{a} \surd)$$

6.

$$(x,y) \in R \implies (\langle x \rangle \overset{r}{\longmapsto}_{\perp} \iff \langle y \rangle \overset{r}{\longmapsto}_{\perp})$$

7.

$$(x,y) \in R \implies (\langle \texttt{consistent } x \rangle \iff \langle \texttt{consistent } y \rangle)$$

Two process terms x and y are called bisimilar to each other written as $\langle x \rangle \leftrightarroweq \langle y \rangle$ if there exists a bisimulation relation R such that $(x,y) \in R$.

Theorem 9 *Bisimulation is a congruence for the signature of $BPA_{\perp}^{srt}$.*

Proof This theorem is proven on the same lines as Theorem 1 using congruence proof in [Ver95].

It is trivial to show that the given term deduction system with the set of deduction rules given in Table 3.13 is well-founded and all the transition rules are in PANTH format.

A stratification S is defined below which is an extension of the stratification function S given in the proof of Theorem 1. S is a strict stratification for the term deduction system under study.

The stratification S is defined as follows:

$$\begin{array}{lcl} S(\langle \texttt{consistent } x \rangle) & = & |x| \\ S(\langle x \rangle \xrightarrow{a} \surd) & = & |x| \\ S(\langle x \rangle \xrightarrow{a} \langle x' \rangle) & = & |x| \\ S(\langle x \rangle \overset{r}{\longmapsto} \langle x' \rangle) & = & |x| \\ S(\langle x \rangle \overset{r}{\longmapsto}_{\perp}) & = & |x| \end{array}$$

⊠

Table 3.14 consists a list of axioms we present for this proposal.

Theorem 10 *(Soundness of Proposal 2)*

For all closed terms t_1, t_2 of $BPA_{\perp}^{srt}$, we have,

$$\textit{Proposal 2} \models t_1 = t_2 \implies t_1 \leftrightarroweq t_2$$

Proof The soundness proofs of the axioms are given in Appendix I ⊠

Theorem 11 *Our Proposal 2 for $BPA_{\perp}^{srt}$ is a conservative extension of BPA^{srt} and BPA.*

Table 3.14: Proposal 2 $BPA_{\perp}^{srt}$-Axioms ($a \in A_{\delta} u, v \geq 0$)

$x + y = y + x$	$A1$	$\sigma_{\mathsf{rel}}^{0}(x) = x$	$SRT1$
$(x + y) + z = x + (y + z)$	$A2$	$\sigma_{\mathsf{rel}}^{u}(\sigma_{\mathsf{rel}}^{v}(x)) = \sigma_{\mathsf{rel}}^{u+v}(x)$	$SRT2$
$x + x = x$	$A3$	$\sigma_{\mathsf{rel}}^{u}(x) + \sigma_{\mathsf{rel}}^{u}(y) = \sigma_{\mathsf{rel}}^{u}(x + y)$	$SRT3$
$(x + y) \cdot z = x \cdot z + y \cdot z$	$A4$	$\sigma_{\mathsf{rel}}^{u}(x) \cdot y = \sigma_{\mathsf{rel}}^{u}(x \cdot y)$	$SRT4$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$A5$		
$x + \tilde{\tilde{\delta}} = x$	$A6SR$	$\nu_{rel}(\tilde{\tilde{a}}) = \tilde{\tilde{a}}$	$SRU1$
$\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$	$A7SR$	$\nu_{rel}(\sigma_{\mathsf{rel}}^{r}(x)) = \tilde{\tilde{\delta}}$	$SRU2$
		$\nu_{rel}(x + y) = \nu_{rel}(x) + \nu_{rel}(y)$	$SRU3$
$x + \perp = \perp$	$NE1$	$\nu_{rel}(x \cdot y) = \nu_{rel}(x) \cdot y$	$SRU4$
$\perp \cdot x = \perp$	$NE2$		
$\tilde{\tilde{a}} \cdot \perp = \tilde{\tilde{\delta}}$	$NE3SR$	$\nu_{rel}(\perp) = \perp$	$NESRU$

Proof Trivial ☒

This concludes our research for our possible second proposal for $BPA_{\perp}^{srt}$, where the time determinism axiom SRT3 holds unconditionally.

3.6.5 Concluding Remarks

Observing the importance of time determinism for timed systems, we set out to finding a time deterministic proposal of $BPA_{\perp}^{srt}$. We have presented three attempts of preserving time determinism axiom in $BPA_{\perp}^{srt}$. Each attempt proposes a modification of or addition to the semantics of $BPA_{\perp}^{srt}$ with conditional time determinism (see Section 3.5). The first attempt (Section 3.6.2) proposing a modification in alternative composition was given up due to its limitations regarding the extension with integration.

The second attempt ("Modifying Relative Delay Operator $\sigma_{\mathsf{rel}}^{r}$", Section 3.6.3) was dropped in favor of the third semantics ("Testing for future Inconsistency", Section 3.6.4) which treats both actions and delays before an inconsistent state uniformly.

Next, we examine the options of extending $BPA_{\perp}^{srt}$ with conditional time determinism (Section 3.5) and the time deterministic $BPA_{\perp}^{srt}$ (of Section 3.6.4) with other operators of hybrid process algebra.

3.7 Extensions of $BPA_{\perp}^{srt}$

$BPA_{\perp}^{srt}$ can only describe a subset of processes expressible by Process Algebra for Hybrid Systems. Ofcourse, we are interested in analyzing the possibilities of extensions of $BPA_{\perp}^{srt}$ towards BPA_{hs}^{srt} and ACP_{hs}^{srt}. In this section we throw some light on our insights regarding this matter. The section is outlined as follows: We dedicate a separate section to discuss addition of the operator integration (Section 3.7.1); afterwards, we discuss extending $BPA_{\perp}^{srt}$ to represent hybrid processes. This section comprises of ideas and suggestions. The implementation of these ideas is left as future work.

3.7.1 Integration

The addition of integration to Basic Timed Process Algebra BPA^{srt} enables it to model processes that can perform actions at any instance in a time interval. Recognizing the importance of integration, we first consider adding integration to $BPA^{srt}_{\perp}$.

Integration provides for alternative composition over a set of alternatives that can be uncountable. Let F be a function from non-negative reals to processes in BPA^{srt}, then an integral of $F(u)$ over an interval U, represented by $\int_{u \in U} F(u)$, behaves like an alternative composition of all the process terms in the set $\{F(p) \mid p \in U\}$.

The set of transition rules and axioms for integration in BPA^{srt} [BM02a] are given in the Appendix C.

The semantics of integration is defined on the same principle as that of alternative composition. We discuss how our two proposals of $BPA^{srt}_{\perp}$ can be extended with integration below:

1. In our first proposal of $BPA^{srt}_{\perp}$, i.e. the proposal with conditional time determinism, the transition rules for integration are similar to the rules of integration given for the process algebra BPA^{srt} [BM02a]. The only difference is that here (in integration in $BPA^{srt}_{\perp}$), we add consistency checking of all process terms constituting an integral expression.

   Thus adding integration to the first proposal is straightforward. The Rules for Integration are given in Table 3.15.

Table 3.15: Proposal 1 Rules for Integration ($a \in A, p, q, \geq 0, r > 0$)

$$\frac{\langle F(p)\rangle \xrightarrow{a} \langle x'\rangle, \{\langle \texttt{consistent}\ F(q)\rangle \mid q \in U\}}{\langle \int_{u \in U} F(u)\rangle \xrightarrow{a} \langle x'\rangle}\quad p \in U \quad \text{P1-}\underline{27}$$

$$\frac{\langle F(p)\rangle \xrightarrow{a} \langle \surd\rangle, \{\langle \texttt{consistent}\ F(q)\rangle \mid q \in U\}}{\langle \int_{u \in U} F(u)\rangle \xrightarrow{a} \langle \surd\rangle}\quad p \in U \quad \text{P1-}\underline{28}$$

$$\frac{\begin{array}{l} \{\langle F(q)\rangle \overset{r}{\longmapsto} \langle F_1(q)\rangle \mid q \in U_1\}, \\ \vdots \\ \{\langle F(q)\rangle \overset{r}{\longmapsto} \langle F_n(q)\rangle \mid q \in U_n\}, \\ \{\langle F(q)\rangle \overset{r}{\not\longmapsto}, \langle \texttt{consistent}\ F(q)\rangle \\ \mid q \in U_{n+1}\} \end{array}}{\langle \int_{u \in U} F(u)\rangle \overset{r}{\longmapsto} \langle \int_{u \in U_1} F_1(u) + \ldots + \int_{u \in U_n} F_n(u)\rangle}\quad \begin{array}{l} \{U_1, \ldots U_n\} \\ \text{partition of } U \backslash U_{n+1}, \\ \text{and } U_{n+1} \subset U \end{array} \quad \text{P1-}\underline{29}$$

$$\frac{\{\langle \texttt{consistent}\ F(q)\rangle \mid q \in U\}}{\langle \texttt{consistent}\ \int_{u \in U}(F(u))\rangle}\quad \text{P1-}\underline{30}$$

With the integration available, we can derive interesting equalities concerning the non-existence process constant and relative delay operator. In the semantics of our first proposal of $BPA^{srt}_{\bot}$ with integration, the following equality holds:

$$\sigma^{t}_{\mathsf{rel}}(\bot) = \int_{u<t}\sigma^{u}_{\mathsf{rel}}(\tilde{\tilde{\delta}}) \tag{3.10}$$

Equation 3.10 states that it is not possible to enter into inconsistency after a delay. A process term $\sigma^{t}_{\mathsf{rel}}(\bot)$ deadlocks at some point before time instance t.

2. Adding integration to our second proposal turns out to be more complex.

   In the second proposal for $BPA^{srt}_{\bot}$, a future inconsistency relation has been added in the semantics. A future inconsistency predicate with duration t for a process x, represents that if allowed to delay x would enter into inconsistency after time t. A process term $\sigma^{t}_{\mathsf{rel}}(\bot)$ has a future inconsistency predicate with duration t. It is written as follows:

   $$\sigma^{t}_{\mathsf{rel}}(\bot) \overset{t}{\longmapsto}_{\bot}$$

   Integration rules can be defined for the second proposal of $BPA^{srt}_{\bot}$ in the same way as the rules for the alternative composition. Then, in the second proposal an integral expression $\int_{u\in U}F(u)$ would be allowed to delay for a duration r only if none of the process terms $\{F(q) \mid q \in U\}$ have a future inconsistency predicate of duration shorter than or equal to r (See Rules P2-25 and Rules P2-26 in Table 3.13). Similarly, the term $\int_{u\in U}F(u)$ would have a future inconsistency predicate that has the shortest duration among future inconsistency predicates for all constituent terms $\{F(q) \mid q \in U\}$ (See Rules P2-28 and Rules P2-29 in Table 3.13). This straightforward extension of our second proposal for $BPA^{srt}_{\bot}$ does not give a satisfactory result. We explain it below:

   Consider the following equation:

   $$\int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot) = \tilde{\tilde{\delta}} \tag{3.11}$$

   Equation 3.11 holds in the second proposal of $BPA^{srt}_{\bot}$. Consider the set of process terms $\{\sigma^{t}_{\mathsf{rel}}(\bot) \mid t > 0\}$ that constitute the integral $\int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot)$. A time transition with any duration for $\int_{t>0}(\sigma^{t}_{\mathsf{rel}}(\bot))$ cannot be derived as with the shortest possible delay, a future inconsistency predicate for one of the process terms in the set $\{\sigma^{t}_{\mathsf{rel}}(\bot) \mid t > 0\}$ holds. On the other hand, a future inconsistency predicate for $\int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot)$ cannot be derived, as that requires finding the the smallest real number greater than zero which does not exist. This sets the process term $\int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot)$ bisimilar to immediate deadlock.

   On close investigation of our equational system we find out that this poses a problem. Equation 3.11, together with axiom SRT3, NE1 and some standard axioms of integration allows the following derivation:

   $$\begin{array}{lcll} \int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot) & = & \int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot + x) & \text{By NE1} \\ & = & \int_{t>0}(\sigma^{t}_{\mathsf{rel}}(\bot) + \sigma^{t}_{\mathsf{rel}}(x)) & \text{By SRT3} \\ & = & \int_{t>0}\sigma^{t}_{\mathsf{rel}}(\bot) + \int_{t>0}\sigma^{t}_{\mathsf{rel}}(x) & \text{By INT11} \\ & = & \tilde{\tilde{\delta}} + \int_{t>0}\sigma^{t}_{\mathsf{rel}}(x) & \text{By Equation 3.11} \\ & = & \int_{t>0}\sigma^{t}_{\mathsf{rel}}(x) & \text{By A6SR} \end{array}$$

The process term $\int_{t>0}\sigma^t_{\mathsf{rel}}(\perp)$ can be proven equal to $\int_{t>0}\sigma^t_{\mathsf{rel}}(x)$, with x being any process term. So, when the semantics of Section 3.6.4 is straightforwardly extended with integration, the time determinism axiom leads to unsound derivations once more. This cannot be allowed. Hence, the semantics must differentiate between $\int_{t>0}\sigma^t_{\mathsf{rel}}(\perp)$ and immediate deadlock $\tilde{\tilde{\delta}}$. [1]

A solution can be to add an extra relation to the semantics once more, that holds for process term $\int_{t>0}\sigma^t_{\mathsf{rel}}(\perp)$. A unary relation $\twoheadrightarrow_{\perp} \subseteq P$ that includes processes that enter into inconsistency with a delay of shortest possible duration, i.e. "immediately after now."

We agree that the resulting semantics will be complex with six relations. The relations Future inconsistency and Consistency can actually be combined. The Consistency relation can be removed from the semantics. Instead, a future inconsistency predicate of duration 0 holds for a process term x, whenever $\neg\langle \texttt{consistent}\ x\rangle$ was holding. For example,

$$\begin{gathered}
\perp \overset{0}{\longmapsto}_{\perp} \\
\sigma^0_{\mathsf{rel}}(\perp) \overset{0}{\longmapsto}_{\perp} \\
\perp \cdot x \overset{0}{\longmapsto}_{\perp} \\
\perp + x \overset{0}{\longmapsto}_{\perp}
\end{gathered}$$

Extending the second proposal of $BPA^{srt}_{\perp}$ with integration in this way is left as future work.

We extended our first proposal for $BPA^{srt}_{\perp}$ with integration defined by rules given in Table 3.15. We expect that the axioms (excluding the time determinism axiom INT10SR) that hold for BPA^{srt} with integration (see Appendix C) also hold for our first proposal of $BPA^{srt}_{\perp}$. Formulating and proving axioms of integration is left as future work.

While considering the proposal of Section 3.6.4 (which constitutes our second proposal of $BPA^{srt}_{\perp}$), we find that extending it with integration is not that simple. We propose a solution for adding integration to it and leave it as future work.

3.7.2 A Hybrid Process Algebra-Adding Flow Determinism

A Process Algebra for Hybrid Systems is inherently more complex than a timed process algebra. Hence an extension of $BPA^{srt}_{\perp}$ to a basic hybrid process algebra requires a thorough research. Below, we present our views on extending the two proposals of $BPA^{srt}_{\perp}$ to a hybrid setting:

1. An extension of our first proposal of $BPA^{srt}_{\perp}$ to a hybrid process algebra is quite obvious. The signature of $BPA^{srt}_{\perp}$ is extended with extra operators i.e. conditional operator, signal emission operator , signal evolution operator and signal transition operator defined in [BM05]. The semantics is also extended with details like valuations of model

[1] Note that this problem will also arise in the semantics described in Section 3.6.3. Adding the integration on the same lines as the rules for alternative composition, requires that during a delay for $\int_{t>0}\sigma^t_{\mathsf{rel}}(\perp)$, none of the members of the set $\{\sigma^t_{\mathsf{rel}}(\perp) \mid t > 0\}$ become inconsistent. For this we need to know the smallest number greater than zero.

variables and their trajectories during delays that are necessary to describe hybrid behaviour of processes. New rules are added to the semantic to define the behaviour of new operators.

Here, we need a careful approach, because with the extended signature, comes a chance of repeating the mistakes of Process Algebra for Hybrid Systems. As mentioned before, in Process Algebra for Hybrid Systems, alternative composition is not associative.

This error can be corrected by modifying the semantics of alternative composition, so that too much emphasis on duration of delays is replaced by an equal emphasis on durations and trajectories of model variables during delays.

The delay behaviour of alternative composition as it is **currently** defined in Process Algebra for Hybrid Systems is narrated below:

Consider an alternative composition $p + q$.

> If p and q can delay together for a given duration of delay with the same trajectory of variables, then $p + q$ delays so that at the end of delay, choice is retained.
>
> The process term $p + q$ can also delay for a given duration and proceed as one of the process terms p or q only if the process term left behind cannot delay for the same duration with **any** trajectory of variables.

The latter behaviour should be modified as follows:

> The process term $p + q$ can also delay for a given duration and proceed as one of the process terms p or q only if the process term left behind cannot delay for the same duration with the **same** trajectory of variables.

Consider the transition rules HS-12 and HS-13, describing the delay behaviour of an alternative composition in Appendix B, Table B.1. These rules will be replaced by the following two rules:

$$\frac{\langle x, \alpha \rangle \xmapsto{r,\rho} \langle x', \alpha' \rangle, \langle y, \alpha \rangle \not\xmapsto{r,\rho}, \alpha \in [\mathsf{s}(y)]}{\langle x + y, \alpha \rangle \xmapsto{r,\rho} \langle x', \alpha' \rangle}$$

$$\frac{\langle y, \alpha \rangle \xmapsto{r,\rho} \langle y', \alpha' \rangle, \alpha \in [\mathsf{s}(x)], \langle x, \alpha \rangle \not\xmapsto{r,\rho}}{\langle x + y, \alpha \rangle \xmapsto{r,\rho} \langle y', \alpha' \rangle}$$

Note that the negative formulaes $\langle y, \alpha \rangle \not\xmapsto{r}$ and $\langle y, \alpha \rangle \not\xmapsto{r}$ in rules 12 and 13 have been replaced by $\langle y, \alpha \rangle \not\xmapsto{r,\rho}$ and $\langle x, \alpha \rangle \not\xmapsto{r,\rho}$ here. A predicate $\langle x, \alpha \rangle \not\xmapsto{r,\rho}$ represents that there does not exists a x', such that

$$\langle x, \alpha \rangle \xmapsto{r,\rho} \langle x', \alpha_r^\rho \rangle$$

is derivable.

With this modification, the alternative composition will remain associative with the new operators added to $BPA_\perp^{srt}$. This definition of alternative composition models what we call *flow determinism* (see Section 2.3, chapter 2). By Flow determinism, we mean that unique flows (variable trajectories) have unique targets.

With regards to time determinism, we expect the following axiom to hold:

$$\sigma^r_{\mathsf{rel}}(x) + \sigma^r_{\mathsf{rel}}(y) = \sigma^r_{\mathsf{rel}}(x+y) + \sigma^r_{\mathsf{rel}}(\neg s_\rho(y) \mathbin{\nearrow} x) + \sigma^r_{\mathsf{rel}}(\neg s_\rho(x) \mathbin{\nearrow} y)$$

where $r > 0$.

The operator s_ρ defined in [BM03] returns the signal emitted by a process.

We expect that some axioms that are currently not sound (for example, HSE7, INT11, HSINT7) will hold with the new definition of alternative composition. On the other hand some of the unsound axioms with emphasis on time determinism (for example HSE13, HSSCRM given below) will be dropped.

(HSE13):
$\phi \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(x) + \phi' \mathbin{\curvearrowright}_{V'} \sigma^r_{\mathsf{rel}}(\nu(y)) = \phi \mathbin{\curvearrowright}_V (\sigma^r_{\mathsf{rel}}(x) + \phi' \mathbin{\curvearrowright}_{V'} \sigma^r_{\mathsf{rel}}(\nu(y)))$

(HSSRCM):
$\sigma^r_{\mathsf{rel}}(x) \mathbin{\lfloor\!\lfloor} (\phi \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(y) + z) = \sigma^r_{\mathsf{rel}}(x) \mathbin{\lfloor\!\lfloor} (\sigma^r_{\mathsf{rel}}(\phi \mathbin{\curvearrowright}_V y) + z) + \phi \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(\tilde{\tilde{\delta}})$

We find that an axiom reflecting flow determinism (that should replace HSE13) poses a problem. Consider a BPA^{srt}_{hs} process term $\phi \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(x) + \psi \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(y)$. It behaves as one of the process terms, $(\phi \wedge \psi) \mathbin{\curvearrowright}_{V \cup V'} \sigma^r_{\mathsf{rel}}(x+y)$, $\psi' \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(x)$ or $\psi'' \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(y)$, where the proposition ψ' describes a signal evolution that satisfies the proposition ϕ but during the delay of x violates ψ at least once. Similarly, ψ'' describes an evolution that satisfies the proposition ψ but during the delay of y violates ϕ at least once. In the current syntax of ACP^{srt}_{hs}, where propositions can only be algebraic or differential (in)equalities, the conditions ψ' and ψ'' cannot be expressed.

The properties ψ' and ψ'' express the "eventually true" conditions in modal logic. Another problem we face while axiomatizing flow deterministic behaviour is that the evolution operator $\curvearrowright$ is not appropriate to use with propositions expressing "eventually true" properties. An evolution operator persists over time. This is represented by the following equation:

$$\phi \mathbin{\curvearrowright}_V \sigma^r_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) = \phi \mathbin{\curvearrowright}_V (\sigma^r_{\mathsf{rel}}(\phi \mathbin{\curvearrowright}_V \sigma^u_{\mathsf{rel}}(x))) \tag{3.12}$$

If ϕ expresses an "eventually true" then Equation 3.12 would express that the condition must happen twice during the delay. Therefore to axiomatize the flow deterministic choice, we need another evolution operator that only holds for the first time transition and does not persist over time.

2. The semantics of the second proposal for $BPA^{srt}_{\perp}$ is more elaborate than that of the first proposal. Extending it to a hybrid setting also turns out to be more involved than the first.

   In addition to the relations of first proposal, the semantics of second proposal contains a future inconsistency relation. When we extend this proposal with the operators for hybrid processes, then for each operator we need to define rules for deriving future inconsistency predicates ($\mapsto_\perp$) and inconsistency immediately after now predicates ($\twoheadrightarrow_\perp$).

This needs further research. An idea is that inconsistency is absence of a solution to a given set of constraints on variable trajectories. We cannot say more at this point.

Studying the possibilities of extensions of $BPA_{\perp}^{srt}$ to a hybrid process algebra, we find at this point that extending $BPA_{\perp}^{srt}$ with conditional time determinism with operators of BPA_{hs}^{srt} is much simpler than the time deterministic $BPA_{\perp}^{srt}$.

3.8 Conclusions

This chapter considers Process Algebra for Hybrid Systems [BM05] which is a well-known formalism for specification of hybrid systems. Recently, a number of errors have been uncovered in this algebra. This chapter is an initial development towards correcting these errors.

Process Algebra for Hybrid Systems, denoted by ACP_{hs}^{srt}, has a hierarchical structure. It is built from the most basic algebra BPA [BW90] in several layers to a process algebra ACP_{hs}^{srt} for description of hybrid systems. The hierarchical structure of ACP_{hs}^{srt} is shown in the figure 3.1.

This chapter is confined to the discussion of Process Algebra for Hybrid Systems without parallelism i.e. BPA_{hs}^{srt}.

As shown in Figure 3.1, the errors found in ACP_{hs}^{srt} appear at two levels. First at the level of $BPA_{\perp}^{srt}$ and secondly at the level of BPA_{ps}^{srt}. The error in algebra $BPA_{\perp}^{srt}$ is the unsoundness of axiom of Time Determinism ($SRT3$). The error in BPA_{ps}^{srt} and its derived theories (BPA_{hs}^{srt},ACP_{hs}^{srt}, etc.) is that the Choice is non-associative and related to this error, a number of other axioms turn out to be unsound.

As a first step towards making corrections in BPA_{hs}^{srt}, we present in this chapter two proposals for correcting the algebra $BPA_{\perp}^{srt}$.

In our first proposal, we replace axiom of Time Determinism (SRT3) by a conditional axiom (SRTD). The conditional axiom represents time determinism in cases when the target process terms are not bisimilar to Non-existence. Also a new axiom ($SRTD\perp$) is added to $BPA_{\perp}^{srt}$ [BM05], that reflects that passage of time makes choices in the presence of Non-existence process. These axioms ($SRTD$, $SRTD\perp$) also hold in the semantics of [BM05]. In fact we show that for all $BPA_{\perp}^{srt}$ processes, our first proposal is equivalent to the process algebra BPA_{hs}^{srt}.

Due to the importance of Time Determinism, we also search for a variant of $BPA_{\perp}^{srt}$, where the axiom SRT3 holds for all process terms including the Non-existence process. Finding a time deterministic $BPA_{\perp}^{srt}$ turns out to be non-trivial. To achieve our goal, we try modifying the semantics of the operators reasoned about in SRT3, i.e. alternative composition and the relative delay operator. Finally, we adopt an approach that we call "Testing for Future Inconsistency" (Section 3.6.4) for our second proposal of $BPA_{\perp}^{srt}$. The axiom of time determinism SRT3 holds in this proposal.

Both our first and second proposals of $BPA_{\perp}^{srt}$ are conservative ground-extensions of BPA^{srt} and BPA.

Lastly, we consider extensions of $BPA_{\perp}^{srt}$ with the operators from [BM05] to describe hybrid behaviour of processes.

It appears that extending the first proposal is simple. When extending the signature of $BPA_{\perp}^{srt}$ with operators of BPA_{hs}^{srt}, we need to modify the alternative composition of [BM05] so that it is associative. Nevertheless, axiomatization of flow deterministic ACP_{hs}^{srt} still poses a problem as mentioned in Section 3.7. It is left as future work.

Extending the second proposal to a hybrid process algebra needs more research. For the second proposal, we need to determine what is the meaning of "Future Inconsistency" ($\mapsto_\perp$) and "Inconsistency immediately after now" ($\twoheadrightarrow_\perp$) in a hybrid environment.

In our work on Process Algebra for Hybrid Systems, tracing the error of time determinism back to $BPA_\perp^{srt}$ was a major step. After isolating the error, the rest of the road map was evident. The reason for presenting both the conditional time determinism approach and the general time determinism approach for $BPA_\perp^{srt}$ is to clarify the choices available in each case. An interested researcher can then make his own comparison and decide according to the problem at hand. The idea of "Testing for Future Inconsistency" which constitutes our second proposal is new and the research on it is in progress. There exist better possibilities for the implementation of the idea (one of them proposed by Jos Baeten) than given in this chapter. Realizing these suggestions is left as future work.

Chapter 4

Linearization of Hybrid χ

The process algebra Hybrid χ was introduced in Chapter 2 as a process algebra for specification and verification of hybrid systems. A number of tools for simulations and analysis of specifications written in Hybrid χ are available (see [The06] and the Hybrid χ website [Chi]). This chapter describes a new method of linearizing Hybrid χ process terms. By linearization, we mean the procedure of rewriting a process term into a linear form. A linear form consists of only basic operators of a process language such as actions, choice and sequential composition. In particular a linear form does not include a parallel operator.

As mentioned in Chapter 2, now Hybrid χ 2.0 is available [BHR$^+$08]. During the commencement of the research presented in this chapter, see [KBC07], Hybrid χ 2.0 was being developed. Hence, this chapter gives a linearization algorithm for an early version of Hybrid χ 2.0. After the completion of our research on linearization of Hybrid χ processes in [KBC07], some more changes were made to the syntax and semantics of Hybrid χ 2.0. Nevertheless, the main constructs of the process algebra remain the same. Therefore, we think that the reasoning presented here for linearization for Hybrid χ processes will be of utility for linearizing process terms of Hybrid χ 2.0. Updating the present linearization algorithm for Hybrid χ 2.0 is discussed in the conclusion section of this chapter.

This chapter is structured as follows: First of all, in Section 4.1, we explain the process of linearization, a brief history of linearization in process algebra and its specific challenges. In order to make the reader more familiar with our goal without yet going into formal details, we present the result of linearizing a Hybrid χ process in Section 4.2. In Section 4.3, we define the set of input process terms to our linearization algorithm. In Section 4.4, we define the syntax of the linear form. Section 4.5 informally gives a visualization of the linear form. The purpose of this visualization is to help in understanding the essentials of the linearization procedure. Section 4.6 inductively defines the linearization of a Hybrid χ specification by giving a linearization algorithm for atomic constructs of Hybrid χ and for all the operators allowed in an input process term. In Section 4.7, we argue about the complexity of our linearization algorithm. In Section 4.8, we discuss the benefits and shortcomings of the algorithm and its position among other linearizations [Use02, BRC06]. We also discuss updating our linearization algorithm for Hybrid χ 2.0 [BHR$^+$08].

4.1 Linearization

Linearization is similar to *elimination* found in many ACP style process algebras such as [BW90, BM02a, BM05, CR05, BMR$^+$06]. In these process algebras we find elimination theorems that state that any process specification in a given process algebra can be rewritten into a simpler form, called a basic term. Each of these process algebras also contains a set of basic terms into which all closed terms of that process algebra can be rewritten. A basic term consists of only atomic actions, basic operators of the given process algebra (like choice and sequential composition) and guarded tail recursion. Elimination theorems are very useful in proving properties about closed terms of a process algebra as with these theorems proofs by structural induction become smaller. An important property of a basic term is that it does not contain parallelism. Hence, the term elimination is often used for elimination of a parallel operator from a process term. In this chapter the words *linear term* and *basic term*, *linearization* and *elimination* are used interchangeably.

4.1.1 History

Historically, μCRL [Use02], a process algebra with data, first used the term “linear process equation” (LPE) for its basic terms, and referred to the procedure of rewriting a process specification into a basic term as *linearization.* The terms *linear* and *linearization* refer to the fact that a linear process equation resembles a right linear data parameterized grammar [GPU01]. A linear process equation or an LPE is a subclass of recursive process specifications. A linear process equation defines a complete system specification in a single recursive equation. A linear process equation (LPE) in μCRL has the following form:

$$\begin{aligned} X(d:D) = \; & \textstyle\sum_{i\in I}\sum_{e\in E_i} a_i \cdot X(g_i(d,e)) \lhd c_i(d,e) \rhd \delta \\ & + \textstyle\sum_{j\in J}\sum_{e\in E_j} a_j \lhd c_j(d,e) \rhd \delta \end{aligned}$$

where I and J are disjoint finite sets of indexes and d denotes a state vector. Normally we are interested in a solution of the LPE in a particular initial state d_0. The equation is explained as follows. The process X being in a state d can, for any $e \in E_i$ that satisfies the condition $c_i(d,e)$, perform an action a_i, and then proceed to the state $g_i(d,e)$. Moreover, it can, for any $e \in E_j$ that satisfies the condition $c_j(d,e)$, perform an action a_j, and then terminate successfully. For a comprehensive definition of an LPE, please refer to [Use02]. The format of an LPE is limited to basic operators of μCRL, a single recursion variable and guarded tail recursion. Depending upon the value of the parameter d and the guard conditions (i.e. $c_i(d,e)$ and $c_j(d,e)$) different options of an LPE are activated.

Advantages of rewriting a specification into a linear process equation are that many tools and techniques for analysis and verification of specifications need to operate only on linear terms [Wou01]. Linearization / elimination in process algebra, can be compared to flattening in state charts [Was04] and in automata theory [Alu07].

Following the work on μCRL, linearization algorithms and tools for hybrid process algebra [BRC06] and Hybrid χ [The06] have also been developed. In [BRC06], a similar approach to that of μCRL has been adopted and the final linear form is a linear process equation. For Hybrid χ [The06], the final linear form contains a set of linear recursive equations.

4.1.2 Challenges

A concern in linearization is the size of the resulting linear term. When operators are removed from a specification its size may increase so much that it becomes impossible to automatically linearize a specification of a large system [Use02, BRC06]. Techniques like symbolic reasoning on data variables and variable abstraction [Use02, BRC06] have been developed to reduce the size of the resulting linear term in linearization. In the process of linearization, a parallel composition operator is eliminated from a specification. A parallel composition operator represents the result of simultaneous execution of two processes. Its semantics includes details such as synchronization, communication and interleaving of actions of the process terms executing in parallel. A linear form of the specification of a multi-component system with components running in parallel models the behaviour of the system using only basic operators. Hence the size of a linear form of such a system specification could be very large. In [Use02, BRC06], stack-like data structures are used to model interleaving in the linear form of a parallel composition. It has been pointed out in [Use02] that in cases where process variables are not parameterized by data, a counter with values in natural numbers can also serve the same purpose. In this chapter, we give an algorithm for linearization of Hybrid χ specifications using such counters. We call these counters *program counters.*

4.2 An Example

In the Hybrid χ language, a program counter is a new discrete variable defined locally in the linear form of a process specification. Different values of a program counter activate different atomic constructs of the specification. Using program counters, a Hybrid χ process term is linearized as shown in the following example:

Consider a parallel composition, $(a;\ b \parallel d;\ e;\ f)$, where a, b, c, d, e, f are non-communicating atomic actions. The symbol ; denotes sequential composition in Hybrid χ. Eliminating the parallel operator from $(a;\ b \parallel d;\ e;\ f)$ results in the following linear form:

Let R denote the linear form of $(a;\ b \parallel d;\ e;\ f)$. Then,

$$\begin{array}{rl} R = & a;\ (b;\ d;\ e;\ f \mathbin{[\!]} d;\ (b;\ e;\ f \mathbin{[\!]} e;\ (f;\ b \mathbin{[\!]} b;\ f))) \\ & \mathbin{[\!]} d;\ (a;\ (b;\ e;\ f \mathbin{[\!]} e;\ (f;\ b \mathbin{[\!]} b;\ f))) \end{array}$$

The symbol $[\!]$ denotes choice or alternative composition in Hybrid χ.

Using program counters, the linear form $\widetilde{R}$ of $(a;\ b \parallel d;\ e;\ f)$ is modelled as follows:

$$\begin{array}{rllll} \widetilde{R} = & \llbracket_{\mathrm{V}} \{i_1 \mapsto \bot, i_2 \mapsto \bot\}, \emptyset, \emptyset & & & \\ & ::\ \llbracket_{\mathrm{R}}\ \{X \ \mapsto & (i_2 = 4) & \rightarrow & a, i_2 := 2;\ X \\ & & \mathbin{[\!]}\ (i_2 = 2) \wedge \neg \mathrm{Odd}(\{i_1\}) & \rightarrow & b, i_2 := 1;\ X \\ & & \mathbin{[\!]}\ (i_2 = 2) \wedge \mathrm{Odd}(\{i_1\} & \rightarrow & b, i_2 := 0, i_1 := 0 \\ & & \mathbin{[\!]}\ (i_1 = 6) & \rightarrow & d, i_1 := 4;\ X \\ & & \mathbin{[\!]}\ (i_1 = 4) & \rightarrow & e, i_1 := 2;\ X \\ & & \mathbin{[\!]}\ (i_1 = 2) \wedge \neg \mathrm{Odd}(\{i_2\}) & \rightarrow & f, i_1 := 1;\ X \\ & & \mathbin{[\!]}\ (i_1 = 2) \wedge \mathrm{Odd}(\{i_2\}) & \rightarrow & f, i_1 := 0, i_2 := 0\} \\ & \quad ::\ (i_1 = 6 \wedge i_2 = 4) \curvearrowright X & & & \\ & \quad \rrbracket & & & \\ & & & & \\ & \rrbracket & & & \end{array}$$

where,

$$\begin{array}{ll} [\![_{\mathrm{V}} & \ldots \\ & \ldots \\ ]\!] & \end{array}$$

represents a variable scope in which two local discrete data variables i_1 and i_2 are declared.

Similarly,

$$\begin{array}{ll} [\![_{\mathrm{R}} & \ldots \\ & \ldots \\ ]\!] & \end{array}$$

represents a recursion scope in which a new recursion variable X is declared. The process definition of X is a linear process term that defines the behaviour of $(a;\ b \parallel d;\ e;\ f)$. The data variables, i_1 and i_2 are the program counters used in the linear process equation. The variables i_1, i_2 and X are not part of the original specification. To make them unobservable to an outside observer, they are declared to be local variables.

An alternative of the process definition of X is explained as follows:

$$(i_2 = 4) \to a, i_2 := 2;\ X$$

The predicate $i_2 = 4$ is the condition guarding the given alternative, a is an action and $i_2 := 2$ is an assignment updating the value of i_2. If $(i_2 = 2)$ evaluates to true, the action a can be performed. After the action, the program counter i_2 can be set to 2 and the recursion variable X is called again.

Before, the first call to recursion variable X in process term $(i_1 = 6 \wedge i_2 = 4) \curvearrowright X$, the initialization operator $\curvearrowright$ sets the values of i_1 and i_2 to 6 and 4 respectively. In the definition of recursion variable X, depending upon the action performed, the value of one of the program counters i_1 or i_2 is updated. A requirement of the linearization is that the resulting linear form must be bisimilar to the original process term. In the linearization of a process term with parallel composition, a different program counter for each component of parallel composition is used. The program counter of each component can be updated independently of the other program counters. For example, during the execution of X, the program counter i_1 can have value 2 and the program counter i_2 can have value 4, indicating that only actions d and e have been executed so far. By independently updating the two program counters, all possible interleaving of actions of parallel components are modelled and we do not need to explicitly include these interleavings in the linear form. In this way, the size of the linear form of a parallel composition is approximately of the order of the product of the sizes of its components. Another advantage of doing linearization this way is that parallel components can still be recognized in the linear form.

4.3 Input to the algorithm

This section presents the set of process terms for an early version of Hybrid χ 2.0. We impose some restrictions on the set of process terms $\mathcal{P}_{\mathrm{s}}$ which are allowed as input to the linearization algorithm. The BNF definition given below defines the set of process terms $\mathcal{P}_{\mathrm{s}}$, with $p_{\mathrm{s}} \in \mathcal{P}_{\mathrm{s}}$, that can be linearized by our linearization algorithm.

$$
\begin{array}{rcll}
p_{\mathrm{s}} & ::= & p_{\mathrm{atom}} & \text{atomic actions} \\
& \mid & p_{\mathrm{u}} & \text{invariant and} \\
& & & \text{urgency conditions} \\
& \mid & u \curvearrowright p_{\mathrm{s}} & \text{initialization} \\
& \mid & p_{\mathrm{s}} \,;\, p_{\mathrm{s}} & \text{sequential composition} \\
& \mid & p_{\mathrm{s}} \mathbin{[\,]} p_{\mathrm{s}} & \text{alternative composition} \\
& \mid & p_{\mathrm{s}} \parallel p_s & \text{parallel composition} \\
& \mid & \partial_A(p_{\mathrm{s}}) & \text{encapsulation} \\
& \mid & \partial_H(p_{\mathrm{s}}) & \text{send and receive} \\
& & & \text{action encapsulation} \\
& & & \\
& \mid & \upsilon_H(p_{\mathrm{s}}) & \text{urgent channel communication} \\
& \mid & |[_{\mathrm{H}}\, H \;::\, p_{\mathrm{s}}\,]| & \text{channel scope } |[_{\mathrm{H}}\,]| \\
& \mid & p_{\mathrm{R}} & \text{restricted use of recursion} \\
& \mid & |[_{\mathrm{V}}\, \sigma_{\perp}, C, L :: p_{\mathrm{s}}\,]| & \text{variable scope } |[_{\mathrm{V}}\,]|
\end{array}
$$

where u is a predicate on a set of model variables $\mathcal{V}$, $A \in A_{\mathrm{label}}$ is a set of labels of non-communicating actions and H is a set of channels. In $|[_{\mathrm{V}}\, \sigma_{\perp}, C, L :: p_{\mathrm{s}}\,]|$, C is a set of local continuous variables, L is a set of local algebraic variables and $\sigma_{\perp}$ is a valuation of local variables.

The set $\mathcal{P}_{\mathrm{atom}}$ consists of atomic actions. An atomic action process term $p_{\mathrm{atom}} \in \mathcal{P}_{\mathrm{atom}}$ is defined below:

$$
\begin{array}{rcll}
p_{\mathrm{atom}} & ::= & l_{\mathrm{a}}, W : r & \text{action process term} \\
& \mid & h\,!\,\mathbf{e}_n, W : r & \text{send process term} \\
& \mid & h\,?\,\mathbf{x}_n, W : r & \text{receive process term} \\
& \mid & h\,!?\,\mathbf{x}_n := \mathbf{e}_n, W : r & \text{communication process term}
\end{array}
$$

An atomic action consists of an action label, a set of non-jumping model variables W and a predicate r. We restrict the syntax of predicates on model variables to the set $\mathcal{R}$ of predicates, defined as follows:

Let $r \in \mathcal{R}$.

$$
\begin{array}{rcl}
r & ::= & \text{true} \\
& \mid & \text{false} \\
& \mid & x^{+} \ \mathrm{op}_{\mathrm{r}} \ c \\
& \mid & x^{-} \ \mathrm{op}_{\mathrm{r}} \ c \\
& \mid & r \wedge r
\end{array}
$$

where x is a model variable, x^{-} denotes the values of variable x before an action, x^{+} denotes the values of x after an action, c is any value in the set Λ (the set of all possible values of data variables) and op_{r} is the set of relational operators, i.e.

$$\mathrm{op}_{\mathrm{r}} = \{<, >, =, \geq, \leq\}$$

The action labels can be labels of communication actions (as explained next), or labels of non-communicating actions such as a, b, c. By means of a communication, values are communicated from one process to the other, i.e. synchronization of send action $h\,!\,\mathbf{e}_n$ and receive action $h\,?\,\mathbf{x}_n$ yields communication $h\,!?\,\mathbf{x}_n := \mathbf{e}_n$ by which data $\mathbf{e}_n$ is transferred from the sender to the receiver. The notation $\mathbf{e}_n$ denotes a vector of expressions whose values are sent and $\mathbf{x}_n$ is a vector of variables in which the received values are stored.

The set $\mathcal{P}_\mathrm{u}$ consists of invariants and urgency conditions.

$$\begin{array}{rcl} p_\mathrm{u} & ::= & \mathrm{inv}\ u \\ & \mid & \mathrm{urg}\ u \end{array}$$

A recursion scope operator (denoted by $[\![_\mathrm{R} \ldots]\!]$) is allowed with a restriction that no recursion definition of a recursion variable defined within the scope $[\![_\mathrm{R} \ldots]\!]$, refers to a recursion variable defined outside the scope. The syntax of the process terms defining a recursion variable is also restricted. In a process definition, only tail recursion is allowed and an occurrence of a recursion variable must be guarded.

The restricted recursion scope operator process term is defined by:

$$\begin{array}{rcll} p_\mathrm{R} & ::= & [\![_\mathrm{R}\, R :: X_i\,]\!] & \mathrm{Complete}(R) \wedge X_i \in \mathrm{dom}\,(R) \\ & \mid & [\![_\mathrm{R}\, R :: p\,]\!] & \mathrm{Complete}(R) \wedge \mathrm{Recvars}(p) \in \mathrm{dom}\,(R) \end{array}$$

where,

1. $i \in \mathbb{N}^{>0}$;

2. $R \in \mathcal{R}$, where $\mathcal{R} : \mathcal{X} \mapsto \mathcal{P}$ is the set of all functions from recursion variables to process terms. R is known as a recursion definition. Syntactically, a recursion definition is denoted by a set of pairs $\{X_1 \mapsto p_1, \ldots, X_m \mapsto p_m\}$, where X_i denotes a recursion variable and p_i denotes a process term defining X_i;

3. The set $\mathcal{P}$ of process terms includes the following process terms:

$$\begin{array}{rcl} p & ::= & p_\mathrm{s} \\ & \mid & p_\mathrm{s}; X_i \\ & \mid & p_\mathrm{s}; p \\ & \mid & p \,[\,]\, p \end{array}$$

4. Another restriction on the set of possible process definitions of recursion variables is as follows:

   Recursion variables with process definitions that declare a variable scope operator followed by self recursion are not allowed in the input to the algorithm. For example the following recursion definition is not allowed:

$$\{X_1 \mapsto [\![_\mathrm{V}\, \sigma_\perp, C, L :: p_\mathrm{s}\,]\!]\ ;\ X_1\}$$

   The reason for this restriction is explained in detail in sections 4.6.7 and 4.6.12.

5. The function Recvars : $\mathcal{P} \cup (\mathcal{X} \times \mathcal{R}) \to 2^{\mathcal{X}}$ takes a process term of the form p, or a recursion variable and a recursion definition. It returns the recursion variables present in the given process term or in the defining process term of the given recursion variable, respectively. We make sure that whenever the function Recvars is called with a recursion variable and a recursion definition, the recursion definition contains the definition of the

given recursion variable. Therefore, this check is not included in the function definition given below.

$$
\begin{array}{lcl}
\mathrm{Recvars}(p_{\mathrm{s}}) & = & \emptyset \\
\mathrm{Recvars}(p_{\mathrm{s}};\ X_i) & = & \{X_i\} \\
\mathrm{Recvars}(p_{\mathrm{s}};\ p) & = & \mathrm{Recvars}(p_{\mathrm{s}}) \cup \mathrm{Recvars}(p) \\
\mathrm{Recvars}(p \mathbin{[\!]} q) & = & \mathrm{Recvars}(p) \cup \mathrm{Recvars}(q) \\
\\
\mathrm{Recvars}(X_i, \{X_i \mapsto p\}) & = & \mathrm{Recvars}(p) \\
\mathrm{Recvars}(X_i, \{X_j \mapsto p\})_{j \neq i} & = & \emptyset \\
\mathrm{Recvars}(X_i, R \cup R') & = & \mathrm{Recvars}(X_i, R) \cup \mathrm{Recvars}(X_i, R')
\end{array}
$$

6. The function Complete : $\mathcal{R} \rightarrow \mathcal{B}\mathrm{ool}$ takes a recursion definition R. It collects the recursion variables that are mentioned in the defining process terms of all recursion variables in the domain of R. If the set thus obtained is a subset of the domain of R, then Complete returns true else it returns false. In other words, Complete checks if all recursion variables occurring in the range of R are also defined in R.

$$
\mathrm{Complete}(R) = \begin{array}{ll}
\mathrm{true} & \text{if } \bigcup_{X_i \in \mathrm{dom}(R)} \mathrm{Recvars}(X_i, R) \subseteq \mathrm{dom}(R) \\
\mathrm{false} & \text{otherwise}
\end{array}
$$

4.4 Output Form of the algorithm

The set of linearized process terms $\widetilde{\mathcal{P}}$, with $\widetilde{p} \in \widetilde{\mathcal{P}}$, is defined as:

$$
\begin{array}{lcl}
\widetilde{p} & ::= & [\![_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \sigma, C, L :: \\
& & \quad [\![_{\mathrm{R}}\ \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X\ ]\!] \\
& &]\!]
\end{array}
$$

We discuss the structure of the linear process term $\widetilde{p}$ in the following sections:

4.4.1 Variable scope operator and program counters

1. The set of program counters is defined as follows:

$$
\mathcal{I} = \{i_k \mid k \in \mathbb{N}^{>0}\}, \text{ such that } \mathcal{I} \cap \mathcal{V} = \emptyset
$$

2. The valuation $\sigma_{\mathrm{pc}} : \mathcal{I} \mapsto \{\bot\}$ is a partial function which is syntactically denoted as $\{i_1 \mapsto \bot, \ldots i_k \mapsto \bot\}$, k being the number of program counters used in the definition of a linear form. The valuation σ_{pc} declares the program counters used to describe a linear form as local discrete variables. These are distinct from all other local discrete, algebraic or continuous model variables.

3. The valuation $\sigma : \mathcal{V} \mapsto \{\bot\}$ is syntactically denoted as $\{x \mapsto \bot, y \mapsto \bot, \ldots, z \mapsto \bot\}$, where x, y, z are local discrete or continuous model variables other than program counters.

4. C is the set of local continuous variables and L is the set of local algebraic variables.

4.4.2 Recursion scope operator

The recursion scope operator $|[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X\ ]|$, where u_{pc} is a predicate over program counters and u is a predicate over model variables, defines a single recursion definition. The right hand side of the recursive definition is a linear process term.

The BNF definition of the set of process terms $\overline{\mathcal{P}}$, with $\overline{p} \in \overline{\mathcal{P}}$ is as follows:

$$\begin{array}{rcl} \overline{p} & ::= & b_{\mathrm{pc}} \to p_{\mathrm{u}} \\ & | & b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{update}(X_i);\ X \\ & | & b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X \\ & | & b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \\ & | & \overline{p} \,[\!]\, \overline{p} \end{array}$$

where X_i for any natural number i denotes a recursion variable. The notation update(X_i) occurs only in intermediate output forms as an intermediate result of linearizing a recursion scope operator. How we eliminate these pointers is given in Section 4.6.7.

The set of action process terms $\mathcal{P}_{\mathrm{act}}$, with $p_{\mathrm{act}} \in \mathcal{P}_{\mathrm{act}}$, and the set of action predicates $\mathcal{AP}$, with ap $\in \mathcal{AP}$, are defined as follows:

$$\begin{array}{rcl} p_{\mathrm{act}} & ::= & p_{\mathrm{atom}} \\ & | & p_{\mathrm{atom}}, \mathrm{ap} \\ \mathrm{ap} & ::= & W : r \\ & | & \mathrm{ap}, \mathrm{ap} \end{array}$$

where p_{atom} is an atomic action that has been defined in Section 4.3, W is a subset of model variables and $r \in \mathcal{R}$ is a jump predicate containing model variables.

In the BNF definition of a process term $\overline{p}$, the alternatives consisting of an urgency condition or an invariant are never followed by the recursion variable X, because an urgency condition and an invariant do not terminate. Some alternatives with action process terms are also not followed by a recursive call to X. These are the terminating actions. The process term $\overline{p}$ can terminate by executing one of the terminating actions. When $\overline{p}$ terminates, all program counters are set to zero.

The set of action predicates $\mathcal{AP}_{\mathrm{pc}}$ that update program counters, with $\mathrm{ap}_{\mathrm{pc}} \in \mathcal{AP}_{\mathrm{pc}}$ and the set of predicates on program counters $\mathcal{R}_{\mathrm{pc}}$, with $r_{\mathrm{pc}} \in \mathcal{R}_{pc}$, are defined as follows:

$$\begin{array}{rcl} \mathrm{ap}_{\mathrm{pc}} & ::= & W_{\mathrm{pc}} : r_{\mathrm{pc}} \\ & | & \mathrm{ap}_{\mathrm{pc}}, \mathrm{ap}_{\mathrm{pc}} \\ W_{\mathrm{pc}} & \subseteq & \mathcal{I} \\ r_{\mathrm{pc}} & ::= & \bigwedge_{i_k \in W_{\mathrm{pc}}} i_k = c_k \end{array}$$

where $c_k \in \mathbb{N}$. We use the convention that $\bigwedge_{i_k \in \emptyset} i_k = c_k$ evaluates to true.

4.4.3 Even and Odd values of program counters

Program counters in a linear form either have an odd or an even value. Even values are reserved for the so-called “active” program counters. Odd values are reserved for the so-called inactive program counters. This distinction between active and inactive program counters is needed to be able to properly deal with (partial) termination in parallel composition. In parallel composition, we need two concepts of termination: local termination of a component and global termination. Local termination refers to termination of a component of a parallel composition, when the other components of the composition have not yet terminated. On local termination, the program counters of the terminating component are set to an odd value. Global termination refers to the final termination of the parallel composition, that takes place when the last component of the parallel composition terminates. When performing a terminating action of a component of a parallel composition, in order to determine whether local or global termination should follow, we check the parity of program counters of the other components. We do this by checking the parity of the product of all program counters of other components.

The set of guards, $\mathcal{B}_{\text{pc}}$, where $b_{\text{pc}} \in \mathcal{B}_{\text{pc}}$, is defined as follows:

$$\begin{array}{rcl} b_{\text{pc}} & ::= & b_{\text{even}} \\ & | & b_{\text{even}} \wedge \text{Odd}(I) \\ & | & b_{\text{even}} \wedge \neg\text{Odd}(I) \end{array}$$

where,

$$\begin{array}{rcl} b_{\text{even}} & ::= & i_k = e \\ & | & b_{\text{even}} \wedge b_{\text{even}} \end{array}$$

where e is an even number, $i_k \in \mathcal{I}$, $I \subseteq \mathcal{I}$ and $\text{Odd}(I)$ denotes $\Pi_I \mod 2 = 1$ where Π_I denotes the product of the elements from set I.

The initialization predicate u is used to initialize model variables other than program counters; u is any predicate including true.

The initialization predicate u_{pc} initializes the local program counters. The initial value of a program counter is a natural number. An even value indicates a program counter is active and an odd value indicates a program counter that is inactive.

The set of initialization predicates $\mathcal{U}_{pc}$, with $u_{\text{pc}} \in \mathcal{U}_{\text{pc}}$, is defined as follows:

$$\begin{array}{rcl} u_{\text{pc}} & ::= & i_k = n \\ & | & u_{\text{pc}} \wedge u_{\text{pc}} \end{array}$$

$i_k \in \mathcal{I}$ and $n \in \mathbb{N}$.

When the value of a program counter is set to an odd value by the initilization predicate, we say that the program counter is *initially inactive.* Such program counters in a process term indicate that the process term consists of a sequential composition such that the number of program counters of the second sequent is greater than the number of program counters in the first sequent. The program counters that are only needed in the second sequent are declared at the start, but they remain inactive (evaluate to an odd value) while the process is in the first sequent.

Initial options or initial alternatives in a linear process equation of a given process term are the alternatives of the LPE consisting of the first possible actions or first possible urgency or invariant conditions of the given process term. By convention, the highest values of program counters guard the initial options of an LPE. In a linear process term $\widetilde{p}$, with an initialization predicate u_{pc}, an alternative guarded by a guard b_{pc} is an initial option of $\widetilde{p}$ only if $u_{\text{pc}} \implies b_{\text{pc}}$.

In the definition of a linear form, we observe the following:

1. An option of the LPE is never activated by odd values of program counters alone. The guard predicate b_{pc} always contains an atom that compares the value of at least one program counter against an even value.

2. An odd value of a program counter i_k ($k \in \mathbb{N}$) in a linear form represents one of the two cases:

   (a) The linear form under consideration originates from a parallel composition and the parallel component that uses program counter i_k in the definition of its linear form has terminated;

   (b) Or, the given linear form originates from a sequential composition. The first sequent has less number of parallel components than k whereas the second sequent has at least k parallel components. The odd value of program counter i_k indicates that the first sequent has not terminated yet and therefore i_k is inactive.

   Instead of odd values, we could also have used a specific value (for example $\perp$) to fulfill the purpose mentioned above. A program counter with that value would indicate both local termination of a parallel component and inactive program counter in a sequential composition. However, odd numbers possess some desirable properties. We are able to use these properties to our advantage at several places in the linearization algorithm such as:

   (a) We reuse program counters in the linearization of sequential, alternative and recursion scope operators. When reusing, we increment all the used values of a given program counter by an even number to allow for more values for representing all required guard conditions. Incrementing an odd value by an even number gives an odd number. Therefore an inactive program counter remains inactive when we reuse program counters. Hence we do not have to do any book keeping for inactive program counters.

   (b) For detecting termination of a component in a parallel composition, we check the parity of all program counters used in the linear form of that component. This is easily done by checking the parity of the product of its program counters as the product of odd values is an odd value.

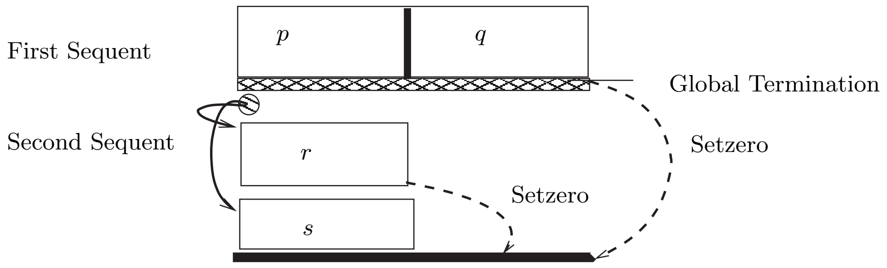
   If a specific value is decided to be used instead of odd values, then the properties of that specific value should be exploited in the linearization algorithm.

4.5 Visualization of a linear form

The linearization algorithm turns out to be complicated and involves many steps for linearization of each operator. To help understanding the basic steps of the algorithm, we devise a visualization of the process term obtained after linearization. As can be seen in the previous section, the linear form contains many features. It is only possible to illustrate a subset of these features in a two dimensional diagram. We focus on the number of program counters used in a linear form, the program counters that are active in a particular segment of a linear form, and the changes in the values of program counters as different actions of a process term are executed.

Figure 4.1 shows a graphical representation of a linear process term. Features such as lines and arrows are incorporated in the graphical representation diagram to indicate jumps in the values of program counters or to indicate the end of a parallel composition.

We discuss them below:



Linear form of $(p \parallel q);\ (r \mathbin{[\,]} s)$

Figure 4.1: A graphical representation of a linear form

1. The width of the block (length along horizontal axis) indicates the total number of program counters in a linear process term. The indices of program counters increase as we move from left to right in a block.

2. The height of a block represents the highest of the maximum values of all program counters in a process term. In our algorithm, we keep the convention that program counter i_1 has the highest maximum value. Therefore, in a parallel composition (see Section 4.6.5) we shift the program counters of the process term that has the lower value for program counter i_1. The values of program counters decrease as we move down in a block.

3. A black line at the bottom of a block stretching the entire length of a block indicates that all program counters have been set to zero.

4. A patterned horizontal line indicates the termination of a parallel composition. The value of a program counter can go below a patterned line only if all the program counters have reached it.

5. A small patterned circle represents the root of an alternative composition. The number of arrows originating from the dot indicates the total number of initially available options in the operands of the alternative composition. The arrows should end at appropriate places in alternatives. The exact place where an arrow ends in a block is kept abstract in our diagrams to avoid cluttering. In finding the linear form of an alternative composition, the values of program counters that are common in the operands of the alternative composition should be made distinct. To do this, we increment the values of program counters in one of the process terms. The zero values of the program counters are not incremented since all program counters must be set to zero at termination. In terms of our visualization, the block of one of the operands of alternative composition is placed on top of the other. A dashed arrow originates from the end of the block placed on the top of the other and ends at the bottom of the alternative composition.

6. A vertical black line in a block divides two operands of a parallel composition.

7. In a recursion scope operator, the blocks representing the linear forms of the process definitions of the recursion variables are placed on top of one another. Each block contains the name of the recursion variable it represents. The blocks may have arrows

(mimicking a call to a recursion variable) originating from their bottom surfaces and ending in the top of other blocks. The pointer update(X_i) can be viewed as ports at the bottom of a block from which arrows originate.

We do not give a visualization for the urgent action operator, the variable scope operator, the encapsulation and the channel scope operator.

4.6 Linearization Algorithm

4.6.1 Notations

The following functions and notations are used in the linearization algorithm:

1. The notation $x[a'_k/a_k]_{k \in S, P(a_k)}$ represents x with every occurrence of a_k that satisfies a certain property P replaced by a'_k, for all k in a set S.

2. The notation $x[a'/a]_{a \in S}$ denotes x with every occurrence of a in x replaced by a', for all a in a set S.

3. The function Normalize : $\mathcal{P} \rightarrow \widetilde{\mathcal{P}}$, returns the linear form of a given process term. In case a process term of the form p_{s} is input to the algorithm, then the linear process term returned does not contain pointers of the form update(X_i).

4. The function rhs : $\widetilde{\mathcal{P}} \rightarrow \overline{\mathcal{P}}$ takes a linear process term and returns the righthand side of its single recursion definition.

$$\mathrm{rhs}(\|[_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L :: \|[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X \;]\|]\|) \quad = \quad \overline{p}$$

5. The function $U : \widetilde{\mathcal{P}} \rightarrow$ Predicate, returns the predicate initializing the model variables of the given linear form.

$$U(\|[_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L :: \|[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X \;]\|]\|) \quad = \quad u$$

6. The function $U_{\mathrm{pc}} : \widetilde{\mathcal{P}} \rightarrow$ Predicate returns the predicate initializing the program counters of the given linear form.

$$U_{\mathrm{pc}}(\|[_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L :: \|[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X \;]\|]\|) \quad = \quad u_{\mathrm{pc}}$$

7. The function Sigma : $\widetilde{\mathcal{P}} \rightarrow (\mathcal{V} \mapsto \Lambda)$ returns the valuation of local model variables in a linear process term. The symbol Λ denotes the set of all possible values for model variables other than program counters.

$$\mathrm{Sigma}(\|[_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L :: \|[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X \;]\|]\|) \quad = \quad \sigma$$

8. The function pcs : $(\overline{\mathcal{P}} \cup \mathcal{B}_{\mathrm{pc}} \cup \mathcal{R}_{\mathrm{pc}} \cup \mathcal{AP}_{\mathrm{pc}}) \rightarrow \mathcal{I}$ returns the set of program counters used in its argument. The set of program counters in a linear process term is the same

as the set of program counters in the right hand side defining its recursion variable.

$$\begin{array}{lcl}
\mathrm{pcs}(\overline{p} \mathbin{[\!]} \overline{q}) & = & \mathrm{pcs}(\overline{p}) \cup \mathrm{pcs}(\overline{q}), \\
\mathrm{pcs}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{u}}) & = & \mathrm{pcs}(b_{\mathrm{pc}}) \\
\mathrm{pcs}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{update}(X_i);\ X) & = & \mathrm{pcs}(b_{\mathrm{pc}}) \\
\mathrm{pcs}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}}) & = & \mathrm{pcs}(b_{\mathrm{pc}}) \cup \mathrm{pcs}(\mathrm{ap}_{\mathrm{pc}}) \\
\mathrm{pcs}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X) & = & \mathrm{pcs}(b_{\mathrm{pc}}) \cup \mathrm{pcs}(\mathrm{ap}_{\mathrm{pc}}) \\
\mathrm{pcs}(i_k = n) & = & \{i_k\} \\
\mathrm{pcs}(\mathrm{Odd}(S)) & = & S \\
\mathrm{pcs}(\neg\mathrm{Odd}(S)) & = & S \\
\mathrm{pcs}(b_{\mathrm{pc}} \wedge b'_{pc}) & = & \mathrm{pcs}(b_{\mathrm{pc}}) \cup \mathrm{pcs}(b'_{\mathrm{pc}}) \\
\mathrm{pcs}(W_{\mathrm{pc}}) & = & W_{\mathrm{pc}} \\
\mathrm{pcs}(r_{\mathrm{pc}} \wedge r'_{\mathrm{pc}}) & = & \mathrm{pcs}(r_{\mathrm{pc}}) \cup \mathrm{pcs}(r'_{\mathrm{pcs}}) \\
\mathrm{pcs}(W_{\mathrm{pc}} : r_{\mathrm{pc}}) & = & \mathrm{pcs}(W_{\mathrm{pc}}) \cup \mathrm{pcs}(r_{\mathrm{pcs}}) \\
\mathrm{pcs}(W_{\mathrm{pc}} : r_{\mathrm{pc}}, \mathrm{ap}_{\mathrm{pc}}) & = & \mathrm{pcs}(W_{\mathrm{pc}} : r_{\mathrm{pc}}) \cup \mathrm{pcs}(\mathrm{ap}_{\mathrm{pc}})
\end{array}$$

where $b_{\mathrm{pc}}, b'_{\mathrm{pc}}$ are guard predicates in set $\mathcal{B}_{\mathrm{pc}}$, $r_{\mathrm{pc}}, r'_{\mathrm{pc}}$ are action predicates on program counters in set $\mathcal{R}_{\mathrm{pc}}$, S and W_{pc} are subsets of program counters and $k \in \mathbb{N}^{>0}, n \in \mathbb{N}$,

9. The function Count : $\overline{P} \rightarrow \mathbb{N}$ takes the linear process equation of a linear form and returns number of program counters used in it.

$$\mathrm{Count}(\overline{p}) = |\mathrm{pcs}(\overline{p})|$$

10. The function value : $\mathcal{U}_{\mathrm{pc}} \times \mathbb{N} \rightarrow \mathbb{N}$ takes a predicate of the form $\mathcal{U}_{\mathrm{pc}}$ and the index of a program counter. It returns the value assigned to the program counter with the given index in the given predicate. In our algorithm we ensure that $\mathrm{value}(u_{\mathrm{pc}}, k)$ is only called when the program counter with index k is present in the predicate u_{pc}.

    The predicate u_{pc} is a conjunction. The conjunctions are such that a program counter with a particular index is used only in one atom of the conjunction. The function value is a recursive function. It checks the atoms of u_{pc} looking for the required program counter. When the program counter with the given index is not in the current atom, it returns 0. The function value applied to a conjunction of predicates is the maximum of the values returned when applied to each predicate individually.

$$\begin{array}{lcl}
\mathrm{value}(i_k = n, k) & = & n \\
\mathrm{value}(i_k = n, l)_{l \neq k} & = & 0 \\
\mathrm{value}(u_{\mathrm{pc}} \wedge u'_{\mathrm{pc}}, k) & = & \max(\mathrm{value}(u_{\mathrm{pc}}, k), \mathrm{value}(u'_{\mathrm{pc}}, k))
\end{array}$$

    where, $k, l \in \mathbb{N}^{>0}, n \in \mathbb{N}$.

11. The function Alt : $\overline{\mathcal{P}} \rightarrow 2^{\overline{\mathcal{P}}}$ returns the set of alternatives of a process term of the form $\overline{\mathcal{P}}$.

$$\begin{array}{lcl}
\mathrm{Alt}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{u}}) & = & \{b_{\mathrm{pc}} \rightarrow p_{\mathrm{u}}\} \\
\mathrm{Alt}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{update}(X_i);\ X) & = & \{b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{update}(X_i);\ X\} \\
\mathrm{Alt}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}}) & = & \{b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}}\} \\
\mathrm{Alt}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X) & = & \{b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X\} \\
\mathrm{Alt}(\overline{p} \mathbin{[\!]} \overline{q}) & = & \mathrm{Alt}(\overline{p}) \cup \mathrm{Alt}(\overline{q})
\end{array}$$

12. The function $[\,]\ : 2^{\overline{\mathcal{P}}} \rightarrow \overline{\mathcal{P}}$ takes a set of process terms of the form $\overline{\mathcal{P}}$ as its argument and returns the term obtained after alternatively composing all the elements of the set. A special case is the empty set which returns inv true, i.e. $[\,]\ \emptyset = \text{inv true}$.

13. The function Nonterm : $\overline{\mathcal{P}} \rightarrow \overline{\mathcal{P}}$: takes a process term $\overline{p}$ and returns a process term consisting of only the non-terminating alternatives of $\overline{p}$. A non-terminating alternative consists of either an action followed by a recursive call to X, or an invariant or an urgency condition.

$$\begin{array}{lcl} \mathrm{Nonterm}(\overline{p}) & = & [\,]\ \{b_{pc} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X \mid b_{pc} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X \in \mathrm{Alt}(\overline{p})\} \\ & & [\,]\ \{b_{pc} \rightarrow p_{\mathrm{u}} \mid b_{pc} \rightarrow p_{\mathrm{u}} \in \mathrm{Alt}(\overline{p})\} \\ & & [\,]\ \{b_{pc} \rightarrow p_{\mathrm{act}}, \mathrm{update}(X_i);\ X \\ & & \mid b_{pc} \rightarrow p_{\mathrm{act}}, \mathrm{update}(X_i);\ X \in \mathrm{Alt}(\overline{p}) \\ & & \} \end{array}$$

14. The function Term : $\overline{\mathcal{P}} \rightarrow \overline{\mathcal{P}}$: takes a process term $\overline{p}$ and returns a process term consisting of only the terminating alternatives of $\overline{p}$. A terminating alternative consists of an action without a trailing call to the recursion variable X.

$$\mathrm{Term}(\overline{p}) \quad = \quad [\,]\ \{b_{pc} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \mid b_{pc} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \in \mathrm{Alt}(\overline{p})\}$$

In the following sections, we linearize different forms of the input process term p_{s} one by one.

4.6.2 Atomic actions

The linear form of an atomic action is defined by the Normalize function as follows:

$$\begin{array}{ll} \mathrm{Normalize}(p_{\mathrm{atom}}) = & \llbracket_{\mathrm{V}}\ \{i_1 \mapsto \bot\}, \emptyset, \emptyset \\ & ::\ \ \llbracket_{\mathrm{R}}\ \{X \mapsto (i_1 = 2) \rightarrow p_{\mathrm{atom}}, \{i_1\} : i_1 = 0\} \\ & \qquad :: \mathrm{true} \wedge (i_1 = 2) \curvearrowright X \\ & \qquad \rrbracket \\ & \rrbracket \end{array}$$

It is represented by a square with a black bottom in Figure 4.2.

p_{atom}

Figure 4.2: An atomic action

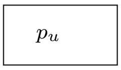
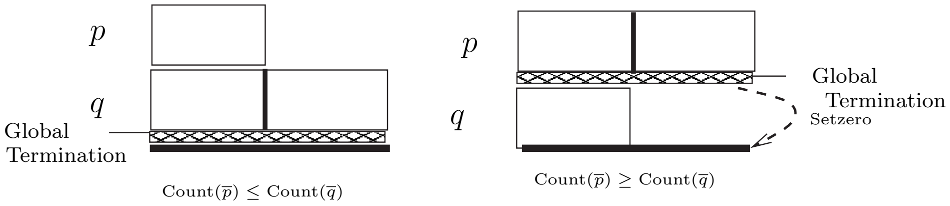


Figure 4.3: An invariant or urgency condition


p
q
Global
Termination
Count($\overline{p}$) $\leq$ Count($\overline{q}$)
p
q
Global
Termination
Setzero
Count($\overline{p}$) $\geq$ Count($\overline{q}$)

(a) (b)

Figure 4.4: Sequential Composition

4.6.3 Invariants or Urgency Conditions

An invariant or urgency condition, denoted by p_{u}, is defined by the Normalize function as follows:

$$
\begin{aligned}
\mathrm{Normalize}(p_{\mathrm{u}}) = \; & [\![_{\mathrm{V}}\, \{i_1 \mapsto \bot\}, \emptyset, \emptyset \\
& ::\; [\![_{\mathrm{R}}\, \{X \mapsto (i_1 = 2) \to p_{\mathrm{u}}\} \\
& \quad\; :: \mathrm{true} \wedge (i_1 = 2) \curvearrowright X \\
& \quad\;]\!] \\
&]\!]
\end{aligned}
$$

An invariant or urgency condition is represented by a square without a black bottom in Figure 4.3.

4.6.4 Sequential Composition

A process $p;\, q$ first behaves as p. After p has terminated, $p;\, q$ continues behaving as q.

Assume

$$
\begin{aligned}
\mathrm{Normalize}(p) = \widetilde{p} = \; & [\![_{\mathrm{V}}\, \sigma_{\mathrm{pc}}^{p} \cup \sigma_p, C_p, L_p \\
& ::\; [\![_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u_p \wedge u_{\mathrm{pc}}^{p} \curvearrowright X \;]\!] \\
&]\!]
\end{aligned}
$$

and

$$
\begin{aligned}
\mathrm{Normalize}(q) = \widetilde{q} = \; & [\![_{\mathrm{V}}\, \sigma_{\mathrm{pc}}^{q} \cup \sigma_q, C_q, L_q \\
& ::\; [\![_{\mathrm{R}}\, \{X \mapsto \overline{q}\} :: u_q \wedge u_{\mathrm{pc}}^{q} \curvearrowright X \;]\!] \\
&]\!]
\end{aligned}
$$

Process terms p and q both belong to the set $\mathcal{P}$. The case where q is a recursion variable (i.e. a process term $p_{\mathrm{s}};\, X_i$, is given as input to the algorithm) is dealt with the linearization of the recursion scope operator (See Section 4.6.7).

In the linear form of a sequential composition, the values of program counters that are common in the linear forms of the two sequents are made distinct by incrementing the values of all program counters in the first sequent by the maximum value of the program counter

i_1 in the second. Recall that in a linear process term, i_1 always has the greatest or one of the greatest values among all program counters. In this way, no two alternatives, where one is in the first sequent and the other is in the second sequent, can get activated by the same values of program counters. Let p be the first sequent and q be the second sequent. Instead of incrementing **all** program counters' values in p by the maximum value of i_1 in q, we could also increment a program counter i_k in p, by the maximum value of i_k in q or zero in case the total number of program counters in q is less than k. This approach will also make the value of i_k distinct in the two operands of sequential composition. Adopting this approach would result in a linear form best explained by modifying Figure 4.4(b) as follows:

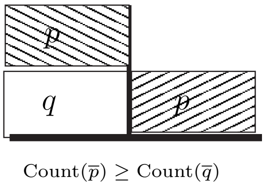

p
q
p
$\text{Count}(\overline{p}) \geq \text{Count}(\overline{q})$

Figure 4.5: A different way of incrementing program counter values

The shaded area in the Figure 4.5 represents the linear form of the first sequent p. We adopt the first approach of incrementing **all** program counter values in the first sequent by the maximum value of i_1 in the second sequent as it is simpler.

The total number of program counters used in the linear form is the maximum of the numbers of program counters in the linear forms of each sequent. Initially only the initial options of the first sequent are enabled. Only when the first sequent terminates, the options of the second sequent are activated.

If $\text{Term}(\overline{p}) = \text{inv}\,\text{true}$, i.e. the first sequent does not terminate, then

$$\text{Normalize}(p;\ q) = \widetilde{p}$$

else,

$$\begin{array}{l}
\text{Normalize}(p;\ q) = \widetilde{r} = \\
\quad [\![_{\text{V}}\ \sigma^r_{\text{pc}} \cup \sigma_p \cup \sigma_q, C_p \cup C_q, L_p \cup L_q \\
\quad ::\ [\![_{\text{R}}\ \{X \mapsto \text{Setzero}(\text{FSequent}(\text{Incrpcs}(\overline{p}, \text{value}(u^q_{\text{pc}}, 1)), \widetilde{q})\ [\!]\ \overline{q})\} \\
\qquad :: u_p \wedge u^r_{\text{pc}} \curvearrowright X \\
\qquad]\!] \\
\quad]\!]
\end{array}$$

where,

1. The valuation σ^r_{pc} defining program counters is given below:

$$\sigma^r_{\text{pc}} \quad = \quad \{i_1 \mapsto \bot, \ldots, i_{\max(\text{Count}(\overline{p}), \text{Count}(\overline{q}))} \mapsto \bot\}$$

The total number of program counters used in the linear form is thus the maximum of the numbers of program counters in the two sequents.

2. The initialization predicate initializes the program counters. Each program counter of the first sequent is initilaized to the sum of its initial value and the initial value of i_1 in the second sequent.

$$\begin{array}{rcl} u^r_{\text{pc}} & = & \text{Incrpcs}(u^p_{\text{pc}}, \text{value}(u^q_{\text{pc}}, 1)) \wedge \\ & & \bigwedge_{i_j \in \text{pcs}(\overline{q}) \backslash \text{pcs}(\overline{p})} i_j = \text{value}(u^p_{\text{pc}}, 1) + \text{value}(u^q_{\text{pc}}, 1) + 1 \end{array}$$

   When the number of program counters in the linear form of the second sequent is greater than the number of program counters in that of the first, the program counters that are not used initially are set to an odd value.

3. The function Incrpcs : $(\mathcal{U}_{\text{pc}} \times \mathbb{N} \rightarrow \mathcal{U}_{\text{pc}}) \cup (\overline{\mathcal{P}} \times \mathbb{N} \rightarrow \overline{\mathcal{P}})$ takes a predicate on program counters or a process term as the first argument and a natural number as the second argument. It increments the values assigned to the program counters in the given predicate or in the given process term by the given number.

$$\text{Incrpcs}(x, n) \quad = \quad x[i_k = c_k + n / i_k = c_k]_{k \in \mathbb{N}}$$

   where c_k is a natural number for all values of k .

4. The function FSequent : $\overline{\mathcal{P}} \times \widetilde{P} \rightarrow \overline{\mathcal{P}}$ takes two process terms that are to be joined in a sequential composition. The first argument in the function call to FSequent should be the right hand side of the linear process term $\bar{p}$ and the second argument should the linear form $\tilde{q}$. The function FSequent removes the action predicate ap_{pc} from the terminating alternatives of the first argument. Inplace of the removed predicates ap_{pc}, the function FSequent appends the following to the terminating alternatives of the first argument:

   (a) An action predicate initializing the local model variables of the second sequent according to its initialization predicate, u_q. The jump set of this action predicate consists of the local discrete and continuous variables of the second sequent;

   (b) An action predicate initializing the program counters according to the initialization predicate u^q_{pc} of the second sequent; and

   (c) A deactivation of the program counters of $\overline{p}$ that are not used in $\overline{q}$, in case the first sequent has more program counters than the second sequent; and

   (d) Finally, a recursive call to X;

$$\begin{array}{lll} \multicolumn{3}{l}{\text{FSequent}(\overline{p}, \widetilde{q}) =} \\ \;\text{Term}(\overline{p}) & [& b_{\text{pc}} \rightarrow p_{\text{act}}, \\ & & \{v \mid v \in \text{dom}(\text{Sigma}(\widetilde{q}))\} : U(\widetilde{q}), \\ & & \text{pcs}(\text{rhs}(\widetilde{q})) : U_{\text{pc}}(\widetilde{q}), \\ & & \text{pcs}(\overline{p}) \backslash \text{pcs}(\text{rhs}(\widetilde{q})) : \\ & & \bigwedge_{i_d \in \text{pcs}(\overline{p}) \backslash \text{pcs}(\text{rhs}(\widetilde{q}))} i_d = \text{value}(U_{\text{pc}}(\widetilde{q}), 1) + 1;\ X \\ & / & b_{\text{pc}} \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}} \\ &] & \\ \;[] \ \text{Nonterm}(\overline{p}) & & \end{array}$$

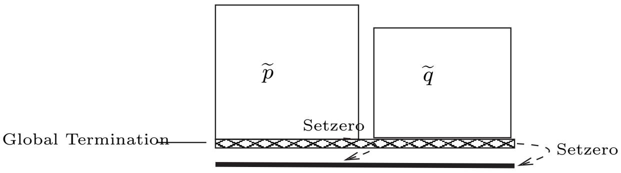


Figure 4.6: Parallel Composition

5. The function Setzero : $\overline{\mathcal{P}} \rightarrow \overline{\mathcal{P}}$ takes a process term of the form $\overline{p}$. It sets all the program counters of $\overline{p}$ to zero, in the terminating options of $\overline{p}$.

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathrm{Setzero}(\overline{p}) =} \\
\ \mathrm{Term}(\overline{p}) & [& b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \\
 & & \mathrm{pcs}(\overline{p}) : \bigwedge_{i_d \in \mathrm{pcs}(\overline{p})} i_d = 0 \\
 & / & b_{\mathrm{pc}} \rightarrow p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \\
 &] & \\
\multicolumn{3}{l}{\ [\!]\ \mathrm{Nonterm}(\overline{p})}
\end{array}
$$

4.6.5 Parallel Composition

Assume

$$
\begin{array}{rl}
\mathrm{Normalize}(p_{\mathrm{s}}) = \widetilde{p} = & \|[_{\mathrm{V}}\ \sigma^p_{\mathrm{pc}} \cup \sigma_p, C_p, L_p \\
 & ::\ \ \|[_{\mathrm{R}}\ \{X \mapsto \overline{p}\} :: u_p \wedge u^p_{\mathrm{pc}} \curvearrowright X\ ]\| \\
 &]\|
\end{array}
$$

and

$$
\begin{array}{rl}
\mathrm{Normalize}(q_{\mathrm{s}}) = \widetilde{q} = & \|[_{\mathrm{V}}\ \sigma^q_{\mathrm{pc}} \cup \sigma_q, C_q, L_q \\
 & ::\ \ \|[_{\mathrm{R}}\ \{X \mapsto \overline{q}\} :: u_q \wedge u^q_{\mathrm{pc}} \curvearrowright X\ ]\| \\
 &]\|\ .
\end{array}
$$

Only the linear form of a parallel composition of the form $p_{\mathrm{s}} \parallel q_{\mathrm{s}}$ is given, as parallel composition between process terms of the form $p, q \in \mathcal{P}$ that cannot both be written as a term in $\mathcal{P}_{\mathrm{s}}$ is not allowed in the input language of the linearization algorithm.

We do not reuse the program counters when joining the linear forms $\widetilde{p}$ and $\widetilde{q}$ in parallel composition. We differentiate between the program counters of $\widetilde{p}$ and $\widetilde{q}$ by shifting the subscripts of all program counters in one of the process terms by the number of program counters in the other. We shift the program counters of the process term that has the smallest maximum value for i_1. In this way, in our visualization the first column of blocks has the maximum height. The total number of program counters in a linear form of a parallel composition is the sum of the number of program counters in the linear forms of the two operands.

Assume that the maximum value of program counter i_1 in $\widetilde{p}$ is greater than or equal to its maximum value in $\widetilde{q}$, i.e.

value$(u^q_{\mathrm{pc}}, 1) \leq$ value$(u^p_{\mathrm{pc}}, 1)$, then,

$$
\begin{array}{l}
\text{Normalize}(p_{\mathrm{s}} \parallel q_{\mathrm{s}}) = \\
\|[_{\mathrm{V}}\ \sigma^{p}_{\mathrm{pc}} \cup \text{Shiftpcs}(\sigma^{q}_{\mathrm{pc}}, \text{Count}(\overline{p})) \cup \sigma_p \cup \sigma_q, C_p \cup C_q, L_p \cup L_q \\
::\ \ \|[_{\mathrm{R}}\ \ \{X \mapsto \text{Setzero}\ \ \left(\begin{array}{l}
\quad\ \text{Extend}(\overline{p}, \text{Shiftpcs}(\overline{q}, \text{Count}(\overline{p}))) \\
[\!]\ \text{Extend}(\text{Shiftpcs}(\overline{q}, \text{Count}(\overline{p})), \overline{p}) \\
\Big[\!\Big]\ \begin{array}{l}
\{\text{COM}(\text{alt}_p, \text{alt}_q) \mid \text{alt}_p \in \text{Alt}(\overline{p}), \\
\text{alt}_q \in \text{Alt}(\text{Shiftpcs}(\overline{q}, \text{Count}(\overline{p}))), \\
\text{match}(\text{alt}_p, \text{alt}_q) \\
\}
\end{array}
\end{array}\right) \\
\quad\ :: (u_p \wedge u_q) \wedge (u^{p}_{\mathrm{pc}} \wedge \text{Shiftpcs}(u^{q}_{\mathrm{pc}}, \text{Count}(\overline{p}))) \curvearrowright X \\
\quad\ ]\!| \\
\\
]\!|,
\end{array}
$$

where,

1. The function Shiftpcs : $((\mathcal{V} \mapsto \Lambda) \times \mathbb{N} \to (\mathcal{V} \mapsto \Lambda)) \cup (\mathcal{U}_{\mathrm{pc}} \times \mathbb{N} \to \mathcal{U}_{\mathrm{pc}}) \cup (\overline{\mathcal{P}} \times \mathbb{N} \to \overline{\mathcal{P}})$ takes as the first parameter a valuation (a set of mappings of variables to values in some value set Λ), or a predicate, or a process term, and as the second parameter a natural number. It shifts the subscripts of all the program counters in the given valuation, predicate or the process term by the given number.

$$\text{Shiftpcs}(x, d) \quad = \quad x[i_{k+d}/i_k]_{k \in \mathbb{N}}$$

2. The symmetric function $\texttt{match} : \overline{\mathcal{P}} \times \overline{\mathcal{P}} \to \mathcal{B}\text{ool}$, takes as its parameters two alternatives, one from the linear process equation of the linear form of each operand of parallel composition. In case its parameters contain matching send and receive actions, the function match returns true else it returns false. The rules for the function match strip off the unnecessary details from an alternative: the guards, calls to the recursive variable X and all action predicates are removed. The following rules define the function match:

   (a) $\text{match}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \text{update}(X_i);\ X), \overline{p}) = \text{match}(p_{\mathrm{act}}, \overline{p})$

   (b) $\text{match}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \text{ap}_{\mathrm{pc}};\ X), \overline{p}) = \text{match}(p_{\mathrm{act}}, \overline{p})$

   (c) $\text{match}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \text{ap}_{\mathrm{pc}}), \overline{p}) = \text{match}(p_{\mathrm{act}}, \overline{p})$

   (d) $\text{match}(p_{\mathrm{act}}, (p'_{\mathrm{atom}}, \text{ap})) = \text{match}(p_{\mathrm{act}},\ p'_{\mathrm{atom}})$

   (e) $\text{match}(h\,!\,\mathbf{e}_n, W : r,\ h\,?\,\mathbf{x}_n, W' : r') = \text{true}$, where W and W' are environment variable sets, r and r' are predicates and h is a communication channel.

   If none of the rules from above can be applied, also after interchanging the arguments, then the function match returns false.

3. The symmetric function COM : $\overline{P} \times \overline{P} \to \overline{P} \cup \{\bot\}$ takes as its parameters two alternatives, one from the linear process equation of the linear form of each operand of parallel composition. In case, its parameters contain matching send and receive actions, the function COM returns the result of communication between its parameters. It is a symmetric function. Parallel composition is only defined for process terms of the form p_{s}.

The linear form of a p_{s} process term does not consist of any alternative with a pointer (See Section 4.4 and Section 4.6.7). Therefore for a parameter containing a pointer, of the form update(X_i), COM returns $\perp$. The function COM is defined below:

(a) $\mathrm{COM}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{update}(X_i);\ X), \overline{p}) = \perp$, where X_i is a recursion variable.

(b) In case $\mathrm{match}(p_{\mathrm{act}}, q_{\mathrm{act}})$,

$$\begin{aligned}
&\mathrm{COM}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}}), (b'_{\mathrm{pc}} \to q_{\mathrm{act}}, \mathrm{ap}'_{\mathrm{pc}})) = \\
&\qquad b_{\mathrm{pc}} \wedge b'_{\mathrm{pc}} \to \mathrm{com}(p_{\mathrm{act}}, q_{\mathrm{act}}), \mathrm{ap}_{\mathrm{pc}}, \mathrm{ap}'_{\mathrm{pc}} \\
&\mathrm{COM}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X), (b'_{\mathrm{pc}} \to q_{\mathrm{act}}, \mathrm{ap}'_{\mathrm{pc}})) = \\
&\qquad b_{\mathrm{pc}} \wedge b'_{\mathrm{pc}} \to \mathrm{com}(p_{\mathrm{act}}, q_{\mathrm{act}}), \mathrm{ap}_{\mathrm{pc}}, \mathrm{ap}'_{\mathrm{pc}}[1/0];\ X \\
&\mathrm{COM}((b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X), (b'_{\mathrm{pc}} \to q_{\mathrm{act}}, \mathrm{ap}'_{\mathrm{pc}};\ X)) = \\
&\qquad b_{\mathrm{pc}} \wedge b'_{\mathrm{pc}} \to \mathrm{com}(p_{\mathrm{act}}, q_{\mathrm{act}}), \mathrm{ap}_{\mathrm{pc}}, \mathrm{ap}'_{\mathrm{pc}};\ X,
\end{aligned}$$

where the notation $\mathrm{ap}_{\mathrm{pc}}[1/0]$ represents an action predicate $\mathrm{ap}_{\mathrm{pc}}$ with all the predicates setting a program counter to value zero replaced by predicates setting them to 1.

$$\mathrm{ap}_{\mathrm{pc}}[1/0] \quad = \quad \mathrm{ap}_{\mathrm{pc}}[i_k = 1/i_k = 0]_{k \in \mathbb{N}}$$

In a communication between two actions, where one of the actions is terminating and the other is non-terminating, the resulting communication cannot be a terminating action. A communication action may be a terminating action of $p \parallel q$, only if it is a communication between terminating actions of $\widetilde{p}$ and $\widetilde{q}$. In the communication between a terminating and a non-terminating action, the program counters that are being set to zero in the terminating action must be set to 1 instead. This is done through $\mathrm{ap}_{\mathrm{pc}}[1/0]$.

(c) In case $\mathrm{match}(\mathrm{alt}_p, \mathrm{alt}_q)$= false, where $\mathrm{alt}_p \in \mathrm{Alt}(\overline{p})$, and $\mathrm{alt}_q \in \mathrm{Alt}(\overline{q})$ then the function COM returns inv true.

$$\mathrm{COM}(\mathrm{alt}_p, \mathrm{alt}_q) = \mathrm{inv}\ \ \mathrm{true},$$

Note that the function COM is always called with those pairs of alternatives of $\overline{p}$ and $\overline{q}$ that match.

4. The function $\mathrm{com} : \mathcal{P}_{\mathrm{act}} \times \mathcal{P}_{\mathrm{act}} \to \mathcal{P}_{\mathrm{act}}$ takes two action process terms and returns their communication.

$$\mathrm{com}((h\,!\,e_{\mathbf{n}}, \mathrm{ap}), (h\,?\,x_{\mathbf{n}}, \mathrm{ap}')) = h\,!?\,x_{\mathbf{n}} := e_{\mathbf{n}}, \mathrm{ap}, \mathrm{ap}'$$

The function com is a partial function. In our linearization algorithm, the function com is only called with parameters that can communicate as calculated by function match.

5. The semantics of parallel operator includes details of communication, synchronization and interleaving of actions of parallel components. The communication between parallel components was dealt within the function COM. The interleaving of actions and synchronization of delays is dealt in the function Extend. The function $\mathrm{Extend} : \overline{P} \times \overline{P} \to \overline{P}$ takes the linear process equations of the parallel components. It returns the first argument with some modifications in its alternatives containing terminating actions. Alternatives containing non-terminating actions and invariant or urgency conditions are

not modified by the function Extend. Termination of a component needs to be handled specially as explained in the paragraph below.

We recall from section 4.4 the concepts of local and global termination. A parallel composition terminates when all its components terminate. When one component of a parallel composition terminates while the other components have not terminated yet, then it is called *local* termination of the terminated component. When the last component of a parallel composition terminates, it is called *global* termination. When a process term terminates locally, its program counters are set to 1 (i.e. an odd value) instead of a 0. On global termination, the program counters of all process terms in parallel are set to 0 and then we do not need another recursive call to variable X.

In the function Extend, when a component of parallel composition performs a terminating action, we check whether the process terms in parallel have already terminated. This check is done in the guard condition of the terminating action. We check the parity of the product of all program counters of the process terms in parallel. If a program counter of the process terms in parallel still has an even value, (i.e. parity of the calculated product is even), then one of the parallel components still has to perform an action. In this case, the program counters of the terminating process are set to 1 and a recursive call to X is added. Else, if the parity of the calculated product is odd, then this indicates that the processes in parallel have already terminated locally and the action under consideration is indeed the terminating action of the parallel composition. Then, in the terminating action, we set the program counters of all processes to zero.

In the linear form of a parallel composition, there are *two* alternatives to represent each terminating action of each parallel component. In the guard condition of such an alternative, we include a condition to check the parity of the program counters of process terms in parallel. If the program counters of all process terms in parallel are odd, then all process terms in parallel have terminated and the action under discussion terminates the whole parallel composition. In this case, we set all the program counters to zero after the terminating action and do not add a recursive call to X. On the other hand, if not all program counters of processes in parallel are odd, then at least one process in parallel has not terminated. In this case, the action under consideration only terminates the process containing it. Then we set the program counters of the termination process to 1 and include a recursive call to X.

The linear form of a parallel composition, or the linear form of a process term with parallel composition in its last sequent, can be identified by its terminating options. The terminating actions of a linear form of such a process term are guarded by predicates of the form $b_{\text{even}} \wedge \text{Odd}(S)$, where S is a set of program counters. In Hybrid χ, as is in other process algebras, the parallel operator is defined as a binary operator. A process term $p \parallel q \parallel r$ is defined as $(p \parallel q) \parallel r$ or $p \parallel (q \parallel r)$. (The parallel operator is associative.) While linearizing a process term $(p \parallel q) \parallel r$, in the function Extend, the terminating options of the linear form of $(p \parallel q)$ are modified to include the parity checking for the program counters of the linear form of r. This restricts the number of new alternatives that will be added to the linear form of $(p \parallel q)$ to obtain the linear form of $(p \parallel q) \parallel r$ to the size of r. Otherwise, not distinguishing that one component (i.e. $p \parallel q$) of a parallel composition is itself a parallel composition, (or contains a parallel composition in its last sequent), results in an increase in size which is equal to the size of r plus the number of parallel components in the last sequent.

In case one of the process terms does not terminate, for example inv u and urg u, then the parallel composition does not terminate. In this case, the terminating alternatives of the process term that terminates are also appended with a recursive call to X.

The function Extend : $\overline{\mathcal{P}} \times \overline{\mathcal{P}} \to (\overline{\mathcal{P}} \cup \{\perp\})$ takes the right hand sides of the linear forms of two process terms in parallel composition with each other. The function modifies the terminating options of the first argument as discussed before.

The function Extend is defined as follows:

(1) $\text{Extend}(\overline{p} \mathbin{[\!]} \overline{p}', \overline{q}) = \text{Extend}(\overline{p}, \overline{q}) \mathbin{[\!]} \text{Extend}(\overline{p}', \overline{q})$

(2) $\text{Extend}(b_{\text{pc}} \to p_{\text{u}}, \overline{q}) = b_{\text{pc}} \to p_{\text{u}}$

(3) $\text{Extend}((b_{\text{pc}} \to p_{\text{act}}, \text{update}(X_i);\ X), \overline{q}) = \perp$

For the remaining alternatives in the linear form of p, two cases for the second parameter are distinguished. If q does not terminate, then the terminating action of the linear form of p are made non terminating as well. As in that case, $p \parallel q$ does not terminate.

If $\text{Term}(\overline{q}) = \text{inv}\ \text{true}$,

(1) $\text{Extend}((b_{\text{pc}} \to p_{\text{act}}, \text{ap}_{\text{pc}};\ X), \overline{q}) = b_{\text{pc}} \to p_{\text{act}}, \text{ap}_{\text{pc}};\ X$

(2) $\text{Extend}((b_{\text{pc}} \to p_{\text{act}}, \text{ap}_{\text{pc}}), \overline{q}) = b_{\text{pc}} \to p_{\text{act}}, \text{ap}_{\text{pc}}[1/0];\ X$

Else if $\text{Term}(\overline{q}) \neq \text{inv}\ \text{true}$, the function Extend is defined as follows:

(1) $\text{Extend}((b_{\text{even}} \to p_{\text{act}}, \text{ap}_{\text{pc}};\ X), \overline{q})$
$$= b_{\text{even}} \to p_{\text{act}}, \text{ap}_{\text{pc}};\ X$$

(2) $\text{Extend}((b_{\text{even}} \to p_{\text{act}}, \text{ap}_{\text{pc}}), \overline{q})$
$$\begin{aligned} &= b_{\text{even}} \wedge \text{Odd}(\text{pcs}(\overline{q})) \to p_{\text{act}}, \text{ap}_{\text{pc}} \\ &\mathbin{[\!]}\ b_{\text{even}} \wedge \neg\text{Odd}(\text{pcs}(\overline{q})) \to p_{\text{act}}, \text{ap}_{\text{pc}}[1/0];\ X \end{aligned}$$

As mentioned before, for each terminating action in all parallel components (in this case p and q), the linear form of the parallel composition of $p \parallel q$ contains two alternatives. In the definition of Extend above, the first alternative represents the case when all process terms in parallel have terminated. The second alternative represents the case when all components in parallel have not terminated. Note that in the definition of the Normalize, the function Setzero is applied to the resulting process term obtained after applying Extend and COM. Therefore, we do not need to set all program counters to zero in the definition of function Extend, even for global termination.

(3) In the linear form of p, an alternative of the form $b_{\text{even}} \wedge \text{Odd}(S) \to p_{\text{act}}, \text{ap}_{\text{pc}}$, where S is a set of program counters, indicates that p contains a parallel operator. As mentioned before, in this case, we simply add the program counters of the linear form of q to the set S of program counters.

$\text{Extend}((b_{\text{even}} \wedge \text{Odd}(S) \to p_{\text{act}}, \text{ap}_{\text{pc}}), \overline{q})$
$$= b_{\text{even}} \wedge \text{Odd}(S \cup \text{pcs}(\overline{q})) \to p_{\text{act}}, \text{ap}_{\text{pc}}$$

Repeating the same procedure with a non-terminating alternative, $b_{\text{even}} \wedge \neg\text{Odd}(S) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X$, requires care. If p is a sequential composition and the parallel operator occurs in the first sequent, then we must not modify the given alternative. For example, suppose $p = (a \parallel b);\ c$. Then in the linear form of $p \parallel q$, the alternatives representing $a \parallel b$ must not be modified. We can distinguish if an alternative $b_{\text{even}} \wedge \neg\text{Odd}(S) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X$ represents an action in the last sequent of p or not. If the given alternative represents an action in the last sequent, then the action predicate ap_{pc} will set program counters to value 1. In case the given alternative represents an action which is not in the last sequent, then during the linearization of the sequential composition, the values in the predicate ap_{pc} will get incremented. Hence we introduce a function allone. The function allone : $\mathcal{AP}_{\text{pc}} \rightarrow \mathcal{B}\text{ool}$ returns true if all the program counters in ap_{pc} are being set to 1. The function allone is defined below:

$$\begin{array}{lcl}\text{allone}(W_{\text{pc}} : \bigwedge_{i_d \in W_{\text{pc}}} i_d = 1) & = & \text{true} \\ \text{allone}(W_{\text{pc}} : r_{\text{pc}}, \text{ap}_{\text{pc}}) & = & \text{allone}(W_{\text{pc}} : r_{\text{pc}}) \wedge \text{allone}(\text{ap}_{\text{pc}})\end{array}$$

The function Extend with $b_{\text{even}} \wedge \neg\text{Odd}(S) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X$ as its first argument is then defined as follows:

(a) $\text{Extend}((b_{\text{even}} \wedge \neg\text{Odd}(S) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X), \overline{q})$
$= b_{\text{even}} \wedge \neg\text{Odd}(S \cup \text{pcs}(\overline{q})) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X,$
where $\text{allone}(\text{ap}_{\text{pc}}) = \text{true}$.

(b) $\text{Extend}((b_{\text{even}} \wedge \neg\text{Odd}(S) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X), \overline{q})$
$= b_{\text{even}} \wedge \neg\text{Odd}(S) \rightarrow p_{\text{act}}, \text{ap}_{\text{pc}};\ X$
where $\text{allone}(\text{ap}_{\text{pc}}) \neq \text{true}$.

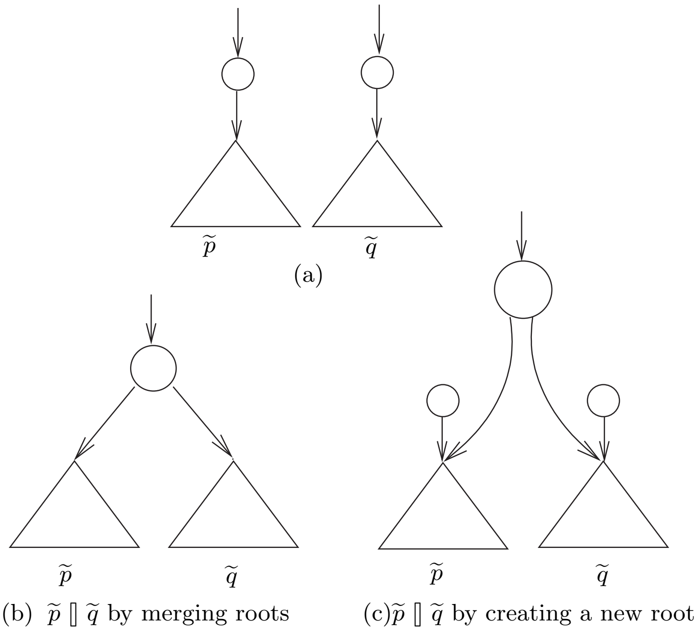
4.6.6 Alternative Composition

The alternative composition of process terms provides a choice between them. The choice is resolved as soon as an action is performed, in favor of the process term the action of which has been executed. A graphical representation of two process terms and their alternative composition is given in Figure 4.7. As shown in the figure, there are two ways in which two process terms, p and q, can be alternatively composed:

1. In Figure 4.7(b), the roots of the two process terms are merged to obtain a root for their alternative composition. Transitions emerging from this root are the same as the transitions emerging from the roots of p and q.

2. In Figure 4.7(c), a new root for the resulting alternative composition is created, which is distinct from the roots of the given alternatives. Transitions emerging from the new root end at proper places within the transition trees of p and q. Note that the original roots of the alternatives are retained in this way of alternative composition but these roots are no longer initial states.

Merging two roots to obtain a new root for the alternative composition works only if the operand process terms do not have self recursion and none of the process terms have initial parallelism. To explain further, we present scenarios of self recursion and initial parallelism in operands of an alternative composition below:



(b) $\widetilde{p} \mathbin{[\,]} \widetilde{q}$ by merging roots (c)$\widetilde{p} \mathbin{[\,]} \widetilde{q}$ by creating a new root

Figure 4.7: Two Techniques for alternative composition

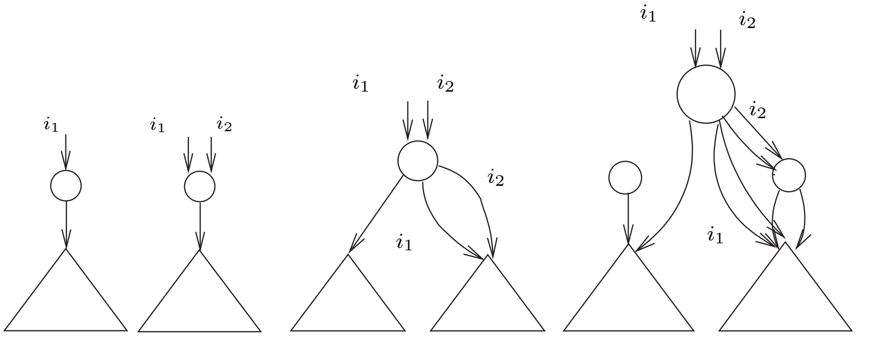
In terms of transition systems, self recursion means that there is a transition emerging from within the tree of a process term and ending at its root. When roots of operands are merged to form the root of the alternative composition, then a transition ending at the root of alternative composition activates both operands which is not intended.

In a linear form of $p \mathbin{[\,]} q$, the initial values of program counters activate the initial options of both p and q. Consider the case where one of the operand, has self recursion, for example let

$$q = \llbracket_{\mathrm{R}} \{X_1 \mapsto a; b; X_1\} :: X_1 \rrbracket$$

In the linear form of $\widetilde{q}$, after actions a and b, a program counter of q will be set back to its initial value. In $p \mathbin{[\,]} q$ obtained by merging the roots of operands, if q is chosen, then resetting a program counter to its initial value activates the initial options of p also. This problem does not arise when a new root is created for the alternative composition. In alternative composition with a new root, the initial values of program counters in $p \mathbin{[\,]} q$ are distinct from their initial values in p and q. The value to which a program counter is reset in case of self recursion is not its initial value in the alternative composition, but its initial value in the operand with self recursion.

To observe the initial parallelism in $p \mathbin{[\,]} q$, let $q = a;\ b \parallel c;\ d$. The term q can start with either performing action a or c. In terms of our linear form, there are two program counters i_1 and i_2 that are active initially in $\widetilde{q}$. Therefore also in the linear form of alternative composition $p \mathbin{[\,]} q$ at least two program counters are initially active. If process q is chosen from the alternative composition $p \mathbin{[\,]} q$, then when the first action of $\widetilde{q}$ is performed, one of its program counters is decremented, whereas the other program counter is still at its initial value. If the roots of p and q are merged to form the root of the alternative composition, then in $p \mathbin{[\,]} q$, after doing an action of q, an action of $\widetilde{p}$ is still possible. See Figure 4.8(b). If first the action of $\widetilde{q}$ governed by program counter i_2 is performed, then after the action, i_1 is still



(a) $\widetilde{p}$ $\widetilde{q}$ (b) $\widetilde{p} \mathbin{[\,]} \widetilde{q}$ by merging roots (c) $\widetilde{p} \mathbin{[\,]} \widetilde{q}$ by creating a new root

Figure 4.8: Alternative Composition with initial parallelism in q

at its initial value and can activate an option of $\widetilde{p}$.

When we create a new root for $p \mathbin{[\,]} q$, then after executing an action of q, all program counters except the one governing the action executed, are reset according to the root of $\widetilde{q}$, i.e. according to the initial values of program counters in the linear form of q. This shown by in the figure 4.8 (c).

The algorithm that creates a new root is a general purpose algorithm but it yields unreachable states, in case there is no self-recursion or initial parallelism. Therefore, we give in this section two algorithms for alternative composition, one that merges the roots of operands to obtain the root of alternative composition and the other that creates a new root for alternative composition. Depending on the scenario at hand, a different algorithm for alternative composition can be adopted.

Assume

$$\begin{aligned}\text{Normalize}(p) = \widetilde{p} = \ & \llbracket_{\text{V}}\, \sigma^p_{\text{pc}} \cup \sigma_p, C_p, L_p \\ & :: \llbracket_{\text{R}}\, \{X \mapsto \overline{p}\} :: u_p \wedge u^p_{\text{pc}} \curvearrowright X \,\rrbracket \\ & \rrbracket\end{aligned}$$

and

$$\begin{aligned}\text{Normalize}(q) = \widetilde{q}\ & \llbracket_{\text{V}}\, \sigma^q_{\text{pc}} \cup \sigma_q, C_q, L_q \\ & :: \llbracket_{\text{R}}\, \{X \mapsto \overline{q}\} :: u_q \wedge u^q_{\text{pc}} \curvearrowright X \,\rrbracket \\ & \rrbracket,\end{aligned}$$

Alternative Composition without a new root

This algorithm is applicable if the linear forms of both p and q have one program counter initially active and none of the operands have self recursion. The linear forms $\widetilde{p}$ and $\widetilde{q}$ are tested for initial parallelism and self recursion as follows:

1. If only the value of i_1 is even in the initialization predicate of a linear form, then the linear form does not have initial parallelism. Thus for absence of initial parallelism in $\widetilde{p}$ and $\widetilde{q}$, the following predicates should be true. :

   $$(\text{value}(u^p_{\text{pc}}, 1) \quad \text{mod } 2 = 0) \wedge \text{Odd}(\text{pcs}(\overline{p}) \backslash \{i_1\})$$

   and

   $$(\text{value}(u^q_{\text{pc}}, 1) \quad \text{mod } 2 = 0) \wedge \text{Odd}(\text{pcs}(\overline{q}) \backslash \{i_1\})$$

2. We look at the allowed syntax of the input language $\mathcal{P}$ to the algorithm. Let $p \in \mathcal{P}$, then:

$$\begin{array}{rcl} p & ::= & p_{\mathrm{s}} \\ & \mid & p_{\mathrm{s}}; X_i \\ & \mid & p_{\mathrm{s}}; p \\ & \mid & p \mathbin{[\!]} p \end{array}$$

The occurrence of a recursion variable is always guarded in an input process term. An operand of alternative composition can have self recursion, only if it is a recursion scope operator.

Consider the following example of a recursion scope operator:

$$\begin{array}{l} [\![_{\mathrm{R}} \ \{ \quad X_1 \mapsto a;\ X_1, \\ \qquad\quad X_2 \mapsto b;\ X_2, \\ \qquad\quad X_3 \mapsto a;\ X_1 \mathbin{[\!]} b;\ X_2 \\ \quad\ \ \} \\ ::\ X_3 \\ ]\!] \end{array}$$

Despite recursion in the process definition of X_3, we use the algorithm without a new root to linearize its process definition. Although semantically, $X_3 \leftrightarrows X_1 \mathbin{[\!]} X_2$, but due to the syntactic difference, there is no loop to the initial states of X_1 and X_2. The initial options of X_3 are activated by different values of program counters than those for the initial options of X_1 and X_2. Thus a new root is always automatically created for the alternative composition due to guarded occurrence of X_2 and X_3.

To test for self recursion in a linear form, we look at all the alternatives of the linear process equation of a given linear form. In case one of the alternatives sets a program counter to the value given in the initialization predicate of the linear form, then the process term has self recursion.

Below we define a function TestforRec : $\widetilde{P} \to \mathcal{B}ool$ that checks for self recursion in a linear form:

$$\mathrm{TestforRec}(\widetilde{p}) = \begin{cases} \text{true} & \text{if } \exists b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X \in \mathrm{Alt}(\mathrm{rhs}(\widetilde{p})) \\ & \wedge \mathrm{matchvalue}(\mathrm{ap}_{\mathrm{pc}}, U_{\mathrm{pc}}(\widetilde{p})) \\ \text{false} & \text{otherwise} \end{cases}$$

where the function matchvalue, of type, $\mathcal{AP}_{\mathrm{pc}} \times \mathcal{U}_{\mathrm{pc}} \to \mathcal{B}ool$ takes an action predicate and an initialization predicate of a linear form. It returns true if the given action predicate is setting a program counter according to the value of the program counter in the given initialization predicate.

$$\begin{array}{rcl} \mathrm{matchvalue}(W_{\mathrm{pc}} : r_{\mathrm{pc}}, u_{\mathrm{pc}}) & = & \begin{cases} \text{true} & \text{if } \exists i_d \in W_{\mathrm{pc}} \wedge \\ & r_{\mathrm{pc}} \implies \\ & (i_d = \mathrm{value}(u_{\mathrm{pc}}, d)) \\ \text{false} & \text{otherwise} \end{cases} \\ \mathrm{matchvalue}((W_{\mathrm{pc}} : r_{\mathrm{pc}}, \mathrm{ap}_{\mathrm{pc}}), u_{\mathrm{pc}}) & = & \mathrm{matchvalue}(W_{\mathrm{pc}} : r_{\mathrm{pc}}, u_{\mathrm{pc}}) \\ & & \vee \mathrm{matchvalue}(\mathrm{ap}_{\mathrm{pc}}, u_{\mathrm{pc}}) \end{array}$$

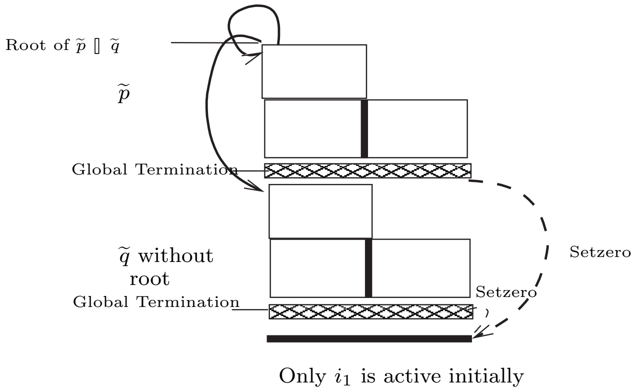

Root of $\widetilde{p}$ [] $\widetilde{q}$
$\widetilde{p}$
Global Termination
$\widetilde{q}$ without root
Global Termination
Setzero
Setzero
Only i_1 is active initially

Figure 4.9: Alternative Composition without a new root

where W_{pc} is a set of program counters, r_{pc} is a predicate on program counters and ap_{pc} is an action predicate on program counters.

Note that in the definition of TestforRec, we do not check the terminating alternatives of a process term nor the alternatives with pointers update(X_i). A terminating option does not have recursion. We know that an operand of alternative composition has self recursion only when it is a recursion scope operator. An alternative with a pointer update(X_i) appears in an intermediate form during the linearization of a process term of the form $p_{\text{s}};\ X_i$ (see Section 4.6.7). The pointers of the form *update*(X_i) are not present in the final linear form of a recursion scope operator.

While joining two process terms in alternative composition, as is done in sequential composition, see Section 4.6.4, the values of the program counters common in the linear forms of operands are made distinct from each other by incrementing the values of program counters in one of the operands. For linearizing $p \ [] \ q$, we can, without loss of generality, decide to increment the values of program counters in $\widetilde{p}$. In the algorithm for alternative composition without a new root, we increment the values of all program counters in one operand by the maximum value of the program counter i_1 in the other operand minus 2. The reason for doing this is that in this algorithm, the root (i.e. initial options) of $\widetilde{q}$ is moved (i.e. incremented) to the same level (i.e. value of program counter i_1) as the root of $\widetilde{p}$ after incrementing. This leaves behind a gap in the value of the program counter i_1 at the border of $\widetilde{p}$ and $\widetilde{q}$. (This is different from sequential composition, where there is no such gap in the values of program counter i_1). Incrementing the program counter i_1 in $\widetilde{p}$ by maximum value of i_1 in $\widetilde{q}$ minus 2, brings the alternative guarded by predicate $i_1 = 2$ in $\widetilde{p}$ to the same level as the initial options of $\widetilde{q}$. Thus, after incrementing, the predicate $i_1 = 2$ in $\widetilde{p}$ becomes equivalent to $i_1 = \text{value}(u^q_{\text{pc}}, 1)$. The function value($u^q_{\text{pc}}, 1$) returns the initial value of i_1 in the linear form of q. This value will not be used to guard the initial option of q in the linear form of $p \ [] \ q$, because the root of q has to be moved to the same level as that of p. Therefore, no overlap of program counter values guarding the options of $\widetilde{p}$ and $\widetilde{q}$ occurs.

$$
\begin{array}{l}
\text{Normalize}(p \,[\!]\, q) = \widetilde{r} = \\
[\![_{\mathrm{V}}\, \sigma^r_{\mathrm{pc}} \cup \sigma_p \cup \sigma_q, C_p \cup C_q, L_p \cup L_q \\
::\ [\![_{\mathrm{R}}\ \{X \mapsto \text{Setzero}\ \ (\ \ \text{Incrpcs}^{>1}(\overline{p}, \text{value}(u^q_{\mathrm{pc}}, 1) - 2) \\
\qquad\qquad\qquad\qquad\qquad\quad [\!]\ \text{IncrInitialpcs}(\widetilde{q}, \text{value}(u^p_{\mathrm{pc}}, 1) - 2) \\
\qquad\qquad\qquad\qquad\quad) \\
\qquad\quad \} \\
\quad\ ::\ u_p \wedge u_q \wedge u^r_{\mathrm{pc}} \curvearrowright X \\
\quad\ ]\!] \\
]\!],
\end{array}
$$

where,

1. The valuation σ^r_{pc} defining program counters is given below:

   $$\sigma^r_{\mathrm{pc}} \quad = \quad \{i_1 \mapsto \bot, \ldots, i_{\max(\text{Count}(\overline{p}), \text{Count}(\overline{q}))} \mapsto \bot\}$$

   The total number of program counters in the alternative composition is the maximum of the numbers of program counters in the two operands.

2. The initialization predicate u^r_{pc} initializing the program counters is as follows:

   $$
   \begin{array}{l}
   u^r_{\mathrm{pc}} = (i_1 = \text{value}(u^p_{\mathrm{pc}}, 1) + \text{value}(u^q_{\mathrm{pc}}, 1) - 2) \wedge \\
   \quad \bigwedge_{1 < k \leq \max(\text{Count}(\overline{p}), \text{Count}(\overline{q}))} i_k = \text{value}(u^p_{\mathrm{pc}}, 1) + \text{value}(u^q_{\mathrm{pc}}, 1) - 1
   \end{array}
   $$

   Initially only program counter i_1 is active.

3. The function $\text{Incrpcs}^{>1}$: (predicate × $\mathbb{N}$ → predicate) $\cup$ $(\overline{\mathcal{P}} \times \mathbb{N} \rightarrow \overline{\mathcal{P}}) \cup (\widetilde{P} \times \mathbb{N} \rightarrow \widetilde{P})$ takes a predicate or a process term as the first parameter, and a natural number as the second parameter. It increments the values (greater than 1) assigned to the program counters in the given predicate or the given process term by the given number.

   $$\text{Incrpcs}^{>1}(x, n) \quad = \quad x[i_k = c_k + n / i_k = c_k]_{k \in \mathbb{N}, c_k > 1}$$

   where $c_k > 1$ for all values of k. The zero and 1 values of program counters are not incremented, as they indicate the (final) terminating actions of an operand.

   In a linear form $\widetilde{p}$, program counters are assigned values in the initialization predicate $U_{\mathrm{pc}}(\widetilde{p})$ and in the right hand side of the recursion definition of $\widetilde{p}$, i.e. $\text{rhs}(\widetilde{p})$. (See Section 4.6.1 for the definition of rhs). $\text{Incrpcs}^{>1}(\widetilde{p}, n)$ increments the non zero values assigned to program counters in both these constructs of $\widetilde{p}$.

4. The function IncrInitialpcs : $\widetilde{P} \times \mathbb{N} \rightarrow \overline{\mathcal{P}}$ takes a process term of the form $\widetilde{p}$ and a natural number. It increments the initial value of i_1 in the right hand side of the recursion definition of $\widetilde{p}$ by the number given.

   $$
   \begin{array}{lll}
   \text{IncrInitialpcs}(\widetilde{p}, n) = \text{rhs}(\widetilde{p}) & [& i_1 = \text{value}(U_{\mathrm{pc}}(\widetilde{p}), 1) + n \\
   & / & i_1 = \text{value}(U_{\mathrm{pc}}(\widetilde{p}), 1) \\
   &] &
   \end{array}
   $$

   (See Section 4.6.1 for the definition of rhs).

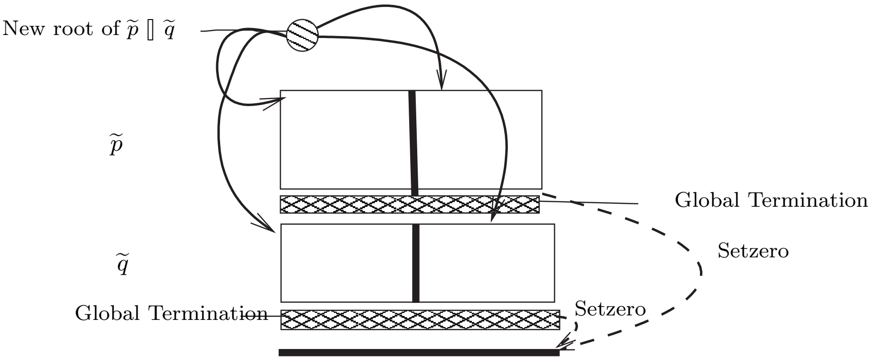


Figure 4.10: Alternative Composition with a new root

This algorithm is used for linearizing alternative compositions with operands lacking self recursion. As self recursion is excluded, therefore an initially active program counter, particularly i_1 (when there is no initial parallelism), will never be reset back to its initial value. This means that the initial value of i_1 only occurs in the guards of the operand process terms. We make use of this fact in the definition of the function IncrIntialpcs and increment all occurrences of the initial value of i_1.

The function IncrIntialpcs makes available the initial options of $\widetilde{q}$ by setting the value of i_1 in $\widetilde{q}$ to the initial value of i_1 in Normalize($p \mathbin{[\!]} q$).

Alternative Composition with a new root

This algorithm can be used to alternatively compose linear process terms with initial parallelism and self recursion. We create a new root for the alternative composition. Initially we only activate program counter i_1 in the alternative composition. The initial parallelism in the operands, if present, is captured by options guarded by i_1. In case of parallelism, more than one option of an operand is initially available. For each initial option in the given operands p and q, an option guarded by $i_1 = \text{value}(u^p_{\text{pc}}, 1) + \text{value}(u^q_{\text{pc}}, 1) + 2$, is added in the alternative composition. Hence a new root is created by a new value for the program counter i_1 which is equal to the sum of its maximum values in $\widetilde{p}$ and $\widetilde{q}$ plus 2.

$$
\begin{array}{l}
\text{Normalize}(p \mathbin{[\!]} q) = \widetilde{r} = \\
\;\llbracket_{\text{V}}\, \sigma^r_{\text{pc}} \cup \sigma_p \cup \sigma_q, C_p \cup C_q, L_p \cup L_q \\
\;::\; \llbracket_{\text{R}}\; \{X \mapsto \text{Setzero} \quad (\quad \text{Incrpcs}^{>1}(\overline{p}, \text{value}(u^q_{\text{pc}}, 1)) \\
\qquad\qquad\qquad\qquad\qquad\qquad\; \mathbin{[\!]}\; \overline{q} \\
\qquad\qquad\qquad\qquad\qquad\qquad\; \mathbin{[\!]}\; \text{Createnewroot}(m_{u_{pq}}, \text{Incrpcs}^{>1}(\widetilde{p}, \text{value}(u^q_{\text{pc}}, 1))) \\
\qquad\qquad\qquad\qquad\qquad\qquad\; \mathbin{[\!]}\; \text{Createnewroot}(m_{u_{pq}}, \widetilde{q}) \\
\qquad\qquad\qquad\qquad\qquad\;) \\
\qquad\qquad \} \\
\qquad :: u_p \wedge u_q \wedge u^r_{\text{pc}} \curvearrowright X \\
\qquad \rrbracket \\
\;\rrbracket,
\end{array}
$$

where,

1. The notation $m_{u_{pq}}$ is an abbreviation for $\mathrm{value}(u^p_{\mathrm{pc}}, 1) + \mathrm{value}(u^q_{\mathrm{pc}}, 1) + 2$.

2. The valuation σ^r_{pc} defining program counters is given below:

$$\sigma^r_{\mathrm{pc}} \quad = \quad \{i_1 \mapsto \bot, \ldots, i_{\max(\mathrm{Count}(\overline{p}),\mathrm{Count}(\overline{q}))} \mapsto \bot\}$$

   The total number of program counters in the alternative composition is the maximum of the numbers of program counters in the two operands.

3. The initialization predicate u^r_{pc} initializing the program counters is as follows:

$$\begin{aligned} u^r_{\mathrm{pc}} \quad = \quad & (i_1 = \mathrm{value}(u^p_{\mathrm{pc}}, 1) + \mathrm{value}(u^q_{\mathrm{pc}}, 1) + 2) \wedge \\ & \textstyle\bigwedge_{1<k\leq\max(\mathrm{Count}(\overline{p}),\mathrm{Count}(\overline{q}))} \\ & \qquad\qquad (i_k = \mathrm{value}(u^p_{\mathrm{pc}}, 1) + \mathrm{value}(u^q_{\mathrm{pc}}, 1) + 1) \end{aligned}$$

   Except for i_1, all program counters are set to odd values.

4. The function Createnewroot : $\mathbb{N} \times \widetilde{P} \to \overline{\mathcal{P}}$ returns part of the new root that is created for the alternative composition. The first parameter of Createnewroot is $m_{u_{pq}}$, the sum of the initial values of i_1 in $\widetilde{p}$ and in $\widetilde{q}$, +2. The second parameter is the linear form of one of the operands p or q. This function makes available the initial options of the given operand in the linear form of $p \mathbin{[]} q$. This is done by taking all the initially active options in the given linear process term and replacing their guard predicates by guards setting i_1 to $m_{u_{pq}}$. After performing the first action, the program counters are initialized according to the initialization predicate of the given operand.

$$\begin{aligned} & \mathrm{Createnewroot}(m, \widetilde{p}) = \\ & [] \quad \{ \quad (i_1 = m) \to p_{\mathrm{u}} \mid b_{\mathrm{pc}} \to p_{\mathrm{u}} \in \mathrm{Alt}(\overline{p}) \wedge U_{\mathrm{pc}}(\widetilde{p}) \implies b_{\mathrm{pc}}\} \\ & [] \quad \{ \quad (i_1 = m) \to p_{\mathrm{act}}, \mathrm{update}(X_i);\ X \\ & \qquad \mid \quad b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{update}(X_i);\ X \in \mathrm{Alt}(\overline{p}) \wedge U_{\mathrm{pc}}(\widetilde{p}) \implies b_{\mathrm{pc}} \\ & \qquad \} \\ & [] \quad \{ \quad (i_1 = m) \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}}, (\mathrm{pcs}(\mathrm{rhs}(\widetilde{p}))\backslash\mathrm{pcs}(\mathrm{ap}_{\mathrm{pc}})) : \\ & \qquad\quad\;\; \textstyle\bigwedge_{i_d\in(\mathrm{pcs}(\mathrm{rhs}(\widetilde{p}))\backslash\mathrm{pcs}(\mathrm{ap}_{\mathrm{pc}}))} i_d = \mathrm{value}(U_{\mathrm{pc}}(\widetilde{p}), d);\ X \\ & \qquad \mid \quad b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}};\ X \in \mathrm{Alt}(\overline{p}) \wedge U_{\mathrm{pc}}(\widetilde{p}) \implies b_{\mathrm{pc}} \\ & \qquad \} \\ & [] \quad \{ \quad (i_1 = m) \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \mid b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \in \mathrm{Alt}(\overline{p}) \wedge \\ & \qquad\quad\;\; U_{\mathrm{pc}}(\widetilde{p}) \implies b_{\mathrm{pc}} \\ & \qquad \} \end{aligned}$$

   In an alternative composition, after doing an action of one alternative, it is not possible to do an action of the other alternative. Therefore, after performing the first action, program counters are reset according to the initialization predicate of the given operand. This is being done in the second alternative of Createnewroot.

4.6.7 Recursion Scope operator

In the input language to the algorithm, recursion variables are only allowed in a recursion scope operator. Only complete recursion definitions are allowed. i.e. the top-level process term of a recursion scope, as well as any process definition of a recursion variable may not mention any recursion variable not defined within the same scope.

We mentioned in Section 4.4 that the pointer update(X_i) appears only in the intermediate linear form during linearization of a recursion scope operator. Recall from Section 4.3 the allowed syntax for process definitions of recursion variables. The set of process definitions P for recursion variables, with $p \in P$, is defined as follows:

$$\begin{array}{lll} p & ::= & p_{\mathrm{s}} \\ & \mid & p_{\mathrm{s}}; X_i \\ & \mid & p_{\mathrm{s}}; p \\ & \mid & p \mathbin{[]} p \end{array}$$

In sections 4.6.2 to 4.6.6, we have defined how to linearize different process terms from the set $\mathcal{P}$. The linearization of a process term of the form $p_{\mathrm{s}};\ X_i$ was not defined. It is defined later in this section. The pointer update(X_i) is introduced during the linearization of a process definition of the form $p_{\mathrm{s}};\ X_i$. When linearizing such a process definition, we come across recursion variables names whose linear forms we may not know yet. For example consider the following process term:

$$\begin{array}{l} [\![_{\mathrm{R}}\, \{X_1 \mapsto p_{\mathrm{s}};\ X_2, X_2 \mapsto q_{\mathrm{s}};\ X_3, X_3 \mapsto r_{\mathrm{s}};\ X_1\} \\ ::\, X_1 \\ ]\!] \end{array}$$

In linearization of such a recursion scope, as we may not yet know the linear form of the recursion variable being referred to, we place a pointer update(X_i) in the terminating options of the linear form of the process term referring to the recursion variable X_i. The pointer update(X_i) is later replaced by some action predicates according to the linear form of the process definition of variable X_i, after all recursion variables in a given recursion scope operator have been partially linearized.

Recall that the restricted form of the recursion scope operator process term is defined by:

$$\begin{array}{llll} p_{\mathrm{R}} & ::= & [\![_{\mathrm{R}}\, R :: X_i\ ]\!] & \mathrm{Complete}(R) \wedge X_i \in \mathrm{dom}\, R \\ & \mid & [\![_{\mathrm{R}}\, R :: p\ ]\!] & \mathrm{Complete}(R) \wedge \mathrm{Recvars}(p) \in \mathrm{dom}\, R \end{array}$$

In this section we give a linearization algorithm for a recursion scope of the form $[\![_{\mathrm{R}}\, R :: X_i\ ]\!]$ only. A recursion scope operator of the form $[\![_{\mathrm{R}}\, R :: p\ ]\!]$ can always be transformed into the recursion scope of the form $[\![_{\mathrm{R}}\, R :: X_i\ ]\!]$ as follows:

- Introduce a new recursion variable in R and define its right hand side to be equal to p;
- rewrite the recursion scope by adding the new definition to R and replacing p by the new recursion variable

If a recursion scope operator of the form $[\![_{\mathrm{R}}\, R :: p\ ]\!]$ is given in the input to the algorithm, we first transform it and then linearize it. Therefore in the linearization algorithm,

$$\mathrm{Normalize}([\![_{\mathrm{R}}\, R :: p\ ]\!]) = \mathrm{Normalize}([\![_{\mathrm{R}}\, R \cup \{X_{\mid\ \mathrm{dom}\, R\ \mid+1} \mapsto p\} :: X_{\mid\ \mathrm{dom}\, R\ \mid+1}\ ]\!]),$$

where $\mid\ \mathrm{dom}\, R\ \mid$ denotes the number of recursion variables in $\mathrm{dom}\, R$.

A restriction on process definitions of recursion variables

Consider the following recursion definition.

$$\{X_1 \mapsto \|[_{\mathrm{V}}\, \sigma, C, L :: p\,]\|\ ;\ X_1\}$$

We can view X_1 as an infinite sequence of variable scopes $\|[_{\mathrm{V}}\, \sigma, C, L :: p\,]\|$, with each scope having a new instance of local variables $\mathrm{dom}(\sigma), C$ and L. In the linear form, (see section 4.4), a variable scope is only present at the top-level. All local variable definitions and valuations of any variable scopes present in the input to the linearization algorithm are moved to the top-level. It is not possible to linearize a process definition as that of X_1 in the current linearization algorithm since it requires infinitely many instances of local variables. See Section 4.6.12 for a detailed discussion of the problem. We disallow in the input to the algorithm, recursion variables with definitions consisting of a variable scope operator followed by self recursion.

Linearization of $p_{\mathrm{s}}; X_i$

Assume

$$\begin{array}{rcl} \mathrm{Normalize}(p_{\mathrm{s}}) = \widetilde{p} & = & \|[_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma_{\perp}, C, L \\ & & ::\ \ \|[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\,]\| \\ & &]\| \end{array}$$

The number of program counters and the initialization predicates of the linear form of $p_s;\ X_i$, for some recursion variable X_i, are the same as for the linear form of p_{s}. The terminating options of p_{s} are modified to include a pointer to the recursion variable X_i.

$$\begin{array}{rcl} \mathrm{Normalize}(p_{\mathrm{s}};\ X_i) & = & \|[_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L \\ & & ::\ \ \|[_{\mathrm{R}}\, \{X \mapsto \mathrm{RFSequent}(\overline{p}, X_i)\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\,]\| \\ & &]\|, \end{array}$$

where the function $\mathrm{RFSequent} : \overline{\mathcal{P}} \times \mathcal{X} \to \overline{\mathcal{P}}_{\mathrm{R}}$ takes a process term of the form $\overline{p}$ and a recursion variable. It removes the action predicate updating program counters from the terminating options of $\overline{p}$ and appends them by a pointer $\mathrm{update}(X_i)$ and a recursive call to X.

$$\begin{array}{rcl} \mathrm{RFSequent}(\overline{p}, X_i) & = & \mathrm{Term}(\overline{p})[b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{update}(X_i);\ X \\ & & \qquad\qquad /b_{\mathrm{pc}} \to p_{\mathrm{act}}, \mathrm{ap}_{\mathrm{pc}} \\ & & \qquad\qquad] \\ & & [\!]\ \mathrm{Nonterm}(\overline{p}) \end{array}$$

Linearization of $\|[_{\mathrm{R}}\, \{X_1 \mapsto p_1, \ldots, X_n \mapsto p_n\} :: X_m\,]\|$

In the linearization of a recursion scope operator, we reuse program counters in the linear form of process definitions of recursion variables. The total number of program counters is equal to the highest number of program counters used in any process definition of a recursion variable. Since recursion variables can only appear at the end of a process definition, therefore updating counters is easy.

We follow the following steps in the linearization of a recursion scope operator:

1. Linearize the right hand sides of the definitions of all recursion variables;

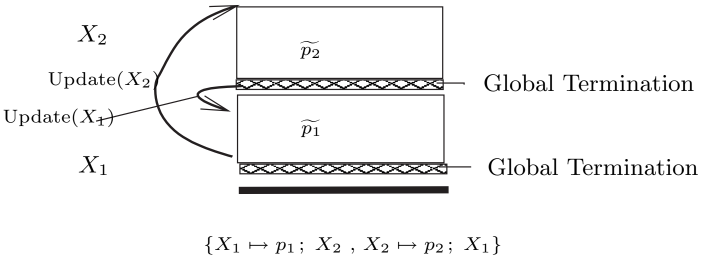


Figure 4.11: A set of recursion definitions

2. Make the values of program counters that are common among the linear forms of recursion variables distinct from each other. This is done by incrementing the values of all program counters in the linear form of a recursion variable X_j, with $j > 1$, by the sum of maximum values of i_1 in linear forms of X_1 to X_{j-1}. The linear form of X_1 is not incremented;

3. Replace the expressions update(X_i) by action predicates setting program counters and model variables according to the initialization predicates of the linear form of X_i;

4. Alternatively compose the linear forms of the recursion definitions of all recursion variables;

5. Finally set all the program counters to zero in the terminating options of the alternative composition thus obtained.

Assume

$$\begin{array}{rcl}
\text{Normalize}(p_1) = \widetilde{p}_1 & = & [\![_{\mathrm{V}}\ \sigma^1_{\mathrm{pc}} \cup \sigma_1, C_1, L_1 \\
& & ::\ \ [\![_{\mathrm{R}}\ \{X \mapsto \overline{p}_1\} :: u_1 \wedge u^1_{\mathrm{pc}} \curvearrowright X\ ]\!] \\
& &]\!] \\
& & \quad\dots \\
\text{Normalize}(p_n) = \widetilde{p}_n & = & [\![_{\mathrm{V}}\ \sigma^n_{\mathrm{pc}} \cup \sigma_n, C_n, L_n \\
& & ::\ \ [\![_{\mathrm{R}}\ \{X \mapsto \overline{p}_n\} :: u_n \wedge u^n_{\mathrm{pc}} \curvearrowright X\ ]\!] \\
& &]\!]
\end{array}$$

Then,

$$\begin{array}{l}
\text{Normalize}([\![_{\mathrm{R}}\ \{X_1 \mapsto p_1, \dots X_n \mapsto p_n\} :: X_m\ ]\!]) = \\
[\![_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \bigcup_{1 \le j \le n} \sigma_j, \bigcup_{1 \le j \le n} C_j, \bigcup_{1 \le j \le n} L_j \\
::\ \ [\![_{\mathrm{R}}\ \{X \mapsto \text{Setzero}(\ [\!]\ _{1 \le j \le n}\text{Update}(j, [\widetilde{p}_1, \dots, \widetilde{p}_n]))\} \\
\qquad :: u_m \wedge u_{\mathrm{pc}} \curvearrowright X \\
\qquad]\!] \\
]\!],
\end{array}$$

where

1. $1 \le m \le n$

2. Let maxpc denote the highest number of program counters used in any of the linear forms, $\widetilde{p}_1 \dots \widetilde{p}_n$. Then,

$$\text{maxpc} = \max(\bigcup_{1 \leq j \leq n} \{\text{Count}(\overline{\mathrm{p}}_{\mathrm{j}})\})$$

The valuation σ_{pc} defining the program counters is as follows:

$$\sigma_{\text{pc}} \quad = \quad \{i_1 \mapsto \bot, \dots, i_{\text{maxpc}} \mapsto \bot\}$$

3. The initialization predicate u_{pc} initializing the program counters is as follows:

$$\begin{array}{rcl} u_{\text{pc}} & = & \text{Incrpcs}^{>1}(u^m_{\text{pc}}, \text{Incrvalue}(m, [\widetilde{p}_1, \dots, \widetilde{p}_n])) \wedge \\ & & \bigwedge_{d \in [1,\dots,\text{maxpc}] \setminus [1,\dots,\text{Count}(p_m)]} \\ & & i_d = \text{value}(\text{Incrpcs}^{>1}(u^n_{\text{pc}}, \text{Incrvalue}(n, [\widetilde{p}_1, \dots, \widetilde{p}_n])), 1) + 1 \end{array}$$

where the function Incrvalue : $\mathbb{N} \times \widetilde{P}^* \rightarrow \mathbb{N}$ takes a natural number and a list of linear forms. The given natural number must be an index of the given list. The function Incrvalue returns the sum of maximum values of the program counter i_1 in the linear forms appearing before the given index in the given list. If the given index points to the first element of the list, then the function Incrvalue returns 0.

$$\begin{array}{lcl} \text{Incrvalue}(1, L) & = & 0 \\ \text{Incrvalue}(j, L)_{j>1} & = & \Sigma^{j-1}_{k=1} \text{value}(U_{\text{pc}}(L.k), 1), \end{array}$$

where the notation $L.k$ denotes the k^{th} element of the list L.

We give a mathematical definition of the function Incrvalue. For implementation purpose, a recursive definition of the function can be adopted.

The initialization predicate of a recursion scope operator is the initialization predicate of the linear form of the initial recursion variable, after incrementing the program counters in the predicate by the sum of maximum values of i_1 in linear forms of X_1 until X_{m-1}.

The program counters that are not used in the linear form $\widetilde{p}_m$, are set to an odd value which is equal to the highest value of program counter i_1 +1. The highest value of program counter i_1 is used in the linear form of the recursion variable X_i with the highest index i. It is given by the expression, value(Incrpcs$^{>1}$(u^n_{pc}, Incrvalue($n, [\widetilde{p}_1, \dots, \widetilde{p}_n]$)), 1).

4. The function Update : $\mathbb{N} \times \widetilde{P}^* \rightarrow \overline{\mathcal{P}}$ takes a natural number and a list of linear process terms. The natural number must point to an element of the list, which consists of linear forms of recursion variables. The function Update(j, L) increments the non zero values of all program counters in the j^{th} element of L, by the increment value Incrvalue(j, L) and replaces any pointers update(X_i) by appropriate action predicates, where $i, j \in [1, \mid L \mid]$. Thus the function Update covers two steps i.e. incrementing process definitions of recursion variables and replacing a pointer update(X_i) by the required action predicates in the linearization procedure.

A pointer update(X_i) in an alternative of rhs($L.j$) is replaced by the following action predicates:

(a) An action predicate setting the local environment variables of $L.i$ according to its initialization predicate u_i. The jump set of this action predicate consists of the local discrete and continuous variables of $L.i$;

(b) An action predicate initializing the program counters according to their initial values in $L.i$, after incrementing the initial values by a factor $\mathrm{Incrvalue}(i, L)$; and

(c) In case the number of program counters of $L.j$ is greater than the number of program counters in $L.i$, the unused (extra) program counters of $L.j$ are deactivated.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathrm{Update}(j, L) =} \\
\multicolumn{2}{l}{\mathrm{Incrpcs}^{>1}(\mathrm{rhs}(L.j), \mathrm{Incrvalue}(j, L))} \\
[& \{v \mid v \in \mathrm{dom}(\mathrm{Sigma}(L.i))\} : U(L.i), \\
 & \mathrm{pcs}(\mathrm{rhs}(L.i)) : \mathrm{Incrpcs}^{>1}(U_{\mathrm{pc}}(L.i), \mathrm{Incrvalue}(i, L)), \\
 & \mathrm{pcs}(\mathrm{rhs}(L.j)) \backslash \mathrm{pcs}(\mathrm{rhs}(L.i)) : \\
 & \bigwedge_{i_d \in \mathrm{pcs}(\mathrm{rhs}(L.j)) \backslash \mathrm{pcs}(\mathrm{rhs}(L.i))} \\
 & i_d = \mathrm{value}(\mathrm{Incrpcs}^{>1}(U_{\mathrm{pc}}(L.\mathrm{len}(L)), \mathrm{Incrvalue}(\mathrm{len}(L), L)), 1) + 1 \\
/ & \mathrm{update}(X_i) \\
], &
\end{array}
$$

where $\mathrm{len}(L)$ denotes the length of the list L. Inactive program counters are set to an odd value equal to the highest value of i_1 in any of the linear forms in the list L plus one. The linear form with the highest value of i_1 is the last element of the list. Therefore extra program counters are set to: $\mathrm{value}(\mathrm{Incrpcs}^{>1}(U_{\mathrm{pc}}(L.\mathrm{len}(L)), \mathrm{Incrvalue}(\mathrm{len}(L), L)), 1) + 1$.

4.6.8 Initialization

An initialization predicate u is a predicate on the model variables. A process term p with an initialization predicate u, denoted by $u \curvearrowright p$, behaves as p whenever the initialization predicate u holds. The linear form of $u \curvearrowright p$ is calculated by concatenating u with the initialization predicate initializing the model variables in the linear form of p.

Assume

$$
\begin{array}{rcl}
\mathrm{Normalize}(p_{\mathrm{s}}) = \widetilde{p} & = & [\![_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L \\
 & & ::\ [\![_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\ ]\!] \\
 & &]\!]
\end{array}
$$

Then,

$$
\begin{array}{rcl}
\mathrm{Normalize}(u \curvearrowright p_{\mathrm{s}}) & = & [\![_{\mathrm{V}}\, \sigma_{\mathrm{pc}} \cup \sigma, C, L \\
 & & ::\ [\![_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u_p \wedge u_{\mathrm{pc}} \curvearrowright X\ ]\!] \\
 & &]\!]
\end{array}
$$

The linear form of $u \curvearrowright p_{\mathrm{s}}$ has been derived from the following property of the semantics of Hybrid χ.

$$u \curvearrowright (u' \curvearrowright p) \leftrightarrows u \wedge u' \curvearrowright (p)$$

4.6.9 Encapsulation

Two kinds of encapsulation operators are allowed in the input language:

1. $\partial_A(p)$ denotes the encapsulation of non communicating actions in the set A. Actions from the set A are blocked.

2. $\partial_H(p)$ denotes the encapsulation of send and receive actions on channels in the set H. $\partial_H(p)$ is defined as:
$$\partial_H(p) \triangleq \partial_{\{h\,!\,\mathrm{cs},h\,?\,\mathrm{cs} \mid h\in H,\mathrm{cs}\in\Lambda^*\}}(p)$$
In a channel encapsulation operator, $\partial_H(p)$, the blocking of send and receive actions is done on the basis of the names of the communication channels and not on the basis of the values sent or received.

Assume
$$\begin{array}{rcl}\mathrm{Normalize}(p_{\mathrm{s}}) = \widetilde{p} & = & [\![_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \sigma, C, L \\ & & ::\ [\![_{\mathrm{R}}\ \{X \mapsto \overline{p}\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\ ]\!] \\ & &]\!]\end{array}$$

Then,
$$\begin{array}{rcl}\mathrm{Normalize}(\partial_L(p_{\mathrm{s}})) & = & [\![_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \sigma, C, L \\ & & ::\ [\![_{\mathrm{R}}\ \{X \mapsto \mathrm{Encaps}(\overline{p}, L)\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\ ]\!] \\ & &]\!]\end{array}$$

where the function Encaps : $\overline{\mathcal{P}} \times (2^{\mathcal{A}} \cup 2^{\mathcal{H}}) \rightarrow \overline{\mathcal{P}}$ takes as the first argument a process term of the form $\overline{p}$, and as the second argument, a set of action labels from the set A_{label} or a set of channels . It scans the given process term for any actions from the given set of action labels or send or receive actions on a channel in the given set of channels. If such an action is present in any alternative of the given process term $\overline{p}$, then that action is replaced by inv true. Any action predicates, pointers or recursive call to X following such an action are removed.

The function Encaps is defined below.

1. $\mathrm{Encaps}(b_{\mathrm{pc}} \rightarrow p_{\mathrm{u}}, L) = b_{\mathrm{pc}} \rightarrow p_{\mathrm{u}}$
2. In case $\mathrm{label}(p_{\mathrm{atom}}) \in L \vee (\mathrm{Ch}(p_{\mathrm{atom}}) \in L \wedge \mathrm{label}(p_{\mathrm{atom}}) \neq \mathrm{Ch}(p_{\mathrm{atom}})\,!?\,cs)$, then
$$\begin{array}{rcl}\mathrm{Encaps}((b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{update}(X_i);\ X), L) & = & b_{\mathrm{pc}} \rightarrow \mathrm{inv\ \ true} \\ \mathrm{Encaps}((b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}}), L) & = & b_{\mathrm{pc}} \rightarrow \mathrm{inv\ \ true} \\ \mathrm{Encaps}((b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}};\ X), L) & = & b_{\mathrm{pc}} \rightarrow \mathrm{inv\ \ true}\end{array}$$
3. In case $\mathrm{label}(p_{\mathrm{atom}}) \notin L \wedge \mathrm{Ch}(p_{\mathrm{atom}}) \notin L$, then
$$\begin{array}{l}\mathrm{Encaps}((b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{update}(X_i);\ X), L) \\ \qquad\qquad\qquad = b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{update}(X_i);\ X \\ \mathrm{Encaps}((b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}}), L) \\ \qquad\qquad\qquad = b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}} \\ \mathrm{Encaps}((b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}};\ X), L) \\ \qquad\qquad\qquad = b_{\mathrm{pc}} \rightarrow p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}};\ X\end{array}$$
4. $\mathrm{Encaps}(\overline{p} \mathbin{[\!]} \overline{q}, L) = \mathrm{Encaps}(\overline{p}, L) \mathbin{[\!]} \mathrm{Encaps}(\overline{q}, L)$

where

1. The function label : $\mathcal{P}_{\mathrm{atom}} \rightarrow \mathrm{A}_{\mathrm{label}} \cup A_{\mathrm{com}}$ takes an atomic action and returns its label.
$$\begin{array}{rcl}\mathrm{label}(l_{\mathrm{a}}, W : r) & = & l_{\mathrm{a}} \\ \mathrm{label}(h\,!\,\mathbf{e}_n, W : r) & = & h\,!\,\mathrm{cs} \\ \mathrm{label}(h\,?\,\mathbf{x}_n, W : r) & = & h\,?\,\mathrm{cs} \\ \mathrm{label}(h\,!?\,\mathbf{x}_n := \mathbf{e}_n, W : r) & = & h\,!?\,\mathrm{cs}\end{array}$$
cs denotes a list of values.

2. The function Ch : $\mathcal{P}_{\text{atom}} \to \mathcal{H} \cup \{\bot\}$ takes an atomic action. If the given action is a send, receive or communication action, it returns the channel of communication else it returns $\bot$.

$$\begin{array}{lcl}
\text{Ch}(l_{\text{a}}, W : r) & = & \bot \\
\text{Ch}(h\,!\,\mathbf{e}_n, W : r) & = & h \\
\text{Ch}(h\,?\,\mathbf{x}_n, W : r) & = & h \\
\text{Ch}(h\,!?\,\mathbf{x}_n := \mathbf{e}_n, W : r) & = & h
\end{array}$$

4.6.10 Channel Scope Operator

Localizing a channel in $[\![_{\text{H}}\, H :: p_{\text{s}}\,]\!]$ has the following effects on the process term p_{s}:

1. It makes communication on a local channel invisible to outside observers. An external observer only observes the silent action τ when communication on a local channel takes place.

2. Send and receive actions on local channels are no longer possible. Only the synchronous execution of a send and receive action resulting in communication is allowed on a local channel.

Assume

$$\begin{array}{rcl}
\text{Normalize}(p_{\text{s}}) = \widetilde{p} & = & [\![_{\text{V}}\, \sigma_{\text{pc}} \cup \sigma, C, L \\
& & ::\ [\![_{\text{R}}\, \{X \mapsto \overline{p}\} :: u_p \wedge u_{\text{pc}} \curvearrowright X\,]\!] \\
& &]\!]
\end{array}$$

Then

$$\begin{array}{rcl}
\text{Normalize}([\![_{\text{H}}\, H :: p_{\text{s}}\,]\!]) & = & [\![_{\text{V}}\, \sigma_{\text{pc}} \cup \sigma, C, L \\
& & ::\ [\![_{\text{R}}\, \{X \mapsto \text{localCh}(\overline{p}, H)\} :: u_p \wedge u_{\text{pc}} \curvearrowright X\,]\!] \\
& &]\!],
\end{array}$$

where the function localCh : $\overline{\mathcal{P}} \times 2^{\mathcal{H}} \to \overline{\mathcal{P}}$ takes a process term of the form $\overline{p}$ and a set of channels that are to be made local to $\overline{p}$. It does the following:

1. It replaces the communication action $h\,!?\,\mathbf{x}_n := \mathbf{e}_n$ with h in the given set by the silent action τ.

2. It replaces the send and receive actions on a cannel in the given set by inv true and removes any action predicates and recursive call to X following such a send or receive action.

It is defined below:

1. $\text{localCh}(b_{\text{pc}} \to p_{\text{u}}, H) = b_{\text{pc}} \to p_{\text{u}}$

2. In case $\text{Ch}(p_{\text{atom}}) \notin H$, then,

$$\begin{array}{l}
\text{localCh}((b_{\text{pc}} \to p_{\text{atom}}, \text{ap}, \text{update}(X_i);\ X), H) \\
\qquad\qquad = b_{\text{pc}} \to p_{\text{atom}}, \text{ap}, \text{update}(X_i);\ X \\
\text{localCh}((b_{\text{pc}} \to p_{\text{atom}}, \text{ap}, \text{ap}_{\text{pc}}), H) \\
\qquad\qquad = b_{\text{pc}} \to p_{\text{atom}}, \text{ap}, \text{ap}_{\text{pc}} \\
\text{localCh}((b_{\text{pc}} \to p_{\text{atom}}, \text{ap}, \text{ap}_{\text{pc}};\ X), H) \\
\qquad\qquad = b_{\text{pc}} \to p_{\text{atom}}, \text{ap}, \text{ap}_{\text{pc}};\ X
\end{array}$$

3. In case $\mathrm{Ch}(p_{\mathrm{atom}}) \in H$ and $\mathrm{label}(p_{\mathrm{atom}}) = h\,!?\,\mathrm{cs}$, where $cs \in \Lambda^*$, then

$$\begin{aligned}
&\mathrm{localCh}((b_{\mathrm{pc}} \to p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{update}(X_i);\ X), H)\\
&\qquad = b_{\mathrm{pc}} \to p_{\mathrm{atom}}[\tau/h\,!?\,\mathbf{x}_n := \mathbf{e}_n], \mathrm{ap}, \mathrm{update}(X_i);\ X\\
&\mathrm{localCh}((b_{\mathrm{pc}} \to p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}}), H)\\
&\qquad = b_{\mathrm{pc}} \to p_{\mathrm{atom}}[\tau/h\,!?\,\mathbf{x}_n := \mathbf{e}_n], \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}}\\
&\mathrm{localCh}((b_{\mathrm{pc}} \to p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}};\ X), H)\\
&\qquad = b_{\mathrm{pc}} \to p_{\mathrm{atom}}[\tau/h\,!?\,\mathbf{x}_n := \mathbf{e}_n], \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}};\ X
\end{aligned}$$

where n is any natural number ≥ 0.

4. In case, $\mathrm{Ch}(p_{\mathrm{atom}}) \in H$ and $\mathrm{label}(p_{\mathrm{atom}}) \neq \mathrm{Ch}(p_{\mathrm{atom}})\,!?\,\mathrm{cs}$, where $cs \in \Lambda^*$, then

$$\begin{aligned}
\mathrm{localCh}(b_{\mathrm{pc}} \to p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{update}(X_i);\ X, H) &= b_{\mathrm{pc}} \to \mathrm{inv\ \ true}\\
\mathrm{localCh}(b_{\mathrm{pc}} \to p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}}, H) &= b_{\mathrm{pc}} \to \mathrm{inv\ \ true}\\
\mathrm{localCh}(b_{\mathrm{pc}} \to p_{\mathrm{atom}}, \mathrm{ap}, \mathrm{ap}_{\mathrm{pc}};\ X, H) &= b_{\mathrm{pc}} \to \mathrm{inv\ \ true}
\end{aligned}$$

5. $\mathrm{localCh}(\overline{p} \mathbin{[\!]} \overline{q}, H) = \mathrm{localCh}(\overline{p}, H) \mathbin{[\!]} \mathrm{localCh}(\overline{q}, H)$

4.6.11 Urgent Channel Operator

Linearizing the urgent communication operator is allowed only for "well-posed systems". Definitions of well-posed dynamical systems found in literature [IS00, PHN03] imply existence and uniqueness of a solution for a given dynamical system. We give a different meaning to well-posed systems. By well-posed systems we mean systems that can be represented by *reactive automata* as defined in [BRSR07]. I.e., if a guard condition for a set of actions is true, then the invariants of the target locations (subprocesses following the guard conditions) to the given actions also hold. Thus invariants of target locations do not prevent actions from taking place immediately.

The urgent communication operator makes communication on a given set of channels urgent. In case communication on a channel $h \in H$ is possible for a process p, then the process $\upsilon_H(p)$ cannot do any time transitions.

Assume

$$\begin{aligned}
\mathrm{Normalize}(p_{\mathrm{s}}) = \widetilde{p} \;=\;& |[_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \sigma, C, L\\
&::\ |[_{\mathrm{R}}\ \{X \mapsto \overline{p}\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\ ]|\\
&]|
\end{aligned}$$

Then

$$\begin{aligned}
\mathrm{Normalize}(\upsilon_H(p_{\mathrm{s}})) \;=\;& |[_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \sigma, C, L\\
&::\ |[_{\mathrm{R}}\ \{X \mapsto \mathrm{Urgent}(\overline{p}, H)\} :: u_p \wedge u_{\mathrm{pc}} \curvearrowright X\ ]|\\
&]|
\end{aligned}$$

where the function Urgent : $\overline{\mathcal{P}} \times \mathcal{H} \to \overline{\mathcal{P}}$ takes a process term of the form $\overline{p}$ and a set of channels. It scans the process term $\overline{p}$ searching for alternatives containing communication actions $h\,!?\,cs$ with h in the given channel set and cs a list of values. If such an alternative is found, then the function Urgent adds another alternative with an urgency condition on the "guard" of the action predicate accompanying the communication action $h\,!?\,cs$. The new alternative is guarded by the same predicate b_{pc} as the found alternative. By "guard" of an action predicate $W : r$, we mean the part of predicate r that imposes conditions on values of model variables before an action takes place.

The function Precond : $(\mathcal{AP} \cup \mathcal{R}) \to \mathcal{R}$ takes an action predicate or predicate on model variables and returns the part of predicate imposing restrictions on previous values of model variables. For the syntax of allowed predicates see Section 4.3.

$$\begin{array}{lcl}
\text{Precond}(\text{true}) & = & \text{true} \\
\text{Precond}(\text{false}) & = & \text{false} \\
\text{Precond}(x^- \ \text{op}_\text{r} \ c) & = & x \ \text{op}_\text{r} \ c \\
\text{Precond}(x^+ \ \text{op}_\text{r} \ c) & = & \text{true} \\
\text{Precond}(r \wedge r') & = & \text{Precond}(r) \wedge \text{Precond}(r') \\
\text{Precond}(W : r) & = & \text{Precond}(r) \\
\text{Precond}(W : r, \text{ap}) & = & \text{Precond}(r) \wedge \text{Precond}(\text{ap})
\end{array}$$

where W is a subset of model variables, x is a model variable, c is a value and op_r denotes the set of relational operators.

The function Urgent is defined below:

1. $\text{Urgent}(b_\text{pc} \to p_\text{u}, H) = b_\text{pc} \to p_\text{u}$

2. In case $\text{Ch}(p_\text{atom}) \notin H$, then

$$\begin{array}{l}
\text{Urgent}((b_\text{pc} \to p_\text{atom}, \text{ap}, \text{update}(X_i);\ X), H) \\
\qquad\qquad = b_\text{pc} \to p_\text{atom}, \text{ap}, \text{update}(X_i);\ X \\
\text{Urgent}((b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc}), H) \\
\qquad\qquad = b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc} \\
\text{Urgent}((b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc};\ X), H) \\
\qquad\qquad = b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc};\ X
\end{array}$$

3. In case $\text{Ch}(p_\text{atom}) \in H$ and $\text{label}(p_\text{atom}) = h\,!?\,\text{cs}$, where $\text{cs} \in \Lambda^*$, then

$$\begin{array}{l}
\text{Urgent}((b_\text{pc} \to p_\text{atom}, \text{ap}, \text{update}(X_i);\ X), H) \\
\qquad\qquad = b_\text{pc} \to p_\text{atom}, \text{ap}, \text{update}(X_i);\ X \\
\qquad\qquad\quad [\!]\ b_\text{pc} \to \text{urg Precond}(\text{ap}) \\
\text{Urgent}((b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc}), H) \\
\qquad\qquad = b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc} \\
\qquad\qquad\quad [\!]\ b_\text{pc} \to \text{urg Precond}(\text{ap}) \\
\text{Urgent}((b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc};\ X), H) \\
\qquad\qquad = b_\text{pc} \to p_\text{atom}, \text{ap}, \text{ap}_\text{pc};\ X \\
\qquad\qquad\quad [\!]\ b_\text{pc} \to \text{urg Precond}(\text{ap})
\end{array}$$

4. $\text{Urgent}(\overline{p}\ [\!]\ \overline{q}, H) = \text{Urgent}(\overline{p}, H)\ [\!]\ \text{Urgent}(\overline{q}, H)$

4.6.12 Variable Scope Operator

Assume

$$\begin{array}{lcl}
\text{Normalize}(p_\text{s}) = \widetilde{p} & = & [\![_\text{V}\ \sigma^p_\text{pc} \cup \sigma_p, C_p, L_p \\
 & & ::\ \ [\![_\text{R}\ \{X \mapsto \overline{p}\} :: u_p \wedge u^p_\text{pc} \curvearrowright X\ ]\!] \\
 & & \ \ ]\!]
\end{array}$$

Then assuming that $\widetilde{p}$ and the variable scope operator $|[_{\mathrm{V}}\, \sigma_\perp, C, L :: p_{\mathrm{s}}\ ]|$ do not share any local environment variable names, i.e.

$$(L \cup \mathrm{dom}(\sigma_\perp)) \cap (\mathrm{dom}(\sigma_p) \cup L_p) \quad = \emptyset,$$

we define the linear form of the variable scope operator as follows:

$$\begin{array}{rcl} \mathrm{Normalize}(|[_{\mathrm{V}}\, \sigma_\perp, C, L :: p_{\mathrm{s}}\ ]|) & = & |[_{\mathrm{V}}\, \sigma^p_{\mathrm{pc}} \cup \sigma_p \cup \sigma'_\perp, C_p \cup C, L_p \cup L \\ & & ::\ \ |[_{\mathrm{R}}\, \{X \mapsto \overline{p}\} :: u \wedge u^p_{\mathrm{pc}} \curvearrowright X\ ]| \\ & &]| \end{array}$$

where,

1. The valuation $\sigma'_\perp$ is defined as follows:

$$\begin{array}{l} \mathrm{dom}(\sigma'_\perp) = \mathrm{dom}(\sigma_\perp) \\ \forall_{x \in \mathrm{dom}(\sigma'_\perp)} \sigma'_\perp(x) = \perp \end{array}$$

2. The initialization predicate u is defined as follows:

$$u = \left(\bigwedge_{x \in \mathrm{dom}(\sigma_\perp) \text{ s.t. } \sigma_\perp(x) \neq \perp} x = \sigma_\perp(x) \right) \wedge u_p$$

All model variables defined in the variable scope operator are initialized in the initialization predicate u. u_p includes the initialization of those variables that are defined in σ_p and are initialized to $\perp$.

In the output form of our linearization algorithm, we allow a variable scope only at the top most level. Hence during linearization, the local valuations and local variables of all variable scope operators present in a given input process term are moved to the top level. We assume that all variable scope operators of an input process term p_{s} have distinct local variables.

In a variable scope operator, the local variables can be initialized in two ways. They are initialized either in the local valuation or in an initialization predicate appearing at the start of the variable scope operator process term. In the output form of our linearization algorithm, see Section 4.4, only initialization predicates are allowed to initialize local variables. The top level valuation σ is undefined for all local variables. In the linear form of an input process term p containing a variable scope, the variables local to the variable scope must be set to their initial values at the same time as when they are initialized in p. To achieve this, we add an action predicate to the action preceding the variable scope in p. This action predicate allows the local variables of the variable scope operator to jump to their required initial values. This has been mentioned in the linearization of a sequential composition (Section 4.6.4) and recursion scope (Section 4.6.7.)

A problem may occur when linearizing a recursion variable with a process definition consisting of a variable scope operator and self recursion. This problem was pointed out in [The06].

Consider the following recursion definition:

$$X_1 \mapsto |[_{\mathrm{V}}\, \{n \mapsto \perp\}, \emptyset, \emptyset :: u \curvearrowright p_{\mathrm{s}}\ ]|\ ;\ X_1$$

Without linearization, we can view X_1 as an infinite sequence of variable scopes $[\![_{\mathrm{V}}\ \{n \mapsto \bot\}, \emptyset, \emptyset :: u \curvearrowright p_{\mathrm{s}}\ ]\!]$ with each member of the sequence having its own instance of the local variable n. In our linearization algorithm, where a variable scope is only allowed at the top level, this is not possible.

We would like to linearize the process definition of X_1 by adding an action predicate $\{n\} : u$ to the last action of p_{s}, where u is the initialization predicate of the variable scope. But situations may arise where appending the last action of p_{s} with $\{n\} : u$ results in a deadlock. For example,

$$X_1 \mapsto [\![_{\mathrm{V}}\ \{n \mapsto \bot\}, \emptyset, \emptyset :: (n = 0) \curvearrowright \tau, \{n\} : n = n^- + 1\ ]\!]\ ;\ X_1$$

The process definition of X_1 is capable of performing infinite number of τ actions. When we linearize X_1, then an action predicate $\{n\} : (n = 0)$ is appended to the last action of the variable scope. In this case, the last action of the variable scope, also updates n in its predicate. This leads to the predicate $n = n^- + 1 \wedge n = 0$ which equals false and therefore in the linear form of X_1, the action τ cannot take place. For this linearization algorithm, we simply disallow recursion variables with process definitions consisting of variable scopes and self recursion. Further research on this topic is left as a future work.

4.7 Size of the linear form

An algorithm for linearization of specifications in Hybrid χ 1.0 is already available (see [The06]). The linearization algorithm presented here is an improvement on the previous algorithm in terms of the size of the linear process term obtained. The linearization algorithm given in [The06] does not use any special data structures to model interleaving in parallel composition. The size of the linear form obtained from that algorithm can be exponential to the size of the input process term. For the linearization algorithm presented here, we claim that the size of the resulting linear form is of the order of the square of the size of the input process term. In this section, we give arguments to support this claim.

Let p be the input process term to our linearization algorithm. Let $\widetilde{p}$ denote the linear process term obtained by linearizing p. Below, we repeat the structure of a linearized process term $\widetilde{p}$ defined in Section 4.4.

$$\begin{array}{rcl} \widetilde{p} & ::= & [\![_{\mathrm{V}}\ \sigma_{\mathrm{pc}} \cup \sigma, C, L :: \\ & & \quad [\![_{\mathrm{R}}\ \{X \mapsto \overline{p}\} :: u \wedge u_{\mathrm{pc}} \curvearrowright X\ ]\!] \\ & &]\!] \end{array}$$

We calculate the size of $\widetilde{p}$ in terms of the number of alternatives present in its linear process equation $\bar{p}$ which is the right hand side of its recursion variable X.

For every atomic action or invariant (urgency) condition in the input process term p, there is at least one alternative in the linear process equation $\bar{p}$. In some cases, the number of alternatives in the linear process equation $\bar{p}$ is greater than the constructs present in the input process term. These cases occur when the input process term contains one of the following:

- a parallel composition;
- an alternative composition, (that requires the algorithm that introduces a new root, see Section 4.6.6); and

- an urgent channel scope operator.

In case of alternative composition with a new root and the urgent channel communication, the size of the linear process equation $\bar{p}$ is still linear with respect to the number of constructs in the input process term p.

Now we draw our attention to linearization of a parallel composition. Let the input process term $p = p_{\mathrm{s}} \parallel q_{\mathrm{s}}$.

Then its linear form $\widetilde{p}$, defined in Section 4.6.5 is repeated below:

$$
\begin{array}{l}
\widetilde{p} = \mathrm{Normalize}(p_{\mathrm{s}} \parallel q_{\mathrm{s}}) = \\
\;|[_{\mathrm{V}}\ \sigma^{p}_{\mathrm{pc}} \cup \mathrm{Shiftpcs}(\sigma^{q}_{\mathrm{pc}}, \mathrm{Count}(\overline{p})) \cup \sigma_p \cup \sigma_q, C_p \cup C_q, L_p \cup L_q \\
\;::\ \ |[_{\mathrm{R}}\ \{X \mapsto \mathrm{Setzero}\quad (\quad \mathrm{Extend}(\overline{p}, \mathrm{Shiftpcs}(\overline{q}, \mathrm{Count}(\overline{p}))) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad [\,]\ \mathrm{Extend}(\mathrm{Shiftpcs}(\overline{q}, \mathrm{Count}(\overline{p})), \overline{p}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \big[\,\big]\ \ \{\mathrm{COM}(\mathrm{alt}_p, \mathrm{alt}_q) \mid \mathrm{alt}_p \in \mathrm{Alt}(\overline{p}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathrm{alt}_q \in \mathrm{Alt}(\mathrm{Shiftpcs}(\overline{q}, \mathrm{Count}(\overline{p}))), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathrm{match}(\mathrm{alt}_p, \mathrm{alt}_q) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad) \\
\qquad :: (u_p \wedge u_q) \wedge (u^{p}_{\mathrm{pc}} \wedge \mathrm{Shiftpcs}(u^{q}_{\mathrm{pc}}, \mathrm{Count}(\overline{p}))) \curvearrowright X \\
\qquad]| \\
\\
\;]|,
\end{array}
$$

The linear form $\widetilde{p}$ is defined in terms of constructs of the linear form of p_{s} and q_{s}. (See Section 4.6.5 for the definitions of the constructs used to define $\widetilde{p}$.) The linear process equation of $\widetilde{p}$ is composed of process terms obtained by applying functions Extend and COM on the linear forms of p_{s} and q_{s}. Let m and n denote the sizes of the linear terms of p_{s} and q_{s} respectively. The size of the process term resulting from applying the function Extend on the linear form of p_{s} (or q_{s}) is again linear with respect to m (or n). On the other hand, the size of the process term resulting from applying the function COM on the linear forms of p_{s} and q_{s} can be equal to the product of their sizes, i.e. $m \times n$, incase all their alternatives can communicate with each other according to the function match.

Hence, we conclude that the size of the linear process term obtained from this linearization algorithm is of the order of the square of the size of the process term input to the linearization algorithm.

4.8 Concluding Discussion

We have presented a linearization algorithm for a subset of Hybrid χ specifications. As mentioned in the introduction to this chapter, a concern in linearization is the increase in size of the resulting linear term due to elimination of a parallel operator. The linearization algorithm presented in this chapter uses discrete counters which we call *program counters*. In the linear form of a parallel composition, a separate program counter is used to represent the constructs of each component running in parallel. Action interleaving of components is modelled by updating of program counters. This reduces the increase in size owing to the elimination of a parallel operator.

This section is composed of several issues of interest to our linearization algorithm.

Limitations and Benefits of the algorithm

Some restrictions have been imposed on the input process terms to the linearization algorithm given in this chapter. These restrictions are the same as those for the previous algorithm given in [The06]. They are as follows: only complete recursion definitions, i.e. recursion definitions that do not refer to recursion variables defined outside the recursion scope are allowed; an occurrence of a recursion variable must be guarded; only tail recursion is permitted and a recursion variable cannot occur in a parallel composition. Completeness of recursion definitions makes the task of linearization easier and it does not pose any limitations on the expressiveness of specifications. The restriction of guardedness is required for uniqueness of solutions for recursion variables. Recursion over parallel composition is disallowed. This restriction prevents possibly infinite parallelism as in $X_1 \mapsto a;\ X_1 \parallel b;\ Y_2$. But parallel composition of recursion scopes is allowed which can model parallelism among different components of a system. As mentioned in Section 4.6.12, recursion variables with process definitions consisting of variable scopes and self recursion are disallowed. Such process terms implement infinite instances of local variables and it is not possible to eliminate the variable scope from them. Corresponding to the variable scope operator in Hybrid χ is the abstraction operator in HyPA [CR05]. In the linearization of HyPA process terms [BRC06], an abstraction operator is only allowed at the top most level in the input to the linearization algorithm. The reason being the same as that an abstraction operator cannot be eliminated from recursive equations.

In comparison to the linearization algorithms for HyPA and μCRL, our linearization algorithm is much simpler. No complex data structures (such as stacks in HyPA and lists, multi-sets and stacks in μCRL) are used during linearization. This makes the intermediate and final linear forms of our linearization algorithm more readable than the linear forms obtained from other linearization algorithms. On the other hand, the use of these data structures would allow more flexibility in input process terms. Stacks are needed in the linearization of a sequential composition of parameterized recursion variables. In μCRL [Use02], recursive occurrences of parallel composition and of renaming operators are linearized using lists of multi-sets.

Updating the algorithm for Hybrid χ 2.0

In our linearization algorithm, we reason about the values of program counters. We manipulate the values of program counters to correctly model a sequential, parallel, alternative composition and other operators of Hybrid χ such as variable scope operator, recursion scope operator etc. Although there have been some changes in Hybrid χ 2.0, (see [BHR$^+$08]), the reasoning applied in the current linearization algorithm is still valid for linearizing a subset of the process terms in the new version given in [BHR$^+$08]. The main difference between the current version (linearized in this chapter) and the new version of Hybrid χ 2.0, is that in the current version, all actions and channels are by default lazy, i.e. they can delay arbitrarily, unless the delay is restricted by an urgent channel scope operator or by the delay conditions of invariants and urgency process terms. In the Hybrid χ 2.0 given in [BHR$^+$08], the channels and actions are classified as lazy or urgent and this information is placed in the environment of a process. The linearization algorithm in this chapter contributes to the linearization of process terms with lazy actions and channels in Hybrid χ 2.0. The linearization of process terms with urgent actions and channels is not covered by the present algorithm.

Future Work

Some suggestions for future work include the following:

- Updating the present linearization algorithm for Hybrid χ 2.0.
- Giving a proof of the correctness of the algorithm that shows that the linearized form of a process obtained from our linearization algorithm, behaves bisimilar to the input process term.
- Making a linearization tool based on our linearization algorithm. It is intended that a linearization tool with the new algorithm will be built in the ASF+SDF [BKV01]. The implementation of this algorithm has been delayed in favor of the ongoing work on the syntax and semantics of Hybrid χ 2.0.

Chapter 5

Conclusions and Open Problems

The research presented in this thesis is related to formal description and analysis of hybrid systems through process algebras. This chapter contains some final remarks concerning the research included in this thesis. The chapter consists of two parts: a conclusion to the research included in this thesis; and a discussion on open problems.

5.1 Conclusion

The aim of the research covered in this thesis was to contribute to the general progress of formal description and analysis of hybrid systems through process algebras. The course we took was to develop some useful extensions to existing process algebras for hybrid systems.

First, we conducted a comparative survey of existing process algebras for hybrid systems. Our aim was to understand the essential ingredients of a process algebra for hybrid systems and to familiarize ourselves with the state of the art in this field. We hoped that this study will guide us to interesting ideas for further developing existing process algebras.

The comparative study was successful in the sense that it presented a number of ideas for improvements and developments in existing process algebras for hybrid systems. The developments suggested by our study include:

1. HyPA: Extend HyPA with a time deterministic choice operator; Extend HyPA with consistency.

2. ϕ-calculus: Correctly define the replication operator $!P$ with regards to urgency in communication; Permit updates of variables according to predicates on variables; Allow interaction among the continuous flows of processes in a parallel composition.

3. Hybrid χ : Develop a linearization algorithm for Hybrid χ that is more efficient than the current one [The06], in terms of the size of the linear form; Extend Hybrid χ with concept of initially stateless bisimulation; Extend the axiomatic reasoning for Hybrid χ.

4. ACP^{srt}_{hs}: Extend ACP^{srt}_{hs} with variable abstraction; Extend the axiomatization of $ACP^{srt}_{hs} + INT$, so that parallel operator can be eliminated from all $ACP^{srt}_{hs} + INT$ closed terms.

As discussed in the introduction chapter, our study is not complete as three process algebras for hybrid systems, HYPE [GBH08], BHPC [Kri06] and Hybrid CSP [Jif94], are not included in it. Process algebras HYPE and BHPC are very recent process algebras and could not be included in our study due to lack of time to do the required research. Hybrid CSP could not be included because we were unclear about the semantics presented for Hybrid CSP in [Jif94]. Completeness of a comparative study is desirable but it was not exactly aimed at by us. Process algebras for hybrid systems is an active field of research–often existing theories undergo changes (like Hybrid χ), and new theories appear (like BHPC and HYPE). Therefore, attaining completeness is elusive.

Analyzing the impact of our work: we chose to implement two ideas put forward by the comparative study, i.e.,

1. Adding a variable abstraction operator to ACP^{srt}_{hs}; and

2. Developing a more efficient linearization algorithm for Hybrid χ (in terms of the size of the linear form obtained) than is currently available.

Two other improvements suggested by our study, namely, correcting the semantics of the replication operator and arbitrarily delaying assignments, (which serve to have the same effect as of updating variables according to predicates), have been taken into account by the developers of ϕ-calculus, see [RS]. The study has also been consulted by the developers of a new process algebra HYPE, see [GBH08].

During our work on adding variable abstraction to ACP^{srt}_{hs} [BM05], we discovered a number of errors in it. The errors found are, that in the current semantics of ACP^{srt}_{hs} [BM05], the choice operator is not associative, the axiom of time determinism (SRT3) does not hold and a number of other (less important axioms) are not preserved by the semantics. Following this discovery, we switched our goal from extending ACP^{srt}_{hs} to correcting it. Switching goals from extending ACP^{srt}_{hs} to correcting it, was justified as associativity is a fundamental property of the choice operator and time determinism expressed by axiom (SRT3) occupies a central place in many process algebras for timed systems, see for example [BM02b, BM02a, NS94, MT90, RR88].

Our new goal was then to present a complete corrected basic process algebra for hybrid systems, abbreviated as BPA^{srt}_{hs}. BPA^{srt}_{hs} is ACP^{srt}_{hs} without parallelism.

In this thesis, the target of presenting a complete and corrected process algebra BPA^{srt}_{hs} has not been achieved. In chapter 3, we present two proposals to correct a basic sub-algebra of ACP^{srt}_{hs}, called $BPA^{srt}_{\perp}$. Process algebra ACP^{srt}_{hs} is composed of a number of simpler sub-algebras. One of the errors in ACP^{srt}_{hs}, i.e. the error in the axiom of time determinism can be traced to its basic component $BPA^{srt}_{\perp}$. Hence, correcting $BPA^{srt}_{\perp}$ is the first step towards rectifying the errors in ACP^{srt}_{hs}. In fact, our contribution is more than just the proposals for $BPA^{srt}_{\perp}$ given in chapter 3. We have raised questions regarding signals and time determinism that were not satisfactorily addressed before.

We expect that solving the issues raised in chapter 3 will lead to a correct proposal for the complete process algebra ACP^{srt}_{hs}.

Our third goal was to develop a linearization algorithm for Hybrid χ, that is more efficient in terms of the size of the linear form, than the current linearization algorithm [The06]. The work in chapter 4, fulfills this goal. An algorithm is presented, whose complexity, in terms of the size of the linear term obtained, is less than that of the current one [The06]. Using a tool implementing our linearization algorithm, we hope to be able to linearize larger Hybrid χ process specifications which could not be linearized before due to a memory overflow.

5.2 A Discussion on Open Issues

Process algebra for hybrid systems is a relatively new field. Hence, there are a number of open issues to be solved. Many of these are well-known problems, such as optimizing linearization of specifications; developing tools and techniques to aid in soundness proofs; developing simulation and verification tools and proving completeness of axioms. Whereas there are others that have not been given ample attention, such as consistency and choice resolution in hybrid process algebras. In this section, we discuss some of the open problems we find interesting.

Consistency

The idea of consistency originates from the idea of signals given in process algebra ACP_{ps} [BB97]. The significance of consistency is that it reduces the reachable state space of a system to a set of states that satisfy some assumptions about the environment. A developer may want to study the behaviour of a system under certain conditions. Consistency provides a mechanism to express these assumptions in a process term and to implement them in a process behaviour.

Process algebras Hybrid χ and ACP^{srt}_{hs} take the idea of consistency from ACP_{ps} and use it to implement invariants as defined in hybrid automata of [Hen96]. In Hybrid χ and ACP^{srt}_{hs}, every delay proposition is in fact an invariant. The semantics of these process algebras is such that during a delay a state violating the delay proposition remains unreachable. If the delay proposition cannot be satisfied, the process deadlocks before it is violated.

On the other hand, process algebras HyPA and ϕ-calculus do not have operators to implement invariants. These process algebras take the approach that a state in which a delay predicate is violated can be reached, and a process cannot delay further on reaching such a state.

We suggest that delays and invariants should be implemented through two separate operators. Viewing every delay proposition as an invariant is not natural. Having separate operators for invariants and delays, we should use invariants only where required.

Choice Resolution

Choice resolution among delaying processes, is another interesting matter concerning hybrid process algebras. In case of a choice among discrete actions only, the choice is resolved non-deterministically. For choice among delaying processes, different approaches are taken. Flow synchronization is the approach adopted by Hybrid χ and ϕ calculus. Flow synchronization requires that all operands of a choice evolve synchronously during a delay. In HyPA, the choice among delaying processes is resolved non-deterministically as in the case of actions.

It is not yet certain what choice mechanism is suitable for ACP^{srt}_{hs}. Flow synchronization is not compatible with weak time determinism of ACP^{srt}, which is one of the basic algebras of ACP^{srt}_{hs}. In weak time determinism a choice between an action and a delayable process is allowed to delay. In flow synchronization, all operands of the choice must participate in a delay. We have proposed flow determinism as the choice resolution mechanism for ACP^{srt}_{hs} in Chapter 3, section 3.7.2. Flow determinism states that unique flows result in unique targets. If all operands can follow a certain flow, then a flow deterministic choice operator forces operands to synchronize. If all operands cannot follow a common flow, a flow deterministic operator makes a choice in favor of one of its operands and delays as the chosen operand. Referring back to our views on consistency, resolving choice between delays in this way implies that delay propositions and invariants are different.

Flow determinism appears to be a suitable semantics for any hybrid process theory. However one cannot insist on exclusive flow deterministic semantics as that reduces ones options for defining variable abstraction. We do not know yet how a variable abstraction operator can be defined in structural operational semantics for a flow deterministic semantics.

Completeness of axioms

Different representations of a process can be examined through applying axioms of a process algebra. Completeness theorem of a process algebra implies that equational reasoning of a process algebra is *complete*, i.e., comprehensive enough to prove any two bisimilar processes equivalent. More value can be added to the equational reasoning of process algebras for hybrid systems if completeness theorems are proved for their axiomatization. Completeness of axioms is not claimed by any process algebra included in our study in Chapter 2. A reason is the impossibility of enlisting all equivalent continuous behaviour in axioms. Abstracting from this detail, relative completeness results for axioms of process algebras for hybrid systems should be attempted to obtain insight in what is, and what is not attainable, through axiomatic reasoning in these algebras.

Theorems to Aid in Soundness Proofs

A noteworthy part of the effort in compiling this thesis has gone in proving the soundness of axioms for process algebra $BPA_{\perp}^{srt}$. In this regard, further development of meta-theorems defining rule-formats that guarantee certain properties of operators will be very useful, as these theorems discharge a developer of a process algebra from the obligation to do long and tedious soundness proofs. Currently, rule formats guaranteeing commutativity and associativity of operators are available see [Mou05, CMR08]. These rule formats are yet to be extended to structural operational semantics (SOS) with data before they can be used or proving soundness of axioms in a hybrid process algebra.

Process algebras for hybrid systems comprise a small part of the hybrid systems world. The regular appearance of new process algebras suggest that the field is growing at a fast pace. The field offers a number of interesting challenges to solve. Apart from solving the problems faced in the field, the progress of process algebras for hybrid systems requires the following:

1)- an insight into the working of hybrid systems, (so that the constructs of an algebra can be tuned to the way in which hybrid systems are usually engineered);

2)- a simplification of constructs, (so that the theory is tangible for engineers and control theorists); and

3)- a correlation with other formalisms, (so that guidance can be taken from techniques available in them).

Bibliography

[ADF$^+$06] Eugene Asarin, Thao Dang, Goran Frehse, Antoine Girard, Colas Le Guernic, and Oded Maler. Recent progress in continuous and hybrid reachability analysis. In *CACSD 2006: Computer Aided Control Systems Design*, 2006.

[AFIL07] Luca Aceto, Wan Fokkink, Anna Ingolfsdottir, and Bas Luttik. A finite equational base for CCS with left merge and communication merge. Technical Report 06-07, Department of Mathematics and Computer Science, Eindhoven University of Technology, 2007.

[AGH$^+$00] Rajeev Alur, Rado Grosu, Yerang Hur, Vijay Kumar, and Insup Lee. Modular specifications of hybrid systems in charon. In *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes In Computer Science*, pages 6–19, 2000.

[ALQ$^+$02] J.P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Transactions on Automatic Control*, 47:2–20, 2002.

[Alu07] R. Alur. Marrying words and trees. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM, Newyork, 2007.

[And02] Suzana Andova. *Probabilistic Process Algebra.* PhD thesis, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, 2002.

[Bae05] J.C.M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335:131–146, May 2005.

[BB97] J.C.M. Baeten and J.A. Bergstra. Process algebra with propositional signals. *Theoretical Computer Science*, 177:381–405, 1997.

[BBC$^+$96] N. Bjorner, A. Browne, E. Chang, M. Colon, A. Kapur, Z. Manna, H. Sipma, and T. Uribe. *STeP: deductive-algorithmic verifcation of reactive and real-time systems*, volume 1102 of *Lecture Notes in Computer Science*, pages 415–418. Springer, Berlin, 1996.

[BBF$^+$00] G. Berry, A. Bouali, X. Fornari, E. Ledinot, E. Nassor, and R. de Simone. Esterel: a formal method applied to avionic software development. *Science of Computer Programming*, 36:5–25, 2000.

[BBR09] J.C.M. Baeten, T. Basten, and M.A. Reniers. Process algebra (equational theories of communicating processes). to appear, 2009.

[BHR+08] D. A. van Beek, A. T. Hofkamp, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Syntax and formal semantics of chi 2.0. Technical Report SE-Report 2008-1, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven Unoversity of Technology, 2008. http://se.wtb.tue.nl/sereports/.

[BJM+03] D.A. van Beek, N.G. Jansen, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Relating chi to hybrid automata. In *Proceedings of the 2003 Winter Simulation Conference*, pages 632–640, 2003.

[BKV01] Mark van den Brand, Paul Klint, and Jurgen Vinju. The asf+sdf meta-environment: A component-based language development environment. In *Compiler Construction : 10th International Conference 2001*, volume 2027 of *Lecture Notes in Computer Science*, 2001.

[BLB08] L.M. Bujorianu, J. Lygeros, and M.C. Bujorianu. Towards a general theory of stochastic hybrid systems. Technical Report 08-25, Centre for Telematics and Information Technology, University of Twente, Enschede, 2008.

[BM97] M.S. Branicky and S.E. Mattsson. *Simulation of hybrid systems*, volume 1273 of *Lecture Notes in Computer Science*, pages 31–56. Springer,Berlin, 1997.

[BM99] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[BM02a] J.C.M. Baeten and C.A. Middelburg. *Process Algebra with Timing*, chapter Continuous Relative Timing. Springer, 2002.

[BM02b] J.C.M. Baeten and C.A. Middelburg. *Process Algebra with Timing*, chapter Discrete Relative Timing. Springer, 2002.

[BM03] J.A. Bergstra and C.A. Middelburg. Process algebra for hybrid systems. Technical Report 03-06, Department of Mathematics and Computer Science, Eindhoven University of Technology, June 2003.

[BM05] J.A. Bergstra and C.A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335, 2005.

[BM06] J.A. Bergstra and C.A. Middelburg. Continuity controlled hybrid automata. *Journal of Logic and Computer Programming*, 68:5–53, June-July 2006.

[BMR+06] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R. Schiffelers. Syntax and consistent equation semantics of hybrid chi. *Journal of Logic and Algebraic Programming*, 68:129–210, 2006.

[BMU01] J.A. Bergstra, C.A. Middelburg, and Y.S. Usenko. *Discrete Time Process Agebra and the Semantics of SDL*, pages 1209–1268. Elsevier, 2001.

[BR00] D.A. van Beek and J.E. Rooda. Languages and applications in hybrid modelling and simulation: Positioning of chi. *Control Engineering Practice*, 8:81–91, 2000.

[BR04] J.C.M. Baeten and M.A. Reniers. *Timed process algebra (with a focus on explicit termination and relative-timing)*, pages 59–97. Number 3185 in Lecture Notes in Computer Science. Springer Verlag, 2004.

[BRC06] P. van de Brand, M.A. Reniers, and P.J.L. Cuijpers. Linearization of hybrid processes. *Journal of Logic and Algebraic Programming*, 68:54–104, 2006.

[BRS$^+$07] D.A. van Beek, J.E. Rooda, R.R.H. Schiffelers, K.L. Man, and M.A. Reniers. Relating hybrid chi to other formalisms. *Electronic Notes in Theoretical Computer Science*, 191:85–113, October 2007.

[BRSR07] D.A. van Beek, M.A. Reniers, R.R.H. Schiffelers, and J.E. Rooda. Foundations of a compositional interchange format for hybrid systems. In *Hybrid Systems: Computation and Control, 10th International Workshop, HSCC 2007*, volume 4416 of *Lecture Notes in Computer Science*, pages 587–600. Springer, 2007.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1990.

[Chi] http://se.wtb.tue.nl/sewiki/chi. Systems Engineering Group, Faculty of mechanical Engineering, Eindhoven University of Technology.

[CJR96] Zhou Chao Chen, Wang Ji, and A.P. Ravn. *A Formal Description of Hybrid Systems*, volume 1066 of *Lecture Notes in Computer Science*, pages 511–530. Springer-Verlag, 1996.

[CMR08] Sjoerd Cranen, MohammadReza Mousavi, and Michel A. Reniers. A rule format for associativity. In Franck van Breugel and Marsha Chechik, editors, *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08)*, volume 5201 of *Lecture Notes in Computer Science*, pages 447–461, Toronto,Canada, 2008. Springer-Verlag, Berlin, Germany.

[Coq] http://coq.inria.fr/. The Coq proof assistant, INRIA, France.

[CR05] P.J.L. Cuijpers and M.A. Reniers. Hybrid process algebra. *Journal of Logic and Algebraic Programming*, 62:191–245, 2005.

[CSP] http://www.afm.lsbu.ac.uk/csp. Virtual Library Formal methods: CSP.

[DA01] Rene David and Hassane Alla. On hybrid petri nets. *Discrete Event Dynamic Systems*, 11:9–40, January 2001.

[DB95] R.C. Dorf and R.H. Bishop. *Modern Control Systems.* Series in Electrical and Computer Engineering: Control Engineering. Addison-Wesley, 1995.

[Fab99] G. Fabian. *A Language and Simulator for Hybrid Systems.* PhD thesis, Faculty of Mechanical Engineering, Eindhoven University of Technology, 1999.

[Fok98] Wan Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science, An EATCS Series. Springer, 1998.

[Fre05] Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Proceedings of the Fifth International Workshop on Hybrid Systems: Computation and Control (HSCC)*, Lecture Notes in Computer Science 3414, pages 258–273. Springer-Verlag, 2005.

[GBH08] Vashti Galpin, Luca Bortolussi, and Jane Hillston. Hype: hybrid systems modelled with flows. Technical Report EDI-INF-RR-1251, School of Informatics, University of Edinburgh, Apr 2008.

[GPU01] Jan Friso Groote, Alban Ponse, and Yaroslav S. Usenko. Linearization in parallel pcrl. *Journal of Logic and Algebraic Programming*, 48(1-2):39–70, 2001.

[GvW01] J.F. Groote and J.J. van Wamel. Analysis of three hybrid systems in timed μcrl'. *Science of Computer Programming*, 39:215–247, 2001.

[Hee99] Maurice Heemels. *Linear Complementarity Systems–A Study in Hybrid Dynamics.* PhD thesis, Department of Electrical Engineering, Eindhoven University of Technology, 1999.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of 11th Annual Symposium on Logic in Computer Science*, pages 278–292, 1996.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. In *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes In Computer Science*, pages 460–463, 1997.

[Hil94] J.A. Hillebrand. *The ABP and CABP-A comparison of performances in real time process algebra*, pages 124–147. Workshop in Computing Series. Springer Berlin, 1994.

[Hil05] Jane Hillston. Fluid flow approximation of pepa models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST05)*, IEEE, 2005.

[HLP$^+$00] Klaus Havelund, Mike Lowry, SeungJoon Park, Charles Pecheur, John Penix, Willem Visser, and Jon L. White. Formal analysis of the remote agent before and after flight. In *5th NASA Langley Formal Methods Workshop*, Williamsburg, Virginia, USA, June 2000.

[Hoa85] C. A. R. Hoare. *Communicating sequential processes.* Prentice Hall, 1985.

[HSB01] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. On the equivalence of classes of hybrid dynamical models. In *Proceedings of Conference on Decision and Control*, pages 364–369, 2001.

[IS00] J.I. Imura and A.J. van der Schaft. Characterization of well-posedness of piecewise linear systems. *IEEE Transactions on Automatic Control*, 45:1600–1619, 2000.

[Jif94] H. Jifeng. From CSP to hybrid systems. In A.W.Roscoe, editor, *A Classical Mind, Essays in Honour of C.A.R. Hoare*, pages 171–189. Prentice-Hall International, 1994.

[Kar98] Pim Kars. *Formal Methods in the Design of a storm surge Barrier Control System*, pages 353–367. Lecture Notes in Computer Science. Springer, Verlag, 1998.

[KBC07] U. Khadim, D.A. van Beek, and P.J.L. Cuijpers. Linearization of hybrid χ using program counters. Technical Report CS 07-18, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, August 2007.

[KC] U. Khadim and P.J.L. Cuijpers. A comaparative study of hybrid process algebras.

[KC08] U. Khadim and P.J.L. Cuijpers. Basic process algebra with standard relative time and non-existence $\text{bpa}_{\perp}^{srt}$. Technical Report CS 08-09, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, 2008.

[Kha05] U. Khadim. A graph model for a basic process algebra for hybrid systems. Technical Report CS 05-31, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, December 2005.

[Kha06] U. Khadim. A comparative study of process algebras for hybrid systems. Technical Report CS 06-23, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, August 2006.

[KO94] M.J. Koens and L.H. Oei. *A realtime μCRL specification of a system for traffic regulation at signalized intersections*, pages 252–279. Workshop in Computing Series. Springer Berlin, 1994.

[Kri06] T. Krilavicius. *Hybrid Techniques for Hybrid Systems.* PhD thesis, University of Twente, 2006.

[LPY99] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A new class of decidable hybrid systems. In *Proceedings of the Second International Workshop on Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes In Computer Science*, pages 137 – 151. Springer Verlag, 1999.

[LPY01] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. "symbolic reachability computation for families of linear vector fields". *Journal of Symbolic Computation*, 32:231–253, 2001.

[LSV03] N. Lynch, R. Segala, and F.W. Vaandrager. Hybrid i/o automata. *Information and Computation*, 185:105–157, 2003.

[Mat] http://www.wolfram.com/products/mathematica/index.html. Wolfrom Research.

[Mee01] J.M.S. van den Meerendonk. Specification and verification of a circuit in $\text{ACP}_{\text{drt}} - \text{ID}$. Master's thesis, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, 2001.

[MFR95] J.M. van de Mortel-Fronczak and J.E. Rooda. Application of concurrent programming to specification of industrial systems. In *Proceedings of the 1995 IFAC Symposium on Information Control Problems in Manufacturing*, pages 421–426, 1995.

[MFRvdN95] J.M. van de Mortel-Fronczak, J.E. Rooda, and N.J.M. van den Nieuwelaar. Specification of a flexible manufacturing system using concurrent programming. *The International Journal of Concurrent Engineering: Research and Applications*, 3(3):187–194, 1995.

[MH04] Mark Ryan Michael Huth. *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press, 2004.

[Mil99] Robin Milner. *Communicating and Mobile Systems: The π-Calculus.* Cambridge University Press, 1999.

[Mol90] F. Moller. The non-existence of finite axiomatisations for CCS congruences. In *Proceedings of Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 142–153. IEEE Computer Society Press, 1990.

[Mou05] MohammadReza Mousavi. *Structuring Structural Operational Semantics.* PhD thesis, Faculty of mathematics and Computer Science, Eindhoven University of Technology, 2005.

[MR07] M.R. Mousavi and M.A. Reniers. A congruence rule format with universal quantification. *Electronic Notes in Theoretical Computer Science*, 192:109–124, 2007.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proceedings CONCUR 90*, volume 458 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 1990.

[NP02] Stacy Nelson and Charles Pecheur. Formal verification of a next-generation space shuttle. In *Second Goddard Workshop on Formal Aspects of Agent-Based Systems (FAABS II)*, volume 2699 of *Lecture Notes in Computer Science.* Springer Verlag, October 2002.

[NS94] X. Nicollin and J. Sifakis. The algebra of timed processes atp: theory and application. *Information and Computaion*, 114:131–178, 1994.

[Pel01] Doron A. Peled. *Software Reliability Methods.* Springer, 2001.

[PHN03] A.Y. Pogromsky, W.P.M.H. Heemels, and H. Nijmeijer. On solution concepts and well-posedness of linear relay systems. *Automatica*, 39:2139–2147, 2003.

[PSE06] Charles Pecheur, Reid Simmons, and Peter Engrand. *Formal Verification of Autonomy Models: From Livingstone to SMV.* NASA Monographs in Systems and Software Engineering. Springer Verlag, 2006.

[PVS] http://pvs.csl.sri.com/. PVS Specification and Verification System, Formal Methods and Dependable Systems Program,Computer Science Laboratory, SRI International.

[PW98] J.W. Polderman and J.C. Willems. *Introduction to Mathematical Systems Theory: A Behavioural Approach*, volume 26 of *Texts in Applied Mathematics.* Springer-Verlag, 1998.

[Rou04] William C. Rounds. A spatial logic for the hybrid ϕ-calculus. In *Hybrid Systems, Computation, and Control 2004*, volume 2993 of *Lecture Notes in theoretical Computer Science*, pages 508–522. Springer-Verlag, 2004.

[RR88] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[RRS03] Mauno Rönkkö, Anders P. Ravn, and Kaisa Sere. Hybrid action systems. *Theoretical Computer Science*, 290:937–973, January 2003.

[RS] W. Rounds and H. Song. The ϕ-calculus: a hybrid extension of the π-calculus to embedded systems. a revision of [RS03], to appear.

[RS03] W.C. Rounds and H. Song. The ϕ-calculus: A language for distributed control of reconfigurable embedded systems. In F. Wiedijk, O. Maler, and A. Pnueli, editors, *Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003*, volume 2623 of *Lecture Notes in Computer Science*, pages 435–449. Springer-Verlag, 2003.

[RSC06] W. Rounds, H. Song, and K.J. Compton. Sphin: A model-checker for reconfigurable hybrid systems based on spin. *Electronic Notes on Theoretical Computer Science*, 145:167–183, 2006.

[RvW07] M.A. Reniers and M. van Weerdenburg. Action abstraction in process algebra : The case for an untimed silent step. In *IPM International Symposium on Fundamentals of Software Engineering (FSEN07)*, Tehran, Iran, 2007.

[Sch05] R.A. Schouten. Simulation of hybrid processes. Master's thesis, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, August 2005.

[SM06] R. Schiffelers and K.L. Man. *Formal specification and analysis of hybrid systems.* PhD thesis, Faculty of mathematics and Computer Science, Eindhoven University of Technology, 2006.

[Son98] E.D. Sontag. Mathematical control theory: Deterministic finite dimensional systems. *Texts in Applied Mathematics*, 6, 1998.

[Son05] H. Song. *Formal Specification and verification of reconfigurable hybrid systems.* PhD thesis, University of Michigan, 2005.

[SS98] A.J. van der Schaft and J.M. Schumacher. Complementarity modeling of hybrid systems. *IEEE Transactions on Automatic Control*, 43:483–490, 1998.

[SSKE01] B. Izaias Silva, Olaf Stursberg, Bruce H. Krogh, and Sebastian Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001.

[TB04] F.D. Torrisi and A. Bemporad. Hysdel–a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12:35–249, 2004.

[The06] R.J.M Theunissen. Process algebraic linearization of hybrid χ. Master's thesis, Faculty of Mechanical Engineering, Systems Engineering Group, Eindhoven University of Technology, June 2006.

[TK95] James H. Taylor and Dawit Kebede. Modeling and simulation of hybrid systems. In *Proceedings of the 34th Conference on Decision and Control*, December 1995.

[Use02] Y. Usenko. *Linearization in μCRL*. PhD thesis, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, 2002.

[Ver94] J.J. Vereijken. *Fischer's protocol in timed process algebra*, pages 245–284. Workshop in Computing Series. Springer Berlin, 1994.

[Ver95] Chris Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2:274–302, 1995.

[Was04] Andrzej Wasowski. Flattening statecharts without explosion. In *Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*. ACM, Newyork, 2004.

[Wee] M. van Weerdenburg. Automating soundness proofs. to appear.

[Wou01] Arno Wouters. Manual for the μCRL tool set (version 2.8.2). Technical Report Sen-R0130, Centrum voor Wiskune en Informatica, Amsterdam, December 2001.

Appendix A

Train Gate Controller in ϕ-calculus

A.1 State Space of the Train Gate Controller

There are no axioms in ϕ-calculus. Therefore algebraic manipulation cannot be applied to simplify the process term $\nu\langle\text{appr,exit,raise,lower}\rangle(Train \mid Gate \mid Contr)$. We can study the behaviour of the process term starting in an empty environment by using the action and time transition rules of ϕ-calculus.

The actual derivation comes out to be very lengthy. In order to condense it, we will often *combine* a number of action transition steps together. For example,

$\xrightarrow{\tau,\tau,\tau}$ implies that three silent steps are performed one after the other.

$\xrightarrow{[x=0].[\text{ reset } \dot{x} \text{ with } \{\dot{x} | \dot{x}=1\}]}$ implies two environmental actions, first just a guard and second, a reset action, take place one after the other.

Since a specific value needs to be given to environment variables, we choose the **maximum** speed, i.e. 52, and the **minimum** distance of the train from the gate, i.e. $-1400m$. In time transitions, we also take the **maximum** delay of 5 seconds for the controller, in order to see how the specification behaves under extreme conditions.

Let $\vec{A}$ denote the channel vector $\langle$appr,exit,raise,lower$\rangle$.

$$\mathbf{F_0}: \quad (\,\emptyset\,, \quad \nu\langle\vec{A}\rangle(Train \mid Gate \mid Contr)\,) \xrightarrow{\tau,\tau,\tau,\tau}$$

$$\mathbf{F_1}: \quad (\,\emptyset\,, \quad \nu\langle\vec{A}\rangle(\,[\,(x,y) := (-1400, 52)\,]\,.T^{far} \mid C^{idle} \mid \;[\,r := 90\,]\,.G^{op})\,)$$

$$\xrightarrow{[(x,y,r):=(-1400,52,90)]}$$

$$\mathbf{F_2}: \quad (\begin{pmatrix} x : -1400, y = 52 \\ r := 90 \\ \emptyset \end{pmatrix}, \qquad \nu\langle\vec{A}\rangle((T^{far} \mid C^{idle}) \mid G^{op})\,)$$

$$\xrightarrow{\begin{array}{c}[\text{ reset } x, \dot{x}, y \text{ with } \{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \{\dot{x} \mid \dot{x} = y\}]\end{array}}$$

$$\mathbf{F_3}: \quad (\begin{pmatrix} x : -1400, y : 52 \\ r : 90 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}\} \end{pmatrix}, \qquad \begin{array}{l}\nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \\ \mid C^{idle} \mid G^{op})\end{array} \quad)$$

$$\xrightarrow{[\ \text{ reset } \dot{r} \text{ with } \{\dot{r}|\dot{r}{=}0\}\ ]}$$

$$\mathbf{F_4}: \quad (\left(\begin{array}{c} x : -1400, y : 52 \\ r : 90 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{\dot{r} \mid \dot{r} = 0\}\} \end{array} \right) , \qquad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000]\,.\,\overline{appr}\,.\,T^{near} \\ \mid C^{idle} \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \quad)$$

$$\xrightarrow{x[0,400/52)}$$

$$\mathbf{F_5}: \quad (\left(\begin{array}{c} x : -1000, y : 52 \\ r : 90 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{\dot{r} \mid \dot{r} = 0\}\} \end{array} \right) , \qquad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000]\,.\,\overline{appr}\,.\,T^{near} \\ \mid C^{idle} \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \quad)$$

$$\xrightarrow{[x=-1000]}$$

$$\mathbf{F_6}: \quad (\left(\begin{array}{c} x : -1000, y : 52 \\ r : 90 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{\dot{r} \mid \dot{r} = 0\}\} \end{array} \right) , \qquad \begin{array}{l} \nu\langle\vec{A}\rangle(\overline{appr}\,.\,T^{near} \ \mid C^{idle} \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \quad)$$

$$\xrightarrow{\tau \equiv (\overline{appr} \mid appr)}$$

$$\mathbf{F_7}: \quad (\left(\begin{array}{c} x : -1000, y : 52 \\ r : 90 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{\dot{r} \mid \dot{r} = 0\}\} \end{array} \right) , \qquad \begin{array}{l} \nu\langle\vec{A}\rangle(T^{near} \ \mid [d := 0]\,.\,C^{dn} \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \quad)$$

$$\xrightarrow{\begin{array}{l} [d := 0].[\text{ reset } x, y \text{ with} \\ \{x \mid -1000 \leq x \leq 0\}, \{y \mid 40 \leq y \leq 52\}] \end{array}}$$

$$\mathbf{F_8}: \quad (\left(\begin{array}{c} x : -1000, y : 52 \\ r : 90, d : 0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{\dot{r} \mid \dot{r} = 0\}\} \end{array} \right) , \qquad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 0]\,.\,T^{past} \ \mid C^{dn} \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \quad)$$

$$\xrightarrow{[\text{reset } d,\dot{d} \text{ with } \{\dot{d}|\dot{d}{=}1\},\{d|d{\leq}5\}]}$$

$$\mathbf{F_9}: \quad (\left(\begin{array}{c} x : -1000, y : 52 \\ r : 90, d : 0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{\dot{r} \mid \dot{r} = 0\}, \\ \{\dot{d} \mid \dot{d} = 1\}, \\ \{d \mid d \leq 5\}\} \end{array} \right) , \qquad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 0]\,.\,T^{past} \mid \\ (\overline{lower}\,.\,[\text{reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}]\,.\,C^{idle} \\ +exit\,.\,C^{dn} + appr\,.\,C^{dn}) \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \quad)$$

$$\xrightarrow{x[0,5)}$$

$$\mathbf{F_{10}}:\quad \left(\left(\begin{array}{c} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x}=y\}, \\ \{\dot{r} \mid \dot{r}=0\}, \\ \{\dot{d} \mid \dot{d}=1\}, \\ \{d \mid d \leq 5\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0]\,.\,T^{past} \mid \\ (\overline{lower}\,.\,[\text{reset } d,\ \dot{d} \text{ with } \{\text{TRUE}\}]\,.\,C^{idle} \\ +exit\,.\,C^{dn} + appr\,.\,C^{dn}) \mid \\ (\text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{op})) \end{array} \right)$$

$$\xrightarrow{\tau\equiv(\overline{lower}|lower)}$$

$$\mathbf{F_{11}}:\quad \left(\left(\begin{array}{c} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x}=y\}, \\ \{\dot{r} \mid \dot{r}=0\}, \\ \{\dot{d} \mid \dot{d}=1\}, \\ \{d \mid d \leq 5\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \\ \mid [\text{ reset } d, \dot{d} \text{ with } \{\text{TRUE}\}].C^{idle} \\ \mid G^{dn}) \end{array} \right)$$

$$\xrightarrow{[\text{reset } d,\ \dot{d} \text{ with } \{\text{TRUE}\}]}$$

$$\mathbf{F_{12}}:\quad \left(\left(\begin{array}{c} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x}=y\}, \\ \{\dot{r} \mid \dot{r}=0\}, \\ \{\text{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \\ \mid C^{idle} \mid G^{dn}) \end{array} \right)$$

$$\xrightarrow{[\text{reset } r,\dot{r} \text{ with } \{r|r\geq 0\},\{\dot{r}|\dot{r}=-20\}]}$$

$$\mathbf{F_{13}}:\quad \left(\left(\begin{array}{c} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x}=y\}, \\ \{r \mid r \geq 0\}, \\ \{\dot{r} \mid \dot{r}=-20\}, \\ \{\text{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \\ \mid C^{idle} \mid ([r=0]\,.\,G^{cl}+ \\ \text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{up})) \end{array} \right) \quad \ldots\mathbf{(A)}$$

$$\xrightarrow{x[0,4.5)}$$

$$\mathbf{F_{14}}:\quad \left(\left(\begin{array}{c} x:-506, y:52 \\ r:0, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x}=y\}, \\ \{r \mid r \geq 0\}, \\ \{\dot{r} \mid \dot{r}=-20\}, \\ \{\text{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \\ \mid C^{idle} \mid ([r=0]\,.\,G^{cl}+ \\ \text{lower}\,.\,G^{dn} + \text{raise}\,.\,G^{up})) \end{array} \right)$$

$$\xrightarrow{[r=0]}$$

$$\mathbf{F_{15}}: \quad (\begin{pmatrix} x : -506, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid r \geq 0\}, \\ \{\dot{r} \mid \dot{r} = -20\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 0].T^{past} \\ \mid C^{idle} \mid G^{cl}) \end{array} \quad)$$

$$\xrightarrow{[\text{ reset } r, \dot{r} \text{ with } \{r \mid \dot{r}=0\}, \{r \mid r=0\}]}$$

$$\mathbf{F_{16}}: \quad (\begin{pmatrix} x : -506, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 0].T^{past} \\ \mid C^{idle} \mid \\ (\text{lower}\,.\,G^{cl} + \text{raise}\,.\,G^{up})) \end{array} \quad)$$

$$\xrightarrow{x[0, 9.730)}$$

$$\mathbf{F_{17}}: \quad (\begin{pmatrix} x : 0, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 0].T^{past} \\ \mid C^{idle} \mid \\ (\text{lower}\,.\,G^{cl} + \text{raise}\,.\,G^{up})) \end{array} \quad)$$

$$\xrightarrow{[x=0]\,.\,[\text{reset } x \text{ with } \{x \mid 0 \leq x \leq 100\}]}$$

$$\mathbf{F_{18}}: \quad (\begin{pmatrix} x : 0, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid 0 \leq x \leq 100\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 100].\overline{exit} \\ .\,[(x, y) := (-1400, 52)]\,.\,T^{far} \\ \mid C^{idle} \mid (\text{lower}\,.\,G^{cl} + \text{raise}\,.\,G^{up})) \end{array} \quad)$$

$$\xrightarrow{x[0, 100/52)}$$

$$\mathbf{F_{19}}: \quad (\begin{pmatrix} x : 100, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid 0 \leq x \leq 100\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = 100].\overline{exit} \\ .\,[(x, y) := (-1400, 52)]\,.\,T^{far} \\ \mid C^{idle} \mid (\text{lower}\,.\,G^{cl} + \text{raise}\,.\,G^{up})) \end{array} \quad)$$

$$\xrightarrow{[x=100]}$$

$$\mathbf{F_{20}}: \quad (\begin{pmatrix} x : 100, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid 0 \leq x \leq 100\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle(\overline{exit} \\ .\,[(x, y) := (-1400, 52)]\,.\,T^{far} \\ \mid C^{idle} \mid \\ (\text{lower}\,.\,G^{cl} + \text{raise}\,.\,G^{up})) \end{array} \quad)$$

$$\xrightarrow{\tau\equiv(\overline{exit}|exit)}$$

$$\mathbf{F_{21}}: \quad \left(\left(\begin{array}{c} x:100, y:52 \\ r:0, d:5 \\ \{\{x \mid 0 \leq x \leq 100\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\mathrm{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle((([(x,y) := (-1400,52)].T^{far} \\ \mid [d := 0].C^{up}) \mid (\mathrm{lower}.G^{cl} + \mathrm{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{[(x,y):=(-1400,52)].[d:=0]}$$

$$\mathbf{F_{22}}: \quad \left(\left(\begin{array}{c} x:-1400, y:52 \\ r:0, d:0 \\ \{\{x \mid 0 \leq x \leq 100\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\mathrm{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle((T^{far} \\ \mid C^{up}) \mid (\mathrm{lower}.G^{cl} + \mathrm{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\text{ reset } x, \dot{x}, y \text{ with} \\ \{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}] \end{array}}$$

$$\mathbf{F_{23}}: \quad \left(\left(\begin{array}{c} x:-1400, y:52 \\ r:0, d:0 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\mathrm{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \mid C^{up} \\ \mid (\mathrm{raise}.G^{cl} + \mathrm{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\text{reset } d, \dot{d} \text{ with } \{\dot{d} \mid \dot{d} = 1\}, \\ \{d \mid d \leq 5\}] \end{array}}$$

$$\mathbf{F_{24}}: \quad \left(\left(\begin{array}{c} x:-1400, y:52 \\ r:0, d:0 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\mathrm{TRUE}\}, \\ \{\dot{d} \mid \dot{d} = 1\}, \\ \{d \mid d \leq 5\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \mid \\ (\overline{raise}.\ [\text{ reset } d,\ \dot{d} \text{ with } \{\mathrm{TRUE}\}\ ]\ .C^{idle} \\ \ +exit.C^{up} + appr.[d := 0]C^{dn}) \\ \mid (\mathrm{lower}.G^{cl} + \mathrm{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{x[0,5)}$$

$$\mathbf{F_{25}} : \quad \left(\begin{pmatrix} x : -1140, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}, \\ \{\dot{d} \mid \dot{d} = 1\}, \\ \{d \mid d \leq 5\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \mid \\ (\overline{raise}.\ [\text{ reset } d,\ \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle} \\ \ +exit.C^{up} + appr.[d := 0]C^{dn}) \\ \mid (\text{lower}.G^{cl} + \text{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{\tau \equiv (\overline{raise} | raise)}$$

$$\mathbf{F_{26}} : \quad \left(\begin{pmatrix} x : -1140, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}, \\ \{\dot{d} \mid \dot{d} = 1\}, \\ \{d \mid d \leq 5\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \mid \\ \ [\text{ reset } d,\ \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle} \\ \mid G^{up}) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\text{ reset } d, \dot{d} \text{ with } \{TRUE\}][\text{ reset } r, \dot{r} \\ \text{ with } \{r \mid r \leq 90\}, \{\dot{r} \mid \dot{r} = 20\}] \end{array}}$$

$$\mathbf{F_{27}} : \quad \left(\begin{pmatrix} x : -1140, y : 52 \\ r : 0, d : 5 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \ \mid C^{idle} \\ \mid ([r = 90].G^{op} + raise.G^{up} + \\ lower.G^{dn})) \end{array} \right)$$

$$\xrightarrow{x[0,2.692)}$$

$$\mathbf{F_{28}} : \quad \left(\begin{pmatrix} x : -1000, y : 52 \\ r : 53.84, d : 5 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x = -1000].\overline{appr}.T^{near} \ \mid C^{idle} \\ \mid ([r = 90].G^{op} + raise.G^{up} + \\ lower.G^{dn})) \end{array} \right)$$

$$\xrightarrow{[x=-1000];\tau\equiv(\overline{appr}|appr)}$$

$$\mathbf{F_{29}}: \quad \left(\begin{pmatrix} x:-1000, y:52 \\ r:53.84, d:5 \\ \{\{x \mid x \leq -1000\}, \\ \{y \mid 48 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\mathrm{TRUE}\}\} \end{pmatrix} , \quad \begin{array}{l} \nu\langle\vec{A}\rangle(T^{near} \mid \\ \ [\ d:=0\ ]\ .C^{dn} \\ \mid ([r=90].G^{op} + raise.G^{up} + \\ lower.G^{dn})) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\text{ reset } x, y \text{ with} \\ \{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}][d:=0] \end{array}}$$

$$\mathbf{F_{30}}: \quad \left(\begin{pmatrix} x:-1000, y:52 \\ r:53.84, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\mathrm{TRUE}\}\} \end{pmatrix} , \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ C^{dn} \\ \mid ([r=90].G^{op} + raise.G^{up} + \\ lower.G^{dn})) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\text{ reset } d, \dot{d} \text{ with } \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}] \end{array}}$$

$$\mathbf{F_{31}}: \quad \left(\begin{pmatrix} x:-1000, y:52 \\ r:53.84, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\mathrm{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{pmatrix} , \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ (\overline{lower}.[\text{reset } d,\, \dot{d} \text{ with } \{\mathrm{TRUE}\}].C^{idle} \\ \ +exit.C^{dn} + appr.C^{dn}) \\ \mid ([r=90].G^{op} + raise.G^{up} + \\ lower.G^{dn})) \end{array} \right)$$

Now the first train has exited the gate, i.e. it has crossed the detector at $100m$, from the gate ($F_{19} \rightarrow F_{20}$). The controller has sent the *raise* signal to the gate ($F_{25} \rightarrow F_{26}$) and the gate is rising. At the same time a new train arrives ($F_{28} \rightarrow F_{29}$). It reaches the $-1000m$ detector and sends an *approach* signal to the gate. From this stage, we derive transitions for two possible scenarios.

Scenario A

The controller takes 5 seconds before it sends *lower* to the gate. (Sum up the delays 1.808 + 3.192 from F_{31} to F_{35A}). The gate is fully opened when it receives the *lower* signal ($F_{34A} \rightarrow F_{35A}$).

$$\xrightarrow{x[0,1.808)}$$

$$\mathbf{F_{32A}}:\quad \left(\begin{pmatrix} x:-905.98, y:52 \\ r:90, d:1.808 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\text{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ (\overline{lower}.\ [\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle} \\ \quad +exit.C^{dn} + appr.C^{dn}) \\ \mid ([r=90].G^{op} + raise.G^{up} + \\ lower.G^{dn})) \end{array} \right)$$

$$\xrightarrow[\{r \mid r = 90\}, \{\dot{r} \mid \dot{r} = 0\}]]{[r=90].[\text{ reset } r, \dot{r} \text{ with}}$$

$$\mathbf{F_{33A}}:\quad \left(\begin{pmatrix} x:-905.98, y:52 \\ r:90, d:1.808 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 90\}, \\ \{\text{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ (\overline{lower}.\ [\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle} \\ \quad +exit.C^{dn} + appr.C^{dn}) \\ \mid (lower.G^{dn} + raise.G^{op})) \end{array} \right)$$

$$\xrightarrow{x[0,3.192)}$$

$$\mathbf{F_{34A}}:\quad \left(\begin{pmatrix} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 90\}, \\ \{\text{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ (\overline{lower}.\ [\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle} \\ \quad +exit.C^{dn} + appr.C^{dn}) \\ \mid (lower.G^{dn} + raise.G^{op})) \end{array} \right)$$

$$\xrightarrow{\tau \equiv (\overline{lower} \mid lower)}$$

$$\mathbf{F_{35A}}:\quad \left(\begin{pmatrix} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 90\}, \\ \{\text{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ \ [\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\ ]\ .C^{idle} \\ \mid G^{dn}) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\ ] \\ [\ \text{ reset } r, \dot{r} \text{ with } \{r \mid r \geq 0\}, \\ \{r \mid \dot{r} = -20\}\ ] \end{array}}$$

$$\mathbf{F_{36A}}: \quad \left(\left(\begin{array}{c} x:-740, y:52 \\ r:90, d:5 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = -20\}, \\ \{r \mid r \geq 0\}, \\ \{\text{TRUE}\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid C^{idle} \\ \mid (\;[\;r=0\;]\;.G^{cl}+ \\ \text{lower}.G^{dn} + \text{raise}.G^{up})) \end{array} \right)$$

The state F_{36A} maps to the state F_{13}. The derivation is then repeated from step (**A**).

Scenario B

The controller sends the *lower* signal immediately ($F_{31} \rightarrow F_{32B}$). The gate is not fully open when it receives the *lower* signal.

$$\mathbf{F_{31}}: \quad \left(\left(\begin{array}{c} x:-1000, y:52 \\ r:53.84, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\text{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ (\overline{lower}.\;[\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\;]\;.C^{idle} \\ \quad +exit.C^{dn} + appr.C^{dn}) \\ \mid ([r=90].G^{op} + raise.G^{up}+ \\ lower.G^{dn})) \end{array} \right)$$

$$\xrightarrow{\tau \equiv (\overline{lower}|lower)}$$

$$\mathbf{F_{32B}} \quad \left(\left(\begin{array}{c} x:-1000, y:52 \\ r:53.84, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 20\}, \\ \{r \mid r \leq 90\}, \\ \{\text{TRUE}\}, \\ \{d \mid d \leq 5\}, \\ \{\dot{d} \mid \dot{d} = 1\}\} \end{array} \right), \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid \\ \;[\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\;]\;.C^{idle} \\ \mid G^{dn}) \end{array} \right)$$

$$\xrightarrow{\begin{array}{c} [\text{ reset } d,\, \dot{d} \text{ with } \{\text{TRUE}\}\;] \\ [\;\text{ reset } r,\dot{r} \text{ with } \{r \mid r \geq 0\}, \\ \{r \mid \dot{r} = -20\}\;] \end{array}}$$

$$\mathbf{F_{33B}} \quad \left(\begin{pmatrix} x:-1000, y:52 \\ r:53.84, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = -20\}, \\ \{r \mid r \geq 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid C^{idle} \\ \mid (\; [\; r=0 \;] \; .G^{cl}+ \\ \text{lower}.G^{dn} + \text{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{x[0,2.692)}$$

$$\mathbf{F_{34B}} \quad \left(\begin{pmatrix} x:-860, y:52 \\ r:0, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = -20\}, \\ \{r \mid r \geq 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid C^{idle} \\ \mid (\; [\; r=0 \;] \; .G^{cl}+ \\ \text{lower}.G^{dn} + \text{raise}.G^{up})) \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} [\; r=0 \;]\; ; [\text{ reset } r, \dot{r} \text{ with } \{r \mid r=0\}, \\ \{r \mid \dot{r}=0\}] \end{array}}$$

$$\mathbf{F_{35B}} \quad \left(\begin{pmatrix} x:-860, y:52 \\ r:0, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid C^{idle} \\ \mid (lower.G^{cl} + raise.G^{up})) \end{array} \right)$$

$$\xrightarrow{x[0,860/52)}$$

$$\mathbf{F_{36B}} \quad \left(\begin{pmatrix} x:0, y:52 \\ r:0, d:0 \\ \{\{x \mid -1000 \leq x \leq 0\}, \\ \{y \mid 40 \leq y \leq 52\}, \\ \{\dot{x} \mid \dot{x} = y\}, \\ \{r \mid \dot{r} = 0\}, \\ \{r \mid r = 0\}, \\ \{\text{TRUE}\}\} \end{pmatrix}, \quad \begin{array}{l} \nu\langle\vec{A}\rangle([x=0].T^{past} \mid C^{idle} \\ \mid (lower.G^{cl} + raise.G^{up})) \end{array} \right)$$

From this stage onwards we can find repeating patterns in the derivation of transitions. F_{36B} behaves as F_{17}. Note the value of d in F_{36B} and F_{17} is different, but the value of d is only reflecting the last delay of the controller and does not play a role in the behaviour of F_{17} or F_{36B}.

A.2 PROMELA-Hybrid code for the train gate controller specification

```
 /*
 * Train-Gate-Controller example for SPHIN
 */

#define DELAY   5       /* controller delay */ #define hskip (1+1)
```

```
/* "skip" can’t be used in a hybrid statement */
 /* channel definition for
synchronization */
 chan app = [0] of {int };
 chan exit = [0] of { int };
  chan raise = [0] of { int };
chan lower = [0] of { int };

/* declared globally to allow monitor process to check the values.
*/ analog x = { [-1400,-1400], [48,52], [,-1000] }; analog g
=[90,90], [0,0], [90,90] };
 analog t = { [0,0], [0,0], [0,0] };

active proctype train() {
 far:     app!0 when { x in [-1000,-1000]} reset { x = { , [40,52],
          [,0] } };
 Near:    hskip when { x in [0,0] } reset { x = { , [40,52],
          [,100] } };
 past:    atomic {exit!0 when { x in [100,100] } reset {
          x = { [-1400,-1400], [48,52], [,-1000] } }; goto far } }


active proctype gate() { open:
        do
        :: raise?_
        :: atomic { lower?_ when {} reset { g = { , [-20,-20], [0,] } };
         goto lowering }
        od;

lowering:
        do
        :: lower?_
        :: atomic { raise?_ when {} reset { g = { , [20,20],
          [,90] }  };goto raising }
        :: atomic { hskip when { g in [0,0] } reset {
            g = { , [0,0], [0,0] } }; goto closed }
        od;

raising:
        do
        :: raise?_
        :: atomic { lower?_ when {} reset { g = { , [-20,-20],
         [0,] } }; goto lowering }
        :: atomic { hskip when { g in [90,90] } reset { g =
        { , [0,0], [90,90] } }; goto open }
        od;

closed:
        do
        :: lower?_
        :: atomic { raise?_ when {} reset { g = { , [20,20],
        [,90] } }; goto raising }
        od
```

```
}

active proctype controller() { idle:
        do
        :: atomic { app?_ when {} reset { t = { [0,0], [1,1],
         [,DELAY] } };  goto about_to_lower }
        :: atomic { exit?_ when {} reset { t = { [0,0], [1,1],
         [,DELAY] } }; goto about_to_raise }
        od;

about_to_lower:
        do
        :: atomic { exit?_ when {} reset { t = { [0,0], [1,1],
         [,DELAY] } }; goto about_to_raise }
        :: atomic { lower!0 when {} reset { t = { , [0,0], } };
         goto idle }
        od;

about_to_raise:
        do
        :: atomic { app?_ when {} reset { t = { [0,0], [1,1],
          [,DELAY] } };
        goto about_to_lower }
        :: atomic { raise!0 when {} reset { t = { , [0,0], } };
         goto idle }
        od
}

#define Dmin -350

active proctype monitor() {
        assert(false) when {x in [Dmin,100], g in (0,)} reset {}
}
```

Appendix B

Introduction to BPA_{hs}^{srt} Semantics

In this Section, we give a brief introduction to the semantics of Basic Process Algebra for Hybrid Systems.

To describe the behaviour of a hybrid process, we need to keep account of the values of model variables. The values of these variables may change gradually over an interval of time or suddenly when an action is performed. Assume a set V of model variables and a set A of actions. Let the set $\dot{V} = \{\dot{v} \mid v \in V\}$ denote the derivatives of all variables $v \in V$. A mapping of variables from the set $V \cup \dot{V}$ to the set of real numbers is called a valuation. We denote the set of all possible valuations by S, i.e. $S = V \cup \dot{V} \to R$. A *valuation* has been called as a *state* in [BM05].

A function of the type $[0,t] \to (V \to R)$ gives the evolution of variables over a duration $[0,t]$, $t \in R^{>0}$.

We define a set $\mathcal{D}$ for pairs of time durations and state evolution functions possible during a delay of that duration.

$$\mathcal{D} = \{(t, \rho) \mid t \in R^{\geq 0} \wedge \rho \in [0,t] \mapsto (V \mapsto R)\}$$

Let $\rho \in ([0,t] \to (V \to R))$. We use the notation $\rho \trianglerighteq r$, with $0 < r < t$, for the state evolution ρ shifted to left by r time units. The duration of $\rho \trianglerighteq r$ is $t - r$.

$$\begin{array}{ll} (\rho \trianglerighteq r)(0) = \rho(r) & \\ (\rho \trianglerighteq r)(s) = \rho(r+s), & s > 0 \\ \rho \trianglerighteq r \in ([0,t-r] \to (V \to R)) & \end{array}$$

The semantics of BPA_{hs}^{srt} uses four different relations. They are:

1. Action step Relation

2. Action termination Relation

3. Time step Relation; and

4. Signal Relation.

As we are dealing with hybrid processes, the sources and targets of transitions include valuations of variables to reflect how variables vary during an action or delay.

The relations are defined below:

(Let P be the set of all closed process terms of BPA_{hs}^{srt}, S be the set of all valuations, A be the set of all actions and $\mathcal{D}$ be the set of pairs of all possible time durations and variable evolutions during them.)

1. Action Step Relation: A process term can perform an action and become another process term.

   The Action Step relation is of type $P \times S \times A \times P \times S$.

   For a tuple $(x, \alpha, a, x', \alpha')$ in the Action Step Relation, we write:

   $$\langle x, \alpha \rangle \xrightarrow{a} \langle x', \alpha' \rangle$$

   This transition represents that in valuation α, x performs action a and then proceeds as process term x'. The new values of variables after the action are given by valuation α'.

2. Action Termination Relation: A process term can perform an action and terminate.

   The Action Termination Relation is of type $P \times S \times A \times S$.

   For a tuple (x, α, a, α') in the Action Termination Relation, we write:

   $$\langle x, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

   This transition represents that in valuation α, x performs action a and terminates. The new values of variables after the action are given by valuation α'.

3. Time Step Relation: A process term can idle for some time and become another process term.

   The Time Step Relation is of type $P \times S \times \mathcal{D} \times P \times S$.

   For a tuple, $(x, \alpha, (t, \rho), x', \alpha')$ in the Time Step Relation, we write:

   $$\langle x, \alpha \rangle \xmapsto{t,\rho} \langle x', \alpha' \rangle$$

   The above time step represents that in valuation α, x idles for t time units and then proceeds as process term x'. The values of variables during idling evolve according to the trajectory ρ. The values of variables at the end of delay are given by valuation α'. The valuations α and α' match with the values assigned by the trajectory ρ at instance 0 and r.

4. Signal Relation: The signal emitted by a process term holds in a given valuation.

   Signal Relation is of type $S \times P$. For a tuple (α, x) in the Signal Relation, we write:

   $$\alpha \in [\mathsf{s}(x)]$$

   The above predicate indicates that the signal emitted by process term x holds in valuation α.

Some operators (namely signal emission and signal evolution) in Process Algebra for Hybrid Systems associate propositions with process terms. These propositions then constitute the signal emitted by that process term. The rules stating when *signal emitted* by a process terms holds in a valuation are given in Table B.3.

A predicate $\langle x, \alpha \rangle \not\xmapsto{t}$ represents the following:

$$\nexists \rho \in [0, t] \to (V \to R), x' \in \mathcal{P}, \alpha' \in S : \langle x, \alpha \rangle \xmapsto{t,\rho} \langle x', \alpha' \rangle$$

B.1 Bisimulation

There are two kinds of bisimulation equivalences for defined in [BM05]. One is called *bisimulation* and the other is called *Interference Compatible bisimulation* or *ic-bisimulation.*

The behaviour of a hybrid process is specified in a valuation of model variables. Each action and time step of a process may modify the valuation.

In bisimulation equivalence, the initial behaviour of two processes is compared in a given valuation and for subsequent steps, the behaviour of two processes is compared in the valuation obtained at the end of the previous step.

It is defined as follows:

Definition 4 *A bisimulation is a symmetric binary relation $B \subseteq (P \times S) \times (P \times S)$ on pairs of closed process terms and valuations called configurations. For all configurations, $\langle x, \alpha\rangle$,$\langle y, \alpha\rangle$ with $(\langle x, \alpha\rangle,\langle y, \alpha\rangle) \in B$ the following conditions hold:*

- *for all actions $a \in A$, process terms $x' \in P$, valuations $\alpha' \in S$, if there is an action step $\langle x, \alpha\rangle \xrightarrow{a} \langle x', \alpha'\rangle$, then there exists a $y' \in P$ and an action step $\langle y, \alpha\rangle \xrightarrow{a} \langle y', \alpha'\rangle$. Also $(\langle x', \alpha'\rangle, \langle y', \alpha'\rangle) \in B$;*
- *for all actions $a \in A$, valuations $\alpha' \in S$, if there is a termination step $\langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$, then there also exists the termination step $\langle y, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$;*
- *for all delays $(r, \rho) \in \mathcal{D}$, process terms $x' \in P$, valuations $\alpha' \in S$, if there exists a time step $\langle x, \alpha\rangle \xmapsto{r,\rho} \langle x', \alpha'\rangle$, then there exists a $y' \in P$ and a time step $\langle y, \alpha\rangle \xmapsto{r,\rho} \langle y', \alpha'\rangle$. Also $(\langle x', \alpha'\rangle, \langle y', \alpha'\rangle) \in B$;*
- *if the signal relation $\alpha \in [\mathsf{s}(x)]$ holds then the signal relation $\alpha \in [\mathsf{s}(y)]$ also holds.*

Two configurations $\langle x, \alpha\rangle$ and $\langle y, \alpha\rangle$ are bisimulation equivalent *or* bisimilar *written as $\langle x, \alpha\rangle \mathrel{\underline{\leftrightarrow}} \langle y, \alpha\rangle$, if there exists a bisimulation relation B such that $(\langle x, \alpha\rangle,\langle y, \alpha\rangle) \in B$.*

Additionally, two process terms x and y are bisimulation equivalent *or* bisimilar *written as $x \mathrel{\underline{\leftrightarrow}} y$, if for all valuations α, there exists a bisimulation relation B such that $(\langle x, \alpha\rangle,\langle y, \alpha\rangle) \in B$.*

The above definition is sufficient when only sequential processes are considered. Bisimulation is not a congruence when parallel processes are studied. In case of parallelism, the valuation can be modified by a third process in parallel.

Consider the following example:

$$\begin{array}{rcl} X & = & = (v^{\bullet} = 1) \ulcorner\!\blacktriangledown \tilde{\tilde{a}} \cdot (v = 1) :\rightarrow \tilde{\tilde{b}} \\ Y & = & = (v^{\bullet} = 1) \ulcorner\!\blacktriangledown \tilde{\tilde{a}} \cdot \tilde{\tilde{b}} \end{array}$$

The two processes are bisimilar but when they are placed parallel with a process Z, they behave differently.

$$Z = (v^{\bullet} = 0) \ulcorner\!\blacktriangledown \tilde{\tilde{c}}$$

For an equivalence on processes to be a congruence with respect to parallel operator, the equivalence definition must cater for interferences by parallel processes.

In *Ic-bisimulation*, the initial behaviour of two processes is compared in all possible valuations and for subsequent steps, the same policy is adopted. I.e. at each stage the behaviour of two processes is compared in all possible valuations.

Its is defined as follows:

Definition 5 *An ic-bisimulation is a symmetric binary relation $B \subseteq P \times P$ on pairs of closed terms. For all pairs, (x, y) with $(x, y) \in B$ the following conditions hold:*

For all valuations α:

- *for all actions $a \in A$, process terms $x' \in P$, valuations $\alpha' \in S$, if there is an action step $\langle x, \alpha\rangle \xrightarrow{a} \langle x', \alpha'\rangle$, then there exists a $y' \in P$ and an action step $\langle y, \alpha\rangle \xrightarrow{a} \langle y', \alpha'\rangle$. Also $(x', y') \in B$;*
- *for all actions $a \in A$, valuations $\alpha' \in S$, if there is a termination step $\langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$, then there also exists the termination step $\langle y, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$;*
- *for all delays $(r, \rho) \in \mathcal{D}$, process terms $x' \in P$, valuations $\alpha' \in S$, if there exists a time step $\langle x, \alpha\rangle \xmapsto{r,\rho} \langle x', \alpha'\rangle$, then there exists a $y' \in P$ and a time step $\langle y, \alpha\rangle \xmapsto{r,\rho} \langle y', \alpha'\rangle$. Also $(x', y') \in B$;*
- *if the signal relation $\alpha \in [\mathsf{s}(x)]$ holds then the signal relation $\alpha \in [\mathsf{s}(y)]$ also holds.*

Two process terms x and y ic- bisimulation equivalent *or* ic-bisimilar *written as $x \underline{\leftrightarrow} y$, if there exists a bisimulation relation B such that $(x, y) \in B$.*

B.2 Transition Rules for BPA_{hs}^{srt}

We have for all closed terms x and x', for all $\alpha, \alpha' : V \cup \dot{V} \to \mathbb{R}$, $a \in A$, $r, s \in \mathbb{R}^{>}$ and $\rho \in \epsilon_r$,$\rho' \in \epsilon_{r+s}$ the following transition rules:

Table B.1: $BPA_{\mathrm{hs}}^{\mathrm{srt}}$-Transition Rules ($a \in A, r, s > 0$)

$$\frac{}{\langle \tilde{\tilde{a}}, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle}\ \text{HS-}\underline{1} \qquad \frac{\langle x, \alpha\rangle \xrightarrow{a} \langle x', \alpha'\rangle}{\langle \sigma^{0}_{\mathsf{rel}}(x), \alpha\rangle \xrightarrow{a} \langle x', \alpha'\rangle}\ \text{HS-}\underline{2}$$

$$\frac{\langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle}{\langle \sigma^{0}_{\mathsf{rel}}(x), \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle}\ \text{HS-}\underline{3} \qquad \frac{\langle x, \alpha\rangle \xmapsto{r,\rho} \langle x', \alpha'\rangle}{\langle \sigma^{0}_{\mathsf{rel}}(x), \alpha\rangle \xmapsto{r,\rho} \langle x', \alpha'\rangle}\ \text{HS-}\underline{4}$$

$$\frac{}{\langle \sigma^{r+s}_{\mathsf{rel}}(x), \alpha\rangle \xmapsto{r,\rho} \langle \sigma^{r}_{\mathsf{rel}}(x), \alpha'\rangle}\ \text{HS-}\underline{5} \qquad \frac{\alpha' \in [\mathsf{s}(x)]}{\langle \sigma^{r}_{\mathsf{rel}}(x), \alpha\rangle \xmapsto{r,\rho} \langle x, \alpha'\rangle}\ \text{HS-}\underline{6}$$

$$\frac{\langle x, \alpha'\rangle \xmapsto{s,\rho' \trianglerighteq r} \langle x', \alpha''\rangle}{\langle \sigma^{r}_{\mathsf{rel}}(x), \alpha\rangle \xmapsto{r+s,\rho'} \langle x', \alpha''\rangle}\ \text{HS-}\underline{7} \qquad \frac{\langle x, \alpha\rangle \xrightarrow{a} \langle x', \alpha'\rangle, \alpha \in [\mathsf{s}(y)]}{\langle x + y, \alpha\rangle \xrightarrow{a} \langle x', \alpha'\rangle}\ \text{HS-}\underline{8}$$

$$\frac{\langle y, \alpha\rangle \xrightarrow{a} \langle y', \alpha'\rangle, \alpha \in [\mathsf{s}(x)]}{\langle x + y, \alpha\rangle \xrightarrow{a} \langle y', \alpha'\rangle}\ \text{HS-}\underline{9} \qquad \frac{\langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle, \alpha \in [\mathsf{s}(y)]}{\langle x + y, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle}\ \text{HS-}\underline{10}$$

Continued on Next Page. . .

Table B.1 – Continued ($a \in A, r, s > 0$)

$$\frac{\langle y,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle, \alpha \in [\mathsf{s}(x)]}{\langle x+y,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle} \quad \text{HS-}\underline{11}$$

$$\frac{\begin{array}{c}\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle,\\ \langle y,\alpha\rangle \overset{r}{\not\longmapsto}, \alpha \in [\mathsf{s}(y)]\end{array}}{\langle x+y,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle} \quad \text{HS-}\underline{12}$$

$$\frac{\begin{array}{c}\langle y,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle y',\alpha'\rangle,\\ \alpha \in [\mathsf{s}(x)], \langle x,\alpha\rangle \overset{r}{\not\longmapsto}\end{array}}{\langle x+y,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle y',\alpha'\rangle} \quad \text{HS-}\underline{13}$$

$$\frac{\begin{array}{c}\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle,\\ \langle y,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle y',\alpha'\rangle\end{array}}{\langle x+y,\alpha\rangle, \overset{r,\rho}{\longmapsto} \langle x'+y',\alpha'\rangle} \quad \text{HS-}\underline{14}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle}{\langle x\cdot y,\alpha\rangle \xrightarrow{a} \langle x'\cdot y,\alpha'\rangle} \quad \text{HS-}\underline{15}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle, \alpha' \in [\mathsf{s}(y)]}{\langle x\cdot y,\alpha\rangle \xrightarrow{a} \langle y,\alpha'\rangle} \quad \text{HS-}\underline{16}$$

$$\frac{\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle}{\langle x\cdot y,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x'\cdot y,\alpha'\rangle} \quad \text{HS-}\underline{17}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle}{\langle \psi :\rightarrow x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle} \quad \alpha \models \psi \;\; \text{HS-}\underline{18}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle}{\langle \psi :\rightarrow x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle} \quad \alpha \models \psi \;\; \text{HS-}\underline{19}$$

$$\frac{\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle}{\langle \psi :\rightarrow x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle} \quad \alpha \models \psi \;\; \text{HS-}\underline{20}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle}{\langle \psi \mathbin{\wedge\!\!\blacktriangle} x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle} \quad \alpha \models \psi \;\; \text{HS-}\underline{21}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle}{\langle \psi \mathbin{\wedge\!\!\blacktriangle} x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle} \quad \alpha \models \psi \;\; \text{HS-}\underline{22}$$

$$\frac{\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle}{\langle \psi \mathbin{\wedge\!\!\blacktriangle} x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle} \quad \alpha \models \psi \;\; \text{HS-}\underline{23}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle}{\langle \phi \curvearrowright_V x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle} \quad \alpha \models \phi \;\; \text{HS-}\underline{24}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle}{\langle \phi \curvearrowright_V x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle} \quad \alpha \models \phi \;\; \text{HS-}\underline{25}$$

$$\frac{\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle}{\langle \phi \curvearrowright_V x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle \phi \curvearrowright_V x',\alpha'\rangle} \quad \alpha \overset{r,\rho}{\longmapsto} \alpha' \models_V \phi \;\; \text{HS-}\underline{26}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle}{\langle \chi \mathbin{\ulcorner\!\!\blacktriangledown} x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle} \quad \alpha \rightarrow \alpha' \models \chi \;\; \text{HS-}\underline{27}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle}{\langle \chi \mathbin{\ulcorner\!\!\blacktriangledown} x,\alpha\rangle \xrightarrow{a} \langle\surd,\alpha'\rangle} \quad \alpha \rightarrow \alpha' \models \chi \;\; \text{HS-}\underline{28}$$

$$\frac{\langle x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle}{\langle \chi \mathbin{\ulcorner\!\!\blacktriangledown} x,\alpha\rangle \overset{r,\rho}{\longmapsto} \langle x',\alpha'\rangle} \quad \alpha \models {}^{\circ}\chi \;\; \text{HS-}\underline{29}$$

$$\frac{\langle x,\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle}{\langle \nu_{rel}(x),\alpha\rangle \xrightarrow{a} \langle x',\alpha'\rangle} \quad \text{HS-}\underline{30}$$

Continued on Next Page. . .

Table B.1 – Continued ($a \in A, r, s > 0$)

$$\frac{\langle x, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle}{\langle \nu_{rel}(x), \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle} \quad \text{HS-}\underline{31}$$

Table B.2: $BPA_{\text{hs}}^{\text{srt}}$-Rules for Integration ($a \in A, p, q, \geq 0, r > 0$)

$$\frac{\langle F(p), \alpha \rangle \xrightarrow{a} \langle x', \alpha' \rangle, \{\alpha \in [\mathsf{s}(F(q))] \mid q \in U\}}{\langle \int_{u \in U} F(u), \alpha \rangle \xrightarrow{a} \langle x', \alpha' \rangle} \quad p \in U \quad \text{HS-}\underline{32}$$

$$\frac{\langle F(p), \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle, \{\alpha \in [\mathsf{s}(F(q))] \mid q \in U\}}{\langle \int_{u \in U} F(u), \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle} \quad p \in U \quad \text{HS-}\underline{33}$$

$$\frac{\begin{array}{l} \{\langle F(q), \alpha \rangle \overset{r,\rho}{\longmapsto} \langle F_1(q), \alpha' \rangle \mid q \in U_1\}, \\ \vdots \\ \{\langle F(q), \alpha \rangle \overset{r,\rho}{\longmapsto} \langle F_n(q), \alpha' \rangle \mid q \in U_n\}, \\ \{\langle F(q), \alpha \rangle \overset{r}{\not\longmapsto}, \alpha \in [\mathsf{s}(F(q))] \mid q \in U_{n+1}\} \end{array}}{\langle \int_{u \in U} F(u), \alpha \rangle \overset{r,\rho}{\longmapsto} \langle \begin{array}{l} \int_{u \in U_1} F_1(u) + \ldots + \\ \int_{u \in U_n} F_n(u) \, , \; \alpha' \end{array} \rangle} \quad \begin{array}{l} \{U_1, \ldots U_n\} \\ \text{is a partition of} \\ U \backslash U_{n+1}, U_{n+1} \subset U \end{array} \quad \text{HS-}\underline{34}$$

Table B.3: $BPA^{\mathrm{srt}}_{\mathrm{hs}}$-Rules for $\alpha \in [\mathsf{s}(\_)]$ $(a \in A_\delta)$

$\dfrac{}{\alpha \in [\mathsf{s}(\tilde{\tilde{a}})]}$ HS-$\underline{35}$	$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(\sigma^{0}_{\mathsf{rel}}(x))]}$ HS-$\underline{36}$
$\dfrac{r > 0}{\alpha \in [\mathsf{s}(\sigma^{r}_{\mathsf{rel}}(x))]}$ HS-$\underline{37}$	$\dfrac{\alpha \in [\mathsf{s}(x)], \alpha \in [\mathsf{s}(y)]}{\alpha \in [\mathsf{s}(x + y)]}$ HS-$\underline{38}$
$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(x \cdot y)]}$ HS-$\underline{39}$	$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(\psi :\rightarrow x)]}$ HS-$\underline{40}$
$\dfrac{}{\alpha \in [\mathsf{s}(\psi :\rightarrow x)]}$ $\alpha \not\models \psi$ HS-$\underline{41}$	$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(\psi \mathbin{\wedge\!\blacktriangle} x)]}$ $\alpha \models \psi$ HS-$\underline{42}$
$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(\phi \mathbin{\curvearrowright\!\blacktriangledown}_V x)]}$ $\alpha \models \phi$ HS-$\underline{43}$	$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(\chi \mathbin{\ulcorner\!\blacktriangledown} x)]}$ HS-$\underline{44}$
$\dfrac{}{\alpha \in [\mathsf{s}(\chi \mathbin{\ulcorner\!\blacktriangledown} x)]}$ $\alpha \not\models {}^{\circ}\chi$ HS-$\underline{45}$	$\dfrac{\alpha \in [\mathsf{s}(x)]}{\alpha \in [\mathsf{s}(\nu_{rel}(x))]}$ HS-$\underline{46}$
$\dfrac{\{\alpha \in [\mathsf{s}(F(q))] \mid q \in U\}}{\alpha \in [\mathsf{s}(\int_{u \in U} F(u))]}$ HS-$\underline{47}$	

Appendix C

BPA^{srt} with Integration

Table C.1: Rules for Integration for BPA^{srt} from [BM02a] ($a \in A, p, q, \geq 0, r > 0$)

$$\frac{\langle F(p)\rangle \xrightarrow{a} \langle x'\rangle}{\langle \int_{u\in U} F(u)\rangle \xrightarrow{a} \langle x'\rangle} \quad p \in U$$

$$\frac{\langle F(p)\rangle \xrightarrow{a} \langle \surd\rangle}{\langle \int_{u\in U} F(u)\rangle \xrightarrow{a} \langle \surd\rangle} \quad p \in U$$

$$\frac{\begin{array}{l}\{\langle F(q)\rangle \overset{r}{\longmapsto} \langle F_1(q)\rangle \mid q \in U_1\},\\ \ldots\\ \{\langle F(q)\rangle \overset{r}{\longmapsto} \langle F_n(q)\rangle \mid q \in U_n\},\\ \{\langle F(q)\rangle \overset{r}{\not\longmapsto} \mid q \in U_{n+1}\}\end{array}}{\langle \int_{u\in U} F(u)\rangle \overset{r}{\longmapsto} \langle \int_{u\in U_1} F_1(u) + \ldots + \int_{u\in U_n} F_n(u)\rangle} \quad \begin{array}{l}\{U_1, \ldots U_n\}\\ \text{partition of}\\ U\backslash U_{n+1}, U_{n+1} \subset U\end{array}$$

The following axioms have been taken from [BM05]. Here we only give the axioms of integration regarding BPA^{srt} process terms and leave other axioms of integration dealing with operators of BPA^{srt}_{ps} and BPA^{srt}_{hs}.

Table C.2: Axioms for Integration in BPA^{srt} $(p \geq 0)$

$\int_{u \in U} F(u) = \int_{u' \in U} F(w)$	INT1
$\int_{u \in \emptyset} F(u) = \tilde{\tilde{\delta}}$	INT2
$\int_{u \in \{p\}} F(u) = F(p)$	INT3
$\int_{u \in U \cup U'} F(u) = \int_{u \in U} F(u) + \int_{u \in U'} F(u)$	INT4
$U \neq \emptyset \implies \int_{u \in U} x = x$	INT5
$(\forall u \in U \bullet F(u) = G(u)) \implies \int_{u \in U} F(u) = \int_{u \in U} G(u)$	INT6
U, U' unbounded $\implies \int_{u \in U} \sigma^{u}_{\mathsf{rel}}(\tilde{\tilde{\delta}}) = \int_{u \in U'} \sigma^{u}_{\mathsf{rel}}(\tilde{\tilde{\delta}})$	INT8SR
$\sup U = p, p \in U \implies \int_{u \in U} \sigma^{u}_{\mathsf{rel}}(\tilde{\tilde{\delta}}) = \sigma^{p}_{\mathsf{rel}}(\tilde{\tilde{\delta}})$	INT9SR
$\int_{u \in U} (\sigma^{p}_{\mathsf{rel}}(F(u))) = \sigma^{p}_{\mathsf{rel}}(\int_{u \in U} F(u))$	INT10SR
$\int_{u \in U} (F(u) + G(u)) = \int_{u \in U} F(u) + \int_{u \in U} G(u)$	INT11
$\int_{u \in U} (F(u) \cdot x) = (\int_{u \in U} F(u)) \cdot x$	INT12
$\int_{u \in U} \nu_{rel}(F(u)) = \nu_{rel}(\int_{u \in U} F(u))$	INT13

Appendix D

Axioms of BPA_{drt}

The axioms of Basic Process Algebra with discrete relative timing are given below:

Table D.1: Axioms of BPA^{-}_{drt}-ID as in [BMU01] ($a \in A_{\delta}$)

$x + y = y + x$	$A1$	$\sigma(x) + \sigma(y) = \sigma(x + y)$	$DRT1$
$(x + y) + z = x + (y + z)$	$A2$	$\sigma(x) \cdot y = \sigma(x \cdot y)$	$DRT2$
$x + x = x$	$A3$	$\underline{\underline{\delta}} \cdot x = \underline{\underline{\delta}}$	$DRT3$
$(x + y) \cdot z = x \cdot z + y \cdot z$	$A4$	$x + \underline{\underline{\delta}} = x$	$DRT4A$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$A5$		
		$\nu_{rel}(\underline{\underline{a}}) = \underline{\underline{a}}$	$DCS1$
		$\nu_{rel}(x + y) = \nu_{rel}(x) + \nu_{rel}(y)$	$DCS2$
		$\nu_{rel}(x \cdot y) = \nu_{rel}(x) \cdot y$	$DCS3$
		$\nu_{rel}(\sigma(x)) = \underline{\underline{\delta}}$	$DCS4$

Appendix E

Theorem 6

We prove the four conditions given in Theorem 6 one by one. The proof is by structural induction on all closed terms of $BPA^{srt}_{\perp}$.

Let p, p' be closed process terms of $BPA^{srt}_{\perp}$, a be an action, r be a delay duration.

Theorem 6.1

$$(\text{Proposal 1}) \quad \langle \texttt{consistent}\ p \rangle \iff (BPA^{srt}_{hs}) \quad \forall \alpha, \quad \alpha \in [\mathsf{s}(p)]$$

Proof

First we prove the above statement for all constants in $BPA^{srt}_{\perp}$.

1. $p = \tilde{\tilde{a}}$.

   By Rule P1-2:

   $$(\text{Proposal 1}) \quad \langle \texttt{consistent}\ \tilde{\tilde{a}} \rangle$$

   By Rule HS-35, for all α:

   $$(BPA^{srt}_{hs}) \quad \forall \alpha, \quad \alpha \in [\mathsf{s}(\tilde{\tilde{a}})]$$

   Hence the left right implication is proved.

2. $p = \tilde{\tilde{\delta}}$.

   By Rule P1-1:

   $$(\text{Proposal 1}) \quad \langle \texttt{consistent}\ \tilde{\tilde{\delta}} \rangle$$

   By Rule HS-35, for all α:

   $$(BPA^{srt}_{hs}) \quad \forall \alpha, \quad \alpha \in [\mathsf{s}(\tilde{\tilde{\delta}})]$$

   Hence the left right implication is proved.

3. $p = \perp$.

   BPA^{srt}_{hs} : A signal relation for $\perp$ cannot be derived.

   Proposal 1: A consistency predicate for $\perp$ cannot be derived.

   Hence the left right implication is proved.

Next, we prove the given statement for operators $\sigma^0_{\mathsf{rel}}, \sigma^r_{\mathsf{rel}}, \cdot, +, \nu_{rel}$, by structural induction. We give the complete proof for σ^0_{rel}. For proofs of other operators we only mention the rules that have been used.

1. $p = \sigma^0_{\mathsf{rel}}(x)$.

   Suppose

   $$(\text{Proposal } 1) \quad \langle \texttt{consistent } \sigma^0_{\mathsf{rel}}(x) \rangle$$

   This can only be derived by Rule P1-4.

   Then from the premise of the rule:

   $$(\text{Proposal } 1) \quad \langle \texttt{consistent } x \rangle$$

   By Induction, for all α:

   $$(BPA^{srt}_{hs}) \quad \alpha \in [\mathsf{s}(x)]$$

   Apply Rule HS-36, for all α:

   $$(BPA^{srt}_{hs}) \quad \alpha \in [\mathsf{s}(\sigma^0_{\mathsf{rel}}(x))]$$

   <u>Vice Versa</u>

   Suppose, for all α,

   $$(BPA^{srt}_{hs}) \quad \alpha \in [\mathsf{s}(\sigma^0_{\mathsf{rel}}(x))]$$

   This can only be derived by Rule 36.

   Then from the premise of the rule:

   For all α:

   $$(BPA^{srt}_{hs}) \quad \alpha \in [\mathsf{s}(x)]$$

   By Induction:

   $$(\text{Proposal } 1) \quad \langle \texttt{consistent } x \rangle$$

   Apply Rule P1-4:

   $$(\text{Proposal } 1) \quad \langle \texttt{consistent } \sigma^0_{\mathsf{rel}}(x) \rangle$$

2. $p = \sigma^r_{\mathsf{rel}}(x)$.

   By Rule P1-8:

   $$(\text{Proposal } 1) \quad \langle \texttt{consistent } \sigma^r_{\mathsf{rel}}(x) \rangle$$

   By Rule HS-37, for all α:

   $$(BPA^{srt}_{hs}) \quad \forall \alpha, \quad \alpha \in [\mathsf{s}(\sigma^r_{\mathsf{rel}}(x))]$$

   Hence the left right implication is proved.

3. $p = x + y$.

   The proof is by induction using Rules P1-16 and HS-38.

4. $p = x \cdot y$.

   The proof is by induction using Rules P1-12 and HS-39.

5. $p = \nu_{rel}(x)$.

   The proof is by induction using Rules P1-24 and HS-46.

$\boxtimes$

<u>Theorem 6.2</u>

$$\text{(Proposal 1)} \quad \langle x \rangle \xrightarrow{a} \surd \iff (BPA^{srt}_{hs}) \quad \forall \alpha, \alpha' : \quad \langle x, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

<u>Proof</u> First we prove the above statement for all constants in $BPA^{srt}_{\perp}$.

1. $p = \tilde{\tilde{a}}$

   From Rule P1-3:

   $$\text{Proposal 1)} \quad \langle \tilde{\tilde{a}} \rangle \xrightarrow{a} \surd$$

   From Rule HS-1:

   For all α, α'

   $$(BPA^{srt}_{hs}) \quad \langle \tilde{\tilde{a}}, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

   Hence the left right implication is proved.

2. $p = \tilde{\tilde{\delta}}$

   BPA^{srt}_{hs} : A termination step for $\tilde{\tilde{\delta}}$ cannot be derived.

   Proposal 1: A termination step for $\tilde{\tilde{\delta}}$ cannot be derived.

   Hence the left right implication is proved.

3. $p = \perp$

   BPA^{srt}_{hs} : A termination step for $\perp$ cannot be derived.

   Proposal 1: A termination step for $\perp$ cannot be derived.

   Hence the left right implication is proved.

Next, we prove the given statement for operators $\sigma^0_{\mathsf{rel}}, \sigma^r_{\mathsf{rel}}, \cdot, +, \nu_{rel}$, by structural induction. We give the complete proof for σ^0_{rel} and for other operators only mention the rules applied.

1. $p = \sigma^0_{\mathsf{rel}}(x)$

   Suppose,

   $$\text{(Proposal 1)} \quad \langle \sigma^0_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \surd$$

   This can only be derived from Rule P1-5. Hence the following must hold:

   $$\text{(Proposal 1)} \quad \langle x \rangle \xrightarrow{a} \surd$$

By Induction, for all α, α', the following holds:

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

Apply rule HS-3:
For all α, α':

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^{0}(x), \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

<u>Vice Versa</u>
Suppose, for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^{0}(x), \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

This can only be derived from Rule HS-3. Hence the following must hold:
For all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

By Induction:

$$(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \surd$$

Apply Rule P1-5, we get:

$$(\text{Proposal 1}) \quad \langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \xrightarrow{a} \surd$$

Hence the left right implication is proved.

2. $p = \sigma_{\mathsf{rel}}^{r}(x)$
   BPA_{hs}^{srt} : An termination step for $\sigma_{\mathsf{rel}}^{r}(x)$ cannot be derived.
   Proposal 1: An termination step for $\sigma_{\mathsf{rel}}^{r}(x)$ cannot be derived.
   Hence the left right implication is proved.

3. $p = x + y$.
   Suppose,

$$(\text{Proposal 1}) \quad \langle x + y\rangle \xrightarrow{a} \surd \tag{E.1}$$

The above Transition can be derived from two rules.

- Rule P1-17
  Then from the premise of the rule, the following must hold:

$$\begin{array}{l}(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \surd \\ \text{Proposal 1} \quad \langle \texttt{consistent}\ y\rangle\end{array}$$

  By Induction for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

  By Theorem 6.1, for all α:

$$(BPA_{hs}^{srt}) \quad \alpha \in [\mathsf{s}(y)]$$

  Apply rule HS-10 on the above transitions and relations:
  For all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x + y, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

- Rule P1-18
  Same as above.

Vice Versa

Suppose, for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x+y, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle \tag{E.2}$$

The above Transition can be derived from two rules. We discuss these rules one by one:

- Rule HS-10
  Then from the premise of the rule, the following must hold:
  For all α, α':

  $$\begin{array}{ll}(BPA_{hs}^{srt}) & \langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle \\ (BPA_{hs}^{srt}) & \alpha \in [\mathsf{s}(y)]\end{array}$$

  By Induction:

  $$(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \surd$$

  By Theorem 6.1:

  $$\text{Proposal 1} \quad \langle \texttt{consistent} \;\; y\rangle$$

  Apply rule P1-17 on the above transitions and relations:

  $$(\text{Proposal 1}) \quad \langle x+y\rangle \xrightarrow{a} \surd$$

- Rule HS-11
  Same as above.

Hence, left right implication is proved.

4. $p = x \cdot y$

   BPA_{hs}^{srt} : A termination step for $x \cdot y$ cannot be derived.

   Proposal 1: A termination step for $x \cdot y$ cannot be derived.

   Hence, left right implication is proved.

5. $p = \nu_{rel}(x)$.

   Suppose,

   $$(\text{Proposal 1}) \quad \langle \nu_{rel}(x)\rangle \xrightarrow{a} \surd$$

   This can only be derived from Rule P1-25. Hence the following must hold:

   $$(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \surd$$

   By Induction, for all α, α', the following holds:

   $$(BPA_{hs}^{srt}) \quad \langle x, \alpha\rangle \xrightarrow{a} \langle \surd, \alpha'\rangle$$

   Apply rule HS-31:

For all α, α':

$$(BPA_{hs}^{srt}) \quad \langle \nu_{rel}(x), \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

Vice Versa
Suppose, for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle \nu_{rel}(x), \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

This can only be derived from Rule HS-31. Hence the following must hold:

For all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

By Induction:

$$\text{(Proposal 1)} \quad \langle x \rangle \xrightarrow{a} \surd$$

Apply Rule P1-25, we get:

$$\text{(Proposal 1)} \quad \langle \nu_{rel}(x) \rangle \xrightarrow{a} \surd$$

Hence the left right implication is proved.

⊠

Theorem 6.3

$$\text{(Proposal 1)} \quad \langle x \rangle \xrightarrow{a} \langle x' \rangle \iff (BPA_{hs}^{srt}) \quad \forall \alpha, \alpha' : \quad \langle x, \alpha \rangle \xrightarrow{a} \langle x', \alpha' \rangle$$

Proof First we prove the above statement for all constants in $BPA_{\perp}^{srt}$.

1. $p = \tilde{\tilde{a}}$

   BPA_{hs}^{srt} : An action step for $\tilde{\tilde{a}}$ cannot be derived.

   Proposal 1: An action step for $\tilde{\tilde{a}}$ cannot be derived.

   Hence, left right implication is proved.

2. $p = \tilde{\tilde{\delta}}$

   BPA_{hs}^{srt} : A action step for $\tilde{\tilde{\delta}}$ cannot be derived.

   Proposal 1: A action step for $\tilde{\tilde{\delta}}$ cannot be derived.

   Hence, left right implication is proved.

3. $p = \perp$

   BPA_{hs}^{srt} : A action step for $\perp$ cannot be derived.

   Proposal 1: A action step for $\perp$ cannot be derived.

   Hence, left right implication is proved.

Next, we prove the given statement for operators $\sigma_{\mathsf{rel}}^{0}, \sigma_{\mathsf{rel}}^{r}, \cdot, +, \nu_{rel}$, by structural induction.

1. $p = \sigma^0_{\mathsf{rel}}(x)$

   Suppose,

   $$(\text{Proposal 1}) \quad \langle \sigma^0_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p'\rangle$$

   This can only be derived from Rule P1-6. Hence the following must hold:

   $$(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \langle p'\rangle$$

   By Induction, for all α, α', the following holds:

   $$(BPA^{srt}_{hs}) \quad \langle x, \alpha\rangle \xrightarrow{a} \langle p', \alpha'\rangle$$

   Apply rule HS-2:

   For all α, α':

   $$(BPA^{srt}_{hs}) \quad \langle \sigma^0_{\mathsf{rel}}(x), \alpha\rangle \xrightarrow{a} \langle p', \alpha'\rangle$$

   <u>Vice Versa</u>

   Suppose, for all α, α':

   $$(BPA^{srt}_{hs}) \quad \langle \sigma^0_{\mathsf{rel}}(x), \alpha\rangle \xrightarrow{a} \langle p', \alpha'\rangle$$

   This can only be derived from Rule HS-2. Hence the following must hold:

   For all α, α':

   $$(BPA^{srt}_{hs}) \quad \langle x, \alpha\rangle \xrightarrow{a} \langle p', \alpha'\rangle$$

   By Induction:

   $$(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \langle p'\rangle$$

   Apply Rule P1-6, we get:

   $$(\text{Proposal 1}) \quad \langle \sigma^0_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p'\rangle$$

   Hence the left right implication is proved.

2. $p = \sigma^r_{\mathsf{rel}}(x)$

   BPA^{srt}_{hs} : An action step for $\sigma^r_{\mathsf{rel}}(x)$ cannot be derived.

   Proposal 1: An action step for $\sigma^r_{\mathsf{rel}}(x)$ cannot be derived.

   Hence, left right implication is proved.

3. $p = x + y$.

   Suppose,

   $$(\text{Proposal 1}) \quad \langle x + y\rangle \xrightarrow{a} \langle p'\rangle \tag{E.3}$$

   The above Transition can be derived from two rules.

   - Rule P1-19

     Then from the premise of the rule, the following must hold:

     $$\begin{array}{l}(\text{Proposal 1}) \quad \langle x\rangle \xrightarrow{a} \langle p'\rangle \\ \text{Proposal 1} \quad \langle \texttt{consistent } y\rangle\end{array}$$

By Induction for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle$$

By Theorem 6.1, for all α:

$$(BPA_{hs}^{srt}) \quad \alpha \in [\mathsf{s}(y)]$$

Apply rule HS-8 on the above transitions and relations:
For all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x + y, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle$$

- Rule P1-20
  Same as above.

<u>Vice Versa</u>

Suppose, for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x + y, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle \tag{E.4}$$

The above Transition can be derived from two rules. We discuss these rules one by one:

- Rule HS-8
  Then from the premise of the rule, the following must hold:
  For all α, α':

  $$\begin{array}{l} (BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle \\ (BPA_{hs}^{srt}) \quad \alpha \in [\mathsf{s}(y)] \end{array}$$

  By Induction:

  $$(\text{Proposal 1}) \quad \langle x \rangle \xrightarrow{a} \langle p' \rangle$$

  By Theorem 6.1:

  $$\text{Proposal 1} \quad \langle \texttt{consistent} \ \ y \rangle$$

  Apply rule P1-19 on the above transitions and relations:

  $$(\text{Proposal 1}) \quad \langle x + y \rangle \xrightarrow{a} \langle p' \rangle$$

- Rule HS-9
  Same as above.

4. $p = x \cdot y$

   Suppose,

   $$(\text{Proposal 1}) \quad \langle x \cdot y \rangle \xrightarrow{a} \langle p' \rangle \tag{E.5}$$

   This can be derived from two rules:

- Rule P1-15

  Then, for some process term x', $p' = x' \cdot y$, and the following must be derivable:

  $$\text{(Proposal 1)} \quad \langle x \rangle \xrightarrow{a} \langle x' \rangle$$

  By Induction, for all α, α':

  $$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle x', \alpha' \rangle$$

  Apply rule HS-15 on the above transition:

  For all α, α':

  $$(BPA_{hs}^{srt}) \quad \langle x \cdot y, \alpha \rangle \xrightarrow{a} \langle x' \cdot y, \alpha' \rangle$$

- Rule P1-16

  If Transition E.5 is derived from this rule, then, $p' = y$, and the following must be derivable:

  $$\begin{array}{ll} \text{(Proposal 1)} & \langle x \rangle \xrightarrow{a} \surd \\ \text{(Proposal 1)} & \langle \texttt{consistent}\ x \rangle \end{array}$$

  By Theorem 6.2, for all α, α':

  $$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle$$

  By Theorem 6.1, for all α:

  $$(BPA_{hs}^{srt}) \quad \alpha \in [\mathsf{s}(y)]$$

  Apply rule HS-16 on the above transition:

  For all α, α':

  $$(BPA_{hs}^{srt}) \quad \langle x \cdot y, \alpha \rangle \xrightarrow{a} \langle y, \alpha' \rangle$$

Vice Versa

Suppose, for all α, α':

$$(BPA_{hs}^{srt}) \quad \langle x \cdot y, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle \tag{E.6}$$

This can be derived from two rules:

- Rule HS-15

  If Transition E.6 is derived from this rule, then for some process term x', $p' = x' \cdot y$, and the following must be derivable:

  For all α, α':

  $$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle x', \alpha' \rangle$$

  By Induction:

  $$\text{(Proposal 1)} \quad \langle x \rangle \xrightarrow{a} \langle x' \rangle$$

  Apply rule P1-13:

  $$\text{(Proposal 1)} \quad \langle x \cdot y \rangle \xrightarrow{a} \langle x' \cdot y \rangle$$

- Rule HS-16

  If Transition E.6 is derived from this rule, then $p' = y$, and the following must be derivable:

  For all α, α':

  $$\begin{array}{ll} (BPA_{hs}^{srt}) & \langle x, \alpha \rangle \xrightarrow{a} \langle \surd, \alpha' \rangle \\ (BPA_{hs}^{srt}) & \alpha' \in [\mathsf{s}(y)] \end{array}$$

  By Theorem 6.2:

  $$(\text{Proposal 1}) \quad \langle x \rangle \xrightarrow{a} \surd$$

  By Theorem 6.1:

  $$(\text{Proposal 1}) \quad \alpha \in [\mathsf{s}(y)]$$

  Apply rule P1-14:

  $$(\text{Proposal 1}) \quad \langle x \cdot y \rangle \xrightarrow{a} \langle y \rangle$$

Hence the left right implication is proved.

5. $p = \nu_{rel}(x)$.

   Suppose,

   $$(\text{Proposal 1}) \quad \langle \nu_{rel}(x) \rangle \xrightarrow{a} \langle p' \rangle$$

   This can only be derived from Rule P1-26. Hence the following must hold:

   $$(\text{Proposal 1}) \quad \langle x \rangle \xrightarrow{a} \langle p' \rangle$$

   By Induction, for all α, α', the following holds:

   $$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle$$

   Apply rule HS-30:

   For all α, α':

   $$(BPA_{hs}^{srt}) \quad \langle \nu_{rel}(x), \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle$$

   Vice Versa

   Suppose, for all α, α':

   $$(BPA_{hs}^{srt}) \quad \langle \nu_{rel}(x), \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle$$

   This can only be derived from Rule HS-30. Hence the following must hold:

   For all α, α':

   $$(BPA_{hs}^{srt}) \quad \langle x, \alpha \rangle \xrightarrow{a} \langle p', \alpha' \rangle$$

   By Induction:

   $$(\text{Proposal 1}) \quad \langle x \rangle \xrightarrow{a} \langle p' \rangle$$

   Apply Rule P1-26, we get:

   $$(\text{Proposal 1}) \quad \langle \nu_{rel}(x) \rangle \xrightarrow{a} \langle p' \rangle$$

   Hence the left right implication is proved.

⊠

Theorem 6.4

$$(\text{Proposal 1})\quad \langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle \iff (BPA^{srt}_{hs})\quad \forall\rho,\quad \langle x,\alpha^{\rho}_{0}\rangle \xrightarrow{r,\rho} \langle x',\alpha^{\rho}_{r}\rangle$$

Proof First we prove the above statement for all constants in $BPA^{srt}_{\perp}$.

1. $p = \tilde{\tilde{a}}$

   BPA^{srt}_{hs} : A time step for $\tilde{\tilde{a}}$ cannot be derived.

   Proposal 1: A time step for $\tilde{\tilde{a}}$ cannot be derived.

   Hence, left right implication is proved.

2. $p = \tilde{\tilde{\delta}}$

   BPA^{srt}_{hs} : A time step for $\tilde{\tilde{\delta}}$ cannot be derived.

   Proposal 1: A time step for $\tilde{\tilde{\delta}}$ cannot be derived.

   Hence, left right implication is proved.

3. $p = \perp$

   BPA^{srt}_{hs} : A time step for $\perp$ cannot be derived.

   Proposal 1: A time step for $\perp$ cannot be derived.

   Hence, left right implication is proved.

Next, we prove the given statement for operators $\sigma^{0}_{\mathsf{rel}}, \sigma^{r}_{\mathsf{rel}}, \cdot, +, \nu_{rel}$, by structural induction.

1. $p = \sigma^{0}_{\mathsf{rel}}(x)$

   Suppose, for all ρ:

   $$(BPA^{srt}_{hs})\quad \langle \sigma^{0}_{\mathsf{rel}}(x), \alpha^{\rho}_{0}\rangle \xrightarrow{r,\rho} \langle p', \alpha^{\rho}_{r}\rangle$$

   This can only be derived from Rule HS-4. Hence the following must hold:

   For all ρ:

   $$(BPA^{srt}_{hs})\quad \langle x, \alpha^{\rho}_{0}\rangle \xrightarrow{r,\rho} \langle p', \alpha^{\rho}_{r}\rangle$$

   By Induction:

   $$(\text{Proposal 1})\quad \langle x\rangle \overset{r}{\longmapsto} \langle p'\rangle$$

   Apply Rule P1-7, we get:

   $$(\text{Proposal 1})\quad \langle \sigma^{0}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle p'\rangle$$

   Vice Versa

   Suppose,

   $$(\text{Proposal 1})\quad \langle \sigma^{0}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle p'\rangle$$

   This can only be derived from Rule P1-7. Hence the following must hold:

   $$(\text{Proposal 1})\quad \langle x\rangle \overset{r}{\longmapsto} \langle p'\rangle$$

By Induction, for all ρ, the following holds:

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle p', \alpha_r^\rho\rangle$$

Apply rule HS-4:

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^0(x), \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle p', \alpha_r^\rho\rangle$$

Hence the left right implication is proved.

2. $p = \sigma_{\mathsf{rel}}^r(x)$

   Suppose, for all ρ:

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^r(x), \alpha_0^\rho\rangle \xmapsto{t,\rho} \langle p', \alpha_t^\rho\rangle \tag{E.7}$$

- Case $t < r$:

  Let $r = u + t$, for some $u > 0$. Then Transition E.7 must be derived from Rule HS-5. Then $p' = \sigma_{\mathsf{rel}}^u(x)$. Rule HS-5 can always be applied.

  Hence for all ρ:

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^{u+t}(x), \alpha_0^\rho\rangle \xmapsto{t,\rho} \langle \sigma_{\mathsf{rel}}^u(x), \alpha_t^\rho\rangle$$

  In Proposal 1, by Rule P1-9, the following is derivable:

$$\text{Proposal} \quad \langle \sigma_{\mathsf{rel}}^{u+t}(x)\rangle \overset{t}{\mapsto} \langle \sigma_{\mathsf{rel}}^u(x)\rangle$$

- Case $t = r$:

  This can only be derived from Rule HS- 6. Then $p' = x$ in Transition E.7. Rewriting Transition E.7:

  For all ρ:

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^t(x), \alpha_0^\rho\rangle \xmapsto{t,\rho} \langle x, \alpha_t^\rho\rangle$$

  From the premise of rule HS-6, for all α_t^ρ:

$$\alpha_t^\rho \in [\mathsf{s}(x)]$$

  Since there is no restriction on ρ and hence on α_t^ρ, therefore we have:

  For all α

$$(BPA_{hs}^{srt}) \quad \alpha \in [\mathsf{s}(x)]$$

  By Theorem 6.1:

$$\text{Proposal 1} \quad \langle \texttt{consistent} \ x\rangle$$

  Then by Rule P1-10, the following is derivable:

$$\text{Proposal1} \quad \langle \sigma_{\mathsf{rel}}^t(x)\rangle \overset{t}{\mapsto} \langle x\rangle$$

- Case $t > r$.

  Let $t = r + r_1$, for some $r_1 > 0$. Rewriting Transition E.7:

  For all ρ:

$$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^r(x), \alpha_0^\rho\rangle \xmapsto{r+r_1,\rho} \langle p', \alpha_t^\rho\rangle \tag{E.8}$$

This can only be derivable from Rule HS-7. Hence, the premise of the rule must hold. From Premise of the Rule HS-7, Transition E.8 can only be derived if the following holds:

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha_r^\rho \rangle \xmapsto{r_1, \rho \trianglerighteq r} \langle p', \alpha_t^\rho \rangle$$

For the definition of symbol $\rho \trianglerighteq r$, see Appendix B. Briefly, $\rho \trianglerighteq r$ denotes the state evolution ρ after r time units have elapsed. If the time interval of ρ is $[0, r + r_1]$, then the time interval of $\rho \trianglerighteq r$ is $[0, r_1]$. As there are no restrictions on ρ, therefore there are no restrictions on $\rho \trianglerighteq r$.

By Structural Induction:

$$\text{Proposal1} \quad \langle x \rangle \xmapsto{r_1} \langle p' \rangle$$

Apply Rule P1-11, the following holds:

$$\text{Proposal1} \quad \langle \sigma_{\mathsf{rel}}^r(x) \rangle \xmapsto{r+r_1} \langle p' \rangle$$

Vice versa

Suppose,

$$(\text{Proposal 1}) \quad \langle \sigma_{\mathsf{rel}}^r(x) \rangle \xmapsto{t} \langle p' \rangle \tag{E.9}$$

- Case $t < r$:

  Let $r = u + t$, for some $u > 0$. The Transition E.9 must have been derived from Rule P1-9 and $p' = \sigma_{\mathsf{rel}}^u(x)$.

  $$(\text{Proposal 1}) \quad \langle \sigma_{\mathsf{rel}}^{u+t}(x) \rangle \xmapsto{t} \langle \sigma_{\mathsf{rel}}^u(x) \rangle$$

  In BPA_{hs}^{srt}, by Rule HS-5, the following can be derived for all ρ:

  $$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^{u+t}(x), \alpha_0^\rho \rangle \xmapsto{t, \rho} \langle \sigma_{\mathsf{rel}}^u(x), \alpha_t^\rho \rangle$$

- Case $t = r$:

  This can only be derived from Rule P1-10. Then $p' = x$ in Transition E.9.

  $$(\text{Proposal 1}) \quad \langle \sigma_{\mathsf{rel}}^t(x) \rangle \xmapsto{t} \langle x \rangle$$

  From the premise of the rule,

  $$\langle \texttt{consistent } x \rangle$$

  By Theorem6.1, for all α:

  $$BPA_{hs}^{srt} \quad \alpha \in [\mathsf{s}(x)]$$

  Then by Rule HS-6, the following is derivable:

  For all ρ:

  $$(BPA_{hs}^{srt}) \quad \langle \sigma_{\mathsf{rel}}^t(x), \alpha_0^\rho \rangle \xmapsto{t, \rho} \langle x, \alpha_t^\rho \rangle$$

- Case $t > r$.
  Let $t = r + r_1$, for some $r_1 > 0$. Rewriting Transition E.9:

  $$(\text{Proposal 1}) \quad \langle \sigma^r_{\mathsf{rel}}(x) \rangle \xmapsto{r+r_1} \langle p' \rangle$$

  This can only be derivable from Rule P1-11. Hence, the premise of the rule must hold:

  $$(\text{Proposal 1}) \quad \langle x \rangle \xmapsto{r_1} \langle p' \rangle$$

  By Structural Induction, for all ρ

  $$(BPA^{srt}_{hs}) \quad \langle x, \alpha^{\rho}_0 \rangle \xmapsto{r_1,\rho} \langle p', \alpha^{\rho}_{r_1} \rangle$$

  Apply Rule HS-7, the following holds:

  $$(BPA^{srt}_{hs}) \quad \langle \sigma^r_{\mathsf{rel}}(x), \alpha^{\rho'}_0 \rangle \xmapsto{r+r_1,\rho'} \langle p', \alpha^{\rho'}_t \rangle$$

  where $\rho = \rho' \trianglerighteq r$. For the definition of the state evolution $\rho = \rho' \trianglerighteq r$, see Appendix B. Briefly, $\rho' \trianglerighteq r$ is the state evolution ρ after the passage of r time units. Since there is no restrictions on ρ, hence there is no restriction on ρ'. Hence, we can write:
  For all ρ':

  $$(BPA^{srt}_{hs}) \quad \langle \sigma^r_{\mathsf{rel}}(x), \alpha^{\rho'}_0 \rangle \xmapsto{r+r_1,\rho'} \langle p', \alpha^{\rho'}_t \rangle$$

Hence, left right implication is proved.

3. $p = x + y$.
   Suppose,

   $$(\text{Proposal 1}) \quad \langle x + y \rangle \xmapsto{r} \langle p' \rangle \tag{E.10}$$

   The above Transition can be derived from three rules. We discuss these rules one by one:

   - Rule P1-21:
     Then for some process term x', y', p' in Transition E.10 is $x' + y'$. Rewriting Transition E.10:

     $$(\text{Proposal 1}) \quad \langle x + y \rangle \xmapsto{r} \langle x' + y' \rangle \tag{E.11}$$

     From the premise of Rule P1-21:

     $$\begin{array}{ll} (\text{Proposal 1}) & \langle x \rangle \xmapsto{r} \langle x' \rangle \\ (\text{Proposal 1}) & \langle y \rangle \xmapsto{r} \langle y' \rangle \end{array}$$

     By Induction, for all ρ:

     $$\begin{array}{ll} (BPA^{srt}_{hs}) & \langle x, \alpha^{\rho}_0 \rangle \xmapsto{r,\rho} \langle x', \alpha^{\rho}_r \rangle \\ (BPA^{srt}_{hs}) & \langle y, \alpha^{\rho}_0 \rangle \xmapsto{r,\rho} \langle y', \alpha^{\rho}_r \rangle \end{array}$$

     Apply rule HS-14 on the above transitions:
     For all ρ:

     $$(BPA^{srt}_{hs}) \quad \langle x + y, \alpha^{\rho}_0 \rangle \xmapsto{r,\rho} \langle x' + y', \alpha^{\rho}_r \rangle$$

- Rule P1-22

  Then from the premise of the rule, the following must hold:

  $$\begin{array}{ll} \text{(Proposal 1)} & \langle x\rangle \overset{r}{\longmapsto} \langle p'\rangle \\ \text{Proposal 1} & \langle \texttt{consistent}\ y\rangle \\ \text{Proposal 1} & \langle y\rangle \overset{r}{\not\longmapsto} \end{array}$$

  By Induction for all ρ:

  $$\begin{array}{ll} (BPA^{srt}_{hs}) & \langle x, \alpha^{\rho}_{0}\rangle \xrightarrow{r,\rho} \langle p', \alpha^{\rho}_{r}\rangle \\ (BPA^{srt}_{hs}) & \langle y, \alpha^{\rho}_{0}\rangle \overset{r}{\not\longmapsto} \end{array}$$

  By Theorem 6.1, for all α:

  $$(BPA^{srt}_{hs}) \quad \alpha \in [\mathsf{s}(y)]$$

  Apply rule HS-12 on the above transitions and relations:
  For all ρ:

  $$(BPA^{srt}_{hs}) \quad \langle x+y, \alpha^{\rho}_{0}\rangle \xrightarrow{r,\rho} \langle p', \alpha^{\rho}_{r}\rangle$$

- Rule P1-23

  Same as above.

<u>Vice Versa</u>

Suppose, for all ρ:

$$(BPA^{srt}_{hs}) \quad \langle x+y, \alpha^{\rho}_{0}\rangle \overset{r}{\longmapsto} \langle p', \alpha^{\rho}_{r}\rangle \tag{E.12}$$

The above Transition can be derived from three rules. We discuss these rules one by one:

- Rule HS-14:

  Then for some process term x', y', p' in Transition E.12 is $x' + y'$. Rewriting Transition E.12:
  For all ρ:

  $$(BPA^{srt}_{hs}) \quad \langle x+y, \alpha^{\rho}_{0}\rangle \overset{r}{\longmapsto} \langle x'+y', \alpha^{\rho}_{r}\rangle \tag{E.13}$$

  From the premise of Rule HS-14:
  For all ρ:

  $$\begin{array}{ll} (BPA^{srt}_{hs}) & \langle x, \alpha^{\rho}_{0}\rangle \overset{r}{\longmapsto} \langle x', \alpha^{\rho}_{r}\rangle \\ (BPA^{srt}_{hs}) & \langle y, \alpha^{\rho}_{0}\rangle \overset{r}{\longmapsto} \langle y', \alpha^{\rho}_{r}\rangle \end{array}$$

  By Induction:

  $$\begin{array}{ll} \text{(Proposal 1)} & \langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle \\ \text{(Proposal 1)} & \langle y,\rangle \overset{r}{\longmapsto} \langle y'\rangle \end{array}$$

  Apply rule P1-21 on the above transitions:

  $$\text{(Proposal 1)} \quad \langle x+y\rangle \overset{r}{\longmapsto} \langle x'+y'\rangle$$

- Rule HS-12

  Then from the premise of the rule, the following must hold:

  For all ρ

  $$\begin{array}{ll} (BPA_{hs}^{srt}) & \langle x, \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle p', \alpha_r^\rho\rangle \\ (BPA_{hs}^{srt}) & \alpha_0^\rho \in [\mathsf{s}(y)] \\ (BPA_{hs}^{srt}) & \langle y, \alpha_0^\rho\rangle \overset{r}{\not\mapsto} \end{array}$$

  By Induction:

  $$\begin{array}{ll} (\text{Proposal } 1) & \langle x\rangle \overset{r}{\mapsto} \langle p'\rangle \\ \text{Proposal } 1 & \langle y\rangle \overset{r}{\not\mapsto} \end{array}$$

  By Theorem 6.1:

  $$\text{Proposal } 1 \quad \langle \texttt{consistent } y\rangle$$

  Apply rule P1-22 on the above transitions and relations:

  $$(\text{Proposal } 1) \quad \langle x + y\rangle \overset{r}{\mapsto} \langle p'\rangle$$

- Rule HS-13

  Same as above.

Hence, left right implication is proved.

4. $p = x \cdot y$

Suppose,

$$(\text{Proposal } 1) \quad \langle x \cdot y\rangle \overset{r}{\mapsto} \langle p'\rangle$$

This can only be derived from Rule P1-15. Then, for some process term x', $p' = x' \cdot y$, and the following must be derivable:

$$(\text{Proposal } 1) \quad \langle x\rangle \overset{r}{\mapsto} \langle x'\rangle$$

By Induction, for all ρ:

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle x', \alpha_r^\rho\rangle$$

Apply rule HS-17 on the above transition:

For all ρ:

$$(BPA_{hs}^{srt}) \quad \langle x \cdot y, \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle x' \cdot y, \alpha_r^\rho\rangle$$

Vice Versa

Suppose, for all ρ:

$$(BPA_{hs}^{srt}) \quad \langle x \cdot y, \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle p', \alpha_r^\rho\rangle$$

This can only be derived from rule HS-17. Hence for some process term x', $p' = x' \cdot y$, and the following must be derivable:

For all ρ:

$$(BPA_{hs}^{srt}) \quad \langle x, \alpha_0^\rho\rangle \xmapsto{r,\rho} \langle x', \alpha_r^\rho\rangle$$

By Induction:

$$(\text{Proposal } 1) \quad \langle x\rangle \overset{r}{\mapsto} \langle x'\rangle$$

Apply rule P1-15:

$$(\text{Proposal } 1) \quad \langle x \cdot y\rangle \overset{r}{\mapsto} \langle x' \cdot y\rangle$$

Hence, left right implication is proved.

5. $p = \nu_{rel}(x)$.

   BPA^{srt}_{hs} : A time step for $\nu_{rel}(x)$ cannot be derived.

   Proposal 1: A time step for $\nu_{rel}(x)$ cannot be derived.

   Hence, left right implication is proved.

⊠

Appendix F

Theorem 7

Thoerem 7

Axiom SRT3 is sound in the semantics of Section 3.6.2.

$$\sigma^{v}_{\mathsf{rel}}(x) + \sigma^{v}_{\mathsf{rel}}(y) \leftrightarroweq \sigma^{v}_{\mathsf{rel}}(x+y) \qquad (SRT3)$$

where $v \geq 0$

Proof

We prove the soundness of Axiom SRT3 in two steps.

Case $u = 0$

From Rules AC-4, AC-5, AC-6 and AC-7, it easy to prove the following holds in the semantics of $BPA^{srt}_{\perp}$ with modified Alternative Composition (Section 3.6.2):

For any process term x,

$$\sigma^{0}_{\mathsf{rel}}(x) \leftrightarroweq x$$

Since Bisimulation is a congruence therefore, then it becomes trivial to prove that:

$$\sigma^{0}_{\mathsf{rel}}(x) + \sigma^{0}_{\mathsf{rel}}(y) \leftrightarroweq \sigma^{0}_{\mathsf{rel}}(x+y)$$

Case $u > 0$

Let $\mathcal{I}$ be the following relation:

$$\mathcal{I} = \{(p, p) \mid p \in P\}$$

Let R be the following relation:

$$R = \{(\sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)), \sigma^{t}_{\mathsf{rel}}(x+y) \mid 0 < t \leq v, x, y \in P\}$$

We prove that $R \cup \mathcal{I}$ is a bisimulation relation:

For all $a \in A, r > 0, z \in P$:

1. $$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \langle z \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^{t}_{\mathsf{rel}}(x+y) \rangle \xrightarrow{a} \langle z' \rangle \\ (z, z') \in R \cup \mathcal{I} \end{array}$$

   Trivial.

2.

$$\langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \xrightarrow{a} \langle z\rangle \implies \begin{array}{l}\exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xrightarrow{a} \langle z'\rangle \\ (z', z) \in R \cup \mathcal{I}\end{array}$$

Trivial.

3.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xrightarrow{a} \surd \iff \langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \xrightarrow{a} \surd$$

Trivial.

4.

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \iff \langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x+y)\rangle$$

Trivial.

5.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \implies \begin{array}{l}\exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle z'\rangle \\ (z, z') \in R \cup \mathcal{I}\end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \tag{F.1}$$

This can be derived from Rules AC-19, AC-20, AC-21 and AC-26.

(a) Rule AC-19

Then $z = z_1 + z_2$.
Rewriting Transition F.1:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z_1 + z_2\rangle \tag{F.2}$$

From premise of the rule, the following holds:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z_1\rangle \tag{F.3}$$

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z_2\rangle \tag{F.4}$$

We distinguish between three cases for different values of r:

i. **Case $r < t$**

Let for some $0 < r_1 < t$,

$$t = r + r_1 \tag{F.5}$$

Rewriting Transitions F.2, F.3 and F.4:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) + \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z_1 + z_2\rangle \tag{F.6}$$

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z_1\rangle \tag{F.7}$$

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z_2\rangle \tag{F.8}$$

Then Transitions F.7 and F.8 can be derived from Rules AC-8 and AC-26. That gives us four cases:

A. Transitions F.7 and F.8 are derived from Rule AC-8.

B. Transition F.7 is derived from Rule AC-8 and Transition F.8 is derived Rule AC-26.

C. Transition F.7 is derived from Rule AC-26 and Transition F.8 is derived Rule AC-8.

D. Transitions F.7 and F.8 are derived from Rule AC-26.

We prove that in all four cases, the target process terms z_1 and z_2 are as follows:

$$z_1 = \sigma^{r_1}_{\mathsf{rel}}(x) \text{ and } z_2 = \sigma^{r_1}_{\mathsf{rel}}(y)$$

In case Rule AC-8 is used to derive Transition F.7 (or Transition F.8), it is easy to see that $z_1 = \sigma^{r_1}_{\mathsf{rel}}(x)$ ($z_2 = \sigma^{r_1}_{\mathsf{rel}}(y)$).

Below we argue the cases when Rule AC-26 is used to derive one or both of the Transitions F.7 and F.8.

A. **Transition F.7 by Rule AC-26**

Suppose this rule is used to derive Transition F.7. By Rule AC-26, we can combine successive time transitions into a single time transition. For a derivable time transition, the process of applying Rule AC-26 must be finite. Hence, we can say that there exists an $n > 1$ such that Transition F.7 is obtained by combining n successive transitions. Each of the n transitions has been derived by rules other than Rule AC-26. Note that n is taken to be greater than 1 because one application of Rule AC-26 joins two successive transitions.

Let $s_1, \ldots, s_n$ denote the durations of the constituent time transitions and let $p_1, \ldots, p_{n-1}$ denote the intermediate process terms.

Splitting Transition F.7 into n transitions:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.9}$$

and

$$s_1 + \ldots + s_n = r \tag{F.10}$$

From F.5 and F.10 we infer that:

$$t = s_1 + \ldots + s_n + r_1$$

Rewriting Transition F.9 by replacing t by the sum of durations:

$$\langle \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle$$

Consider the first transition

$$\langle \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \tag{F.11}$$

For a process term $\sigma^{u}_{\mathsf{rel}}(x)$, with $u > 0$, only two rules (other than Rule AC- 26) are applicable. A time step of duration $v < u$ can only be derived by Rule AC-8 and a time step of duration u can only be derived from

Rule AC-9. Transition F.11 can only be derived from Rule AC-8 as $s_1 < (s_1 + \ldots + s_n + r_1)$. From the rule we infer that:

$$p_1 = \sigma_{\mathsf{rel}}^{s_2+\ldots+s_n+r_1}(x)$$

Rewriting Transition F.9 by replacing p_1 by $\sigma_{\mathsf{rel}}^{s_2+\ldots+s_n+r_1}(x)$:

$$\langle\sigma_{\mathsf{rel}}^{s_1+\ldots+s_n+r_1}(x)\rangle \xmapsto{s_1} \langle\sigma_{\mathsf{rel}}^{s_2+\ldots+s_n+r_1}(x)\rangle \xmapsto{s_2} \ldots \langle p_{n-1}\rangle \xmapsto{s_n} \langle z_1\rangle$$

Again the second transition

$$\langle\sigma_{\mathsf{rel}}^{s_2+\ldots+s_n+r_1}(x)\rangle \xmapsto{s_2} \langle p_2\rangle$$

can only be derived from Rule AC-8 as $s_2 < (s_2 + \ldots + s_n + r_1)$. From the rule we infer that:

$$p_2 = \sigma_{\mathsf{rel}}^{s_3+\ldots+s_n+r_1}(x)$$

Continuing the same reasoning, we infer that all the n time steps of Transition F.9 have been derived from Rule AC-8 and the target of the $(n-1)$ time step in Transition F.9 is as follows:

$$p_{n-1} = \sigma_{\mathsf{rel}}^{s_n+r_1}(x)$$

Rewriting the n^{th} transition of Transition F.9:

$$\langle\sigma_{\mathsf{rel}}^{s_n+r_1}(x)\rangle \xmapsto{s_n} \langle z_1\rangle$$

The above transition is derived from Rule AC-8. Then z_1 must be of the following form:

$$z_1 = \sigma_{\mathsf{rel}}^{r_1}(x) \tag{F.12}$$

B. **Transition F.8 by Rule AC-26**

By reasoning given for Transition F.7, we can say that there exists an $m > 1$ such that Transition F.8 is obtained by combining m successive transitions. Each of the m transitions has been derived by rules other than Rule AC-26. Let $u_1, \ldots, u_m$ denote the durations of the constituent time transitions and let $q_1, \ldots, q_{m-1}$ denote the intermediate process terms.

Splitting Transition F.8 into m transitions:

$$\langle\sigma_{\mathsf{rel}}^{r+r_1}(y)\rangle \xmapsto{u_1} \langle q_1\rangle \xmapsto{u_2} \ldots \langle q_{m-1}\rangle \xmapsto{u_m} \langle z_2\rangle \tag{F.13}$$

and

$$u_1 + \ldots u_m = r \tag{F.14}$$

From F.5 and F.14, we infer that:

$$t = u_1 + \ldots + u_m + r_1$$

Rewriting Transition F.13 by replacing t by the sum of durations:

$$\langle\sigma_{\mathsf{rel}}^{u_1+\ldots+u_m+r_1}(y)\rangle \xmapsto{u_1} \langle q_1\rangle \xmapsto{u_2} \ldots \langle q_{m-1}\rangle \xmapsto{u_m} \langle z_2\rangle$$

By the same reasoning as applied for Transition F.7, we can infer the following:

- All the constituent transitions of Transition F.13 have been derived by Rule AC-8.
- The intermediate process terms $q_1, \ldots, q_{m-1}$, and the final process term are as follows:

$$q_1 = \sigma_{\mathsf{rel}}^{u_2+\ldots+u_m+r_1}(y) \tag{F.15}$$

$$q_2 = \sigma_{\mathsf{rel}}^{u_3+\ldots+u_m+r_1}(y) \tag{F.16}$$

$$\vdots$$

$$q_{m-1} = \sigma_{\mathsf{rel}}^{u_m+r_1}(y) \tag{F.17}$$

$$z_2 = \sigma_{\mathsf{rel}}^{r_1}(y) \tag{F.18}$$

Putting the values of z_1 and z_2 from equations F.12 and F.18 in Transition F.6, we get:

$$\langle \sigma_{\mathsf{rel}}^{r+r_1}(x) + \sigma_{\mathsf{rel}}^{r+r_1}(y) \rangle \overset{r}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y) \rangle \tag{F.19}$$

Again, Rule 8 can derive the following:

$$\langle \sigma_{\mathsf{rel}}^{r+r_1}(x+y) \rangle \overset{r}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{r_1}(x+y) \rangle \tag{F.20}$$

Consider Transitions F.19 and F.20. For $0 < r_1 < t$, the pair $(\sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y), \sigma_{\mathsf{rel}}^{r_1}(x+y)) \in R$.

ii. **Case $r = t$**

Then Transitions F.3 and F.4 can only be derived from Rules AC-9 and AC-26. Rewriting Transitions F.2, F.3 and F.4:

$$\langle \sigma_{\mathsf{rel}}^{r}(x) + \sigma_{\mathsf{rel}}^{r}(y) \rangle \overset{r}{\longmapsto} \langle z_1 + z_2 \rangle \tag{F.21}$$

$$\langle \sigma_{\mathsf{rel}}^{r}(x) \rangle \overset{r}{\longmapsto} \langle z_1 \rangle \tag{F.22}$$

$$\langle \sigma_{\mathsf{rel}}^{r}(y) \rangle \overset{r}{\longmapsto} \langle z_2 \rangle \tag{F.23}$$

Then Transitions F.22 and F.23 can be derived from Rules AC-9 and AC-26. That again gives us four cases:

A. Transitions F.22 and F.23 are derived from Rule AC-9.

B. Transition F.22 is derived from Rule AC-9 and Transition F.23 is derived Rule AC-26.

C. Transition F.22 is derived from Rule AC-26 and Transition F.23 is derived Rule AC-9.

D. Transitions F.22 and F.23 are derived from Rule AC-26.

We prove that in all four cases, the target process terms z_1 and z_2 are as follows:

$$z_1 = x \text{ and } z_2 = y$$

And the following holds:

$$\langle \texttt{consistent}\ z_1 \rangle \text{ and } \langle \texttt{consistent}\ y \rangle$$

In case Rule AC-9 is used to derive Transition F.22 (or Transition F.23), it is easy to see that $z_1 = x$ ($z_2 = y$). From the premise of the rule $\langle \texttt{consistent}\ x \rangle$ ($\langle \texttt{consistent}\ y \rangle$) holds.
Below we argue the cases when Rule AC-26 is used to derive one or both of the Transitions F.22 and F.23.

A. **Transition F.22 by Rule AC-26**
By reasoning given for Transition F.7, we can say that there exists an $n > 1$ such that Transition F.22 is obtained by combining n successive transitions. Each of the n transitions has been derived by rules other than Rule AC-26. Let $s_1, \ldots, s_n$ denote the durations of the constituent time transitions and let $p_1, \ldots, p_{n-1}$ denote the intermediate process terms. Splitting Transition F.22 into n transitions:

$$\langle \sigma^r_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_2 \rangle \tag{F.24}$$

and

$$s_1 + \ldots s_n = r \tag{F.25}$$

From F.25 and the fact that we are considering the case for $r = t$, we infer that:

$$t = s_1 + \ldots + s_n$$

Rewriting Transition F.24 by replacing t by the sum of durations:

$$\langle \sigma^{s_1 + \ldots s_n}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_2 \rangle \tag{F.26}$$

Now , for a process term $\sigma^u_{\mathsf{rel}}(x)$, with $u > 0$, only two rules (other than Rule AC- 26) are applicable. A time step of duration $v < u$ can only be derived by Rule AC-8 and a time step of duration u can only be derived from Rule AC-9.
Consider the first time step of Transition F.26:

$$\langle \sigma^{s_1 + \ldots + s_n}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \tag{F.27}$$

We know that $n > 1$, as Rule AC-26 applied once joins two transitions. For $n > 1$, $s_1 < (s_1 + \ldots + s_n)$, therefore Transition F.27 can only be derived from Rule AC-8. From the rule we infer that:

$$p_1 = \sigma^{s_2 + \ldots + s_n}_{\mathsf{rel}}(x)$$

Rewriting Transition F.26 by replacing p_1 by $\sigma^{s_2 + \ldots + s_n}_{\mathsf{rel}}(x)$:

$$\langle \sigma^{s_1 + \ldots + s_n}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle \sigma^{s_2 + \ldots + s_n}_{\mathsf{rel}}(x) \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle$$

The n^{th} transition will be the final one with its target equal to z_1.

$$\langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.28}$$

Extending the reasoning given above for process term p_1 to other intermediate process terms, we infer the following:

$$\begin{array}{lcll} p_2 & = & \sigma_{\mathsf{rel}}^{s_3+\ldots+s_n}(x) & \text{if } n > 2 \\ p_3 & = & \sigma_{\mathsf{rel}}^{s_4+\ldots+s_n}(x) & \text{if } n > 3 \\ \vdots & & & \\ p_{n-1} & = & \sigma_{\mathsf{rel}}^{s_n}(x) & \end{array}$$

Putting the value of p_{n-1} in Transition F.28, we get:

$$\langle \sigma_{\mathsf{rel}}^{s_n}(x) \rangle \xmapsto{s_n} \langle z_1 \rangle \qquad \text{(F.29)}$$

As the delay duration is equal to the duration of the relative delay operator, therefore the above transition can only be derived from Rule AC-9. Then z_1 is equal to x,

$$z_1 = x \qquad \text{(F.30)}$$

And from the premise of Rule AC-9

$$\langle \texttt{consistent}\ x \rangle \qquad \text{(F.31)}$$

B. **Transition F.23 by Rule AC-26**

By similar reasoning as given above for Transition F.22, we infer that the following holds:

$$z_2 = y \qquad \text{(F.32)}$$

$$\langle \texttt{consistent}\ x \rangle \qquad \text{(F.33)}$$

Putting the values of z_1 and z_2 from equations F.30 and F.32 in Transition F.21, we get:

$$\langle \sigma_{\mathsf{rel}}^{r}(x) + \sigma_{\mathsf{rel}}^{r}(y) \rangle \xmapsto{r} \langle x + y \rangle \qquad \text{(F.34)}$$

Again, using Predicates F.31 and F.33, Rule AC-9 can derive the following:

$$\langle \sigma_{\mathsf{rel}}^{r}(x + y) \rangle \xmapsto{r} \langle x + y \rangle \qquad \text{(F.35)}$$

Consider Transitions F.34 and F.35. The pair $(x + y, x + y) \in \mathcal{I}$.

iii. **Case $r > t$**

If $r > t$, then Transitions F.3 and F.4 can only be derived from Rule AC-26.

A. **Transition F.3 by Rule AC-26**

By reasoning given above for derivation of Transitions F.7, F.8, F.22 and F.23 using Rule AC-26, we say that there exists $n > 1$, such that Transition F.3 is obtained from Rule 26 by combining n successive transitions. Each of the n transitions has been derived by rules other than Rule AC-26. Let $s_1, \ldots, s_n$ denote the durations of the constituent time transitions and let $p_1, \ldots, p_{n-1}$ denote the intermediate process terms.

Splitting Transition F.3 into n transitions:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.36}$$

and

$$s_1 + \ldots + s_n = r \tag{F.37}$$

From F.37 and the fact that we are considering the case with $r > t$, we infer that:

$$s_1 + \ldots + s_n > t \tag{F.38}$$

In each of the constituent transitions of Transition F.36, a single rule other than Rule AC-26 has been applied.
There are only two rules applicable on $\sigma^{t}_{\mathsf{rel}}(x)$, Rule AC-8 and Rule AC-9. Consider the first time step of Transition F.36.

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle p_1 \rangle \tag{F.39}$$

Two cases arise. In Transition F.39, $s_1 < t$ or $s_1 = t$. The case $s_1 > t$ does not arise because no rule (other than Rule AC-26) can derive that.

- Case $s_1 = t$:
  Then Rule AC-9 has been applied to derive Transition F.39. Then $p_1 = x$. Rewriting Transition F.39, we get:

  $$\langle \sigma^{s_1}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle x \rangle \tag{F.40}$$

  Note $s_1 = t$.
  Putting Transition F.40 in Transition F.36, we get:

  $$\langle \sigma^{s_1}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle x \rangle \xmapsto{s_2} \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.41}$$

- Case $s_1 < t$:
  Then Rule AC-8 has been applied to derive Transition F.39. Let

  $$t = s_1 + u \tag{F.42}$$

  Rewriting Transition F.39, we get:

  $$\langle \sigma^{s_1+u}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle \sigma^{u}_{\mathsf{rel}}(x) \rangle \tag{F.43}$$

  Putting Transition F.43 in Transition F.36, we get:

  $$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{s_1} \langle \sigma^{u}_{\mathsf{rel}}(x) \rangle \xmapsto{s_2} \langle p_2 \rangle \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.44}$$

  Again there are only two rules applicable on $\sigma^{u}_{\mathsf{rel}}(x)$, Rule AC-8 and Rule AC-9 and only two cases are possible:

  $$s_2 < u \text{ or } s_2 = u$$

- If $s_2 = u$, then $p_2 = x$. Then from F.42,

$$t = s_1 + s_2 \tag{F.45}$$

Rewriting Transition F.44 by putting the value of p_2, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \rangle \xmapsto{s_1} \langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{s_2} \langle x \rangle \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.46}$$

where $t = s_1 + s_2$.

- If $s_2 < u$, then the second time step in F.44 is derived by Rule AC-8. Let $u = s_2 + v$. Rewriting second time step in Transition F.44, we get:

$$\langle \sigma_{\mathsf{rel}}^{s_2+v}(x) \rangle \xmapsto{s_2} \langle \sigma_{\mathsf{rel}}^{v}(x) \rangle \tag{F.47}$$

Rewriting Transition F.44 by putting the value of p_2, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \rangle \xmapsto{s_1} \langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{s_2} \langle \sigma_{\mathsf{rel}}^{v}(x) \rangle \ldots \langle p_{n-1} \rangle \xmapsto{s_n} \langle z_1 \rangle \tag{F.48}$$

where $t = s_1 + u = s_1 + s_2 + v$.

Consider the instantiations F.41, F.46 and F.48 of Transition F.36. We notice a pattern. Till the duration t is covered by the constituent time steps, no rules other than Rule AC-8 and Rule AC-9 are applicable. The time step in which the duration t is covered, is derived from Rule AC-9. From this observation we infer that there exists a j, such that the sum of delays of first j transitions of Transition F.36 equals t. I.e.,

$$s_1 + \ldots + s_j = t \tag{F.49}$$

From F.38, we know that the duration of Transition F.36 is greater than t. The extra duration $(s_1 + \ldots + s_n) - t$ in Transition F.36 must be due to the delay of x as operator $\sigma_{\mathsf{rel}}^{t}$ in front of x caters for a delay of first j time steps. Then j must be smaller than n, as at least one transition is required to cover the delay of x.

Now $(s_1 + \ldots + s_n) = r$ in Transition F.36. From F.49:

$$r - t = (s_1 + \ldots + s_n) - (s_1 + \ldots + s_j) = s_{j+1} + \ldots + s_n \tag{F.50}$$

We partition Transition F.36 into time transitions of durations '$s_1 + \ldots + s_j$' and '$s_{j+1} + \ldots + s_n$', we get:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(x) \rangle \xmapsto{s_1+\ldots+s_j} \langle x \rangle \tag{F.51}$$

$$\langle x \rangle \xmapsto{s_{j+1}+\ldots+s_n} \langle z_1 \rangle \tag{F.52}$$

The j^{th} transition in Transition F.37 is obtained by Rule AC-9. From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ x \rangle \tag{F.53}$$

B. **Transition F.4 by Rule AC-26**

We can apply the same reasoning for Transition F.4 as given for Transition F.3. There exists $m > 1$, such that Transition F.4 is obtained from Rule 26 by combining m successive transitions.
Let $u_1, \ldots, u_m$ denote the durations of the constituent time transitions and let $q_1, \ldots, q_{m-1}$ denote the intermediate process terms.
Splitting Transition F.4 into m transitions:

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{u_1} \langle q_1 \rangle \xmapsto{u_2} \ldots \langle q_{m-1} \rangle \xmapsto{u_m} \langle z_2 \rangle \tag{F.54}$$

and

$$u_1 + \ldots + u_m = r \tag{F.55}$$

From F.55 and the fact that we are considering the case with $r > t$, we infer that:

$$u_1 + \ldots + u_m > t$$

Now, there are only two rules applicable on $\sigma^t_{\mathsf{rel}}(y)$, Rule AC-8 and Rule AC-9. Together the application of rules Rule AC-8 and Rule AC-9 can cover a duration t for a process term $\sigma^t_{\mathsf{rel}}(y)$. The extra duration $(u_1 + \ldots + u_m) - t$ in Transition F.54 is covered by a delay of y.
By reasoning given for Transition F.36, there exists a k, with $1 \leq k < m$ such that:

$$u_1 + \ldots + u_k = t \tag{F.56}$$

Then from F.55 and the fact that $k < m$, we infer:

$$u_{k+1} + \ldots + u_m = r - t \tag{F.57}$$

We partition Transition F.54 into time transitions of durations '$u_1 + \ldots + u_k$' and '$u_{k+1} + \ldots + u_m$', we get:

$$\langle \sigma^{u_1+\ldots+u_m}_{\mathsf{rel}}(y) \rangle \xmapsto{u_1+\ldots+u_k} \langle y \rangle \tag{F.58}$$

$$\langle y \rangle \xmapsto{u_{k+1}+\ldots+u_m} \langle z_2 \rangle \tag{F.59}$$

The k^{th} transition in Transition F.58 is obtained by Rule AC-9. From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ \ y \rangle \tag{F.60}$$

From Predicates F.53 and F.60, we infer that:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

Then, Rule AC-9 can derive the following transition for the duration $s_1 + \ldots + s_j$ defined in F.49.

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{s_1+\ldots+s_j} \langle x + y \rangle \tag{F.61}$$

From F.50 and F.57, we infer that:

$$s_{j+1} + \ldots + s_n = u_{k+1} + \ldots + u_m$$

Apply Rule 19 on Transitions F.52 and F.59:

$$\langle x + y \rangle \xmapsto{s_{j+1}+\ldots+s_n} \langle z_1 + z_2 \rangle \tag{F.62}$$

Apply Rule AC-26 on Transitions F.61 and F.62:

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{s_1+\ldots+s_n} \langle z_1 + z_2 \rangle$$

From F.37,

$$r = s_1 + \ldots + s_n$$

I.e.,

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{r} \langle z_1 + z_2 \rangle \tag{F.63}$$

Consider Transitions F.2 and F.63. The pair $(z_1 + z_2, z_1 + z_2) \in \mathcal{I}$.

(b) <u>Rule AC-20</u>

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{r} \langle z \rangle \qquad\qquad (F.1)$$

If Transition F.1, given above, is derived from this rule, then the following must hold:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{r} \langle z \rangle \tag{F.64}$$

$$\langle \texttt{consistent}\ \sigma^t_{\mathsf{rel}}(y) \rangle \tag{F.65}$$

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{r} \tag{F.66}$$

$$\forall s < r, \quad \langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{s} \tag{F.67}$$

Consider Predicate F.67, for $s < t$. Let $t = s + s_1$, from some $s_1 > 0$.
Then, the following transition is always derivable from Rule AC-8:

$$\langle \sigma^{s+s_1}_{\mathsf{rel}}(y) \rangle \xmapsto{s} \langle \sigma^{s_1}_{\mathsf{rel}}(y) \rangle$$

Hence, Predicate Predicate F.67 doesn't hold.
We conclude that Rule AC-20 cannot be used to derive Transition F.1.

(c) <u>Rule AC-21</u>

Rule AC-21 is not applicable due to the same reasons as Rule AC-20.

(d) <u>Rule AC-26</u>

Suppose, Transition F.1 (repeated below) is derived from this rule:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{r} \langle z \rangle \qquad\qquad (F.1)$$

By reasoning give above, we can say that there exists an $n > 1$ such that Transition F.1 is obtained by combining n successive transitions.
Let $s_1, \ldots, s_n$ denote the durations of the constituent time transitions and let $p_1, \ldots, p_{n-1}$ denote the intermediate process terms.
Splitting Transition F.1 into n transitions:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y) \rangle \overset{s_1}{\longmapsto} \langle p_1 \rangle \overset{s_2}{\longmapsto} \ldots \langle p_{n-1} \rangle \overset{s_n}{\longmapsto} \langle z \rangle \tag{F.68}$$

and

$$s_1 + \ldots + s_n = r \tag{F.69}$$

We distinguish between three cases:

i. <u>**Case** $s_1 + \ldots + s_n < t$</u>

   Let
   $$t = s_1 + \ldots + s_n + r_1 \tag{F.70}$$
   for some r_1, with $0 < r_1 < t$.
   Rewriting Transition F.68:

   $$\langle \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(x) + \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(y) \rangle \overset{s_1}{\longmapsto} \langle p_1 \rangle \ldots \langle p_{n-1} \rangle \overset{s_n}{\longmapsto} \langle z \rangle \tag{F.71}$$

   A time step for an alternative composition can be derived from Rules AC-19, AC-20 and AC-21.
   Consider the first constituent time step of Transition F.71:

   $$\langle \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(x) + \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(y) \rangle \overset{s_1}{\longmapsto} \langle p_1 \rangle \tag{F.72}$$

   Transition F.72 can only be derived from Rule AC-19 as Rules AC-20 and AC-21 are not applicable. The application of Rule AC-20 (Rule AC-21) requires that the right (left) alternative is undelayable.
   From the premise of Rule AC-19, for some $p', p'' \in P$:

   $$p_1 = p' + p''$$

   and the following holds:

   $$\langle \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(x) \rangle \overset{s_1}{\longmapsto} \langle p' \rangle \tag{F.73}$$
   $$\langle \sigma^{s_1+\ldots+s_n+r_1}_{\mathsf{rel}}(y) \rangle \overset{s_1}{\longmapsto} \langle p'' \rangle \tag{F.74}$$

   Transitions F.73 and F.74 can only be derived from Rules AC-8.
   Then,
   $$p' = \sigma^{s_2+\ldots+s_n+r_1}_{\mathsf{rel}}(x) \text{ and } p'' = \sigma^{s_2+\ldots+s_n+r_1}_{\mathsf{rel}}(y)$$
   By similar reasoning, all n time steps of composite Time Transition F.71 have been derived from Rule AC-19 as Rules AC-20 and AC-21 are not applicable.
   The final process term z is as follows:

   $$\sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y) \tag{F.75}$$

Rewriting Transition F.1:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \xmapsto{r} \langle \sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y)\rangle \tag{F.76}$$

From F.69 and F.70:

$$t = r + r_1$$

By Rule AC-8, the following is derivable:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x+y)\rangle \xmapsto{r} \langle \sigma^{r_1}_{\mathsf{rel}}(x+y)\rangle \tag{F.77}$$

Consider the target terms in Transitions F.76 and F.77. For $0 < r_1 < t$, the pair $(\sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y), \sigma^{r_1}_{\mathsf{rel}}(x+y))$ is in R.

ii. **Case** $s_1 + \ldots + s_n = t$

Replacing the value t in Transition F.68 by the sum $s_1 + \ldots + s_n$:

$$\langle \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(x) + \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(y)\rangle \xmapsto{s_1} \langle p_1\rangle \xmapsto{s_2} \ldots \langle p_{n-1}\rangle \xmapsto{s_n} \langle z\rangle \tag{F.78}$$

A time step for an alternative composition can be derived from Rules AC-19, AC-20 and AC-21.
Consider the first constituent time step of Transition F.78:

$$\langle \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(x) + \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(y)\rangle \xmapsto{s_1} \langle p_1\rangle \tag{F.79}$$

Transition F.79 can only be derived from Rule AC-19 as Rules AC-20 and AC-21 are not applicable. The application of Rule AC-20 (Rule AC-21) requires that the right (left) alternative is undelayable.
From the premise of Rule AC-19, for some $p', p'' \in P$:

$$p_1 = p' + p''$$

and the following holds:

$$\langle \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(x)\rangle \xmapsto{s_1} \langle p'\rangle \tag{F.80}$$

$$\langle \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(y)\rangle \xmapsto{s_1} \langle p''\rangle \tag{F.81}$$

Note that $n > 1$ and for all i, $s_i > 0$. Therefore, Transitions F.80 and F.81 can only be derived from Rules AC-8.
Then,

$$p' = \sigma^{s_2+\ldots+s_n}_{\mathsf{rel}}(x) \text{ and } p'' = \sigma^{s_2+\ldots+s_n}_{\mathsf{rel}}(y)$$

Rewriting Transition F.79 by putting the value of p_1:

$$\langle \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(x) + \sigma^{s_1+\ldots+s_n}_{\mathsf{rel}}(y)\rangle \xmapsto{s_1} \langle \sigma^{s_2+\ldots+s_n}_{\mathsf{rel}}(x) + \sigma^{s_2+\ldots+s_n}_{\mathsf{rel}}(y)\rangle \tag{F.82}$$

Now consider the second constituent time step of Transition F.78:

$$\langle \sigma_{\mathsf{rel}}^{s_2+\ldots+s_n}(x) + \sigma_{\mathsf{rel}}^{s_2+\ldots+s_n}(y)\rangle \xmapsto{s_2} \langle p_2\rangle \tag{F.83}$$

Transition F.83 is again only be derived from Rule AC-19 as Rules AC-20 and AC-21 are not applicable on the source of the above transition.
Till the duration t on process term $\sigma_{\mathsf{rel}}^{t}(x) + \sigma_{\mathsf{rel}}^{t}(y)$ is covered, all the constituent transitions of Transition F.78 are obtained by Rule AC-19. Other rules for deriving delay of an alternative composition, Rules AC-20 and AC-21, require that the passive operand must be undelayable, which is not satisfied till *atleast* the relative delay operator disappears from a process term $\sigma_{\mathsf{rel}}^{t}(x)$. We are considering the case where $t = s_1 + \ldots s_n$, which is the total duration of Transition F.78. Therefore, the duration t is only covered in the last time transition.
The last time step is as follows:

$$\langle \sigma_{\mathsf{rel}}^{s_n}(x) + \sigma_{\mathsf{rel}}^{s_n}(y)\rangle \xmapsto{s_n} \langle z\rangle \tag{F.84}$$

From Rule AC-19, for some $z_1, z_2 \in P$, $z = z_1 + z_2$.
From premise of Rule AC-19 the following holds:

$$\langle \sigma_{\mathsf{rel}}^{s_n}(x)\rangle \xmapsto{s_n} \langle z_1\rangle \tag{F.85}$$

$$\langle \sigma_{\mathsf{rel}}^{s_n}(y)\rangle \xmapsto{s_n} \langle z_2\rangle \tag{F.86}$$

Rewriting Transition F.1 by replacing z by $z_1 + z_2$:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) + \sigma_{\mathsf{rel}}^{t}(y)\rangle \xmapsto{t} \langle z\rangle \tag{F.87}$$

Transitions F.85 and F.86 can only be derived from Rule AC-9.
Then, $z_1 = x$ and $z_2 = y$.
Rule AC-9 requires:

$$\langle \texttt{consistent}\ \ x\rangle \text{ and } \langle \texttt{consistent}\ \ y\rangle$$

which implies:

$$\langle \texttt{consistent}\ \ x + y\rangle \tag{F.88}$$

Rewriting Transition F.87 by putting in the values of z_1 and z_2:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) + \sigma_{\mathsf{rel}}^{t}(y)\rangle \xmapsto{t} \langle x + y\rangle \tag{F.89}$$

From Predicate F.88, Rule AC-9 becomes applicable to derive the following time step.

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_n}(x + y)\rangle \xmapsto{s_1+\ldots+s_n} \langle x + y\rangle \tag{F.90}$$

The pair $(x + y, x + y)$ is in $\mathcal{I}$.

iii. **Case $s_1 + \ldots + s_n > t$**

Repeating Transition F.1 with r replaced by the sum of durations $s_1 + \ldots + s_n$ from F.69.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{s_1+\ldots+s_n} \langle z \rangle \tag{F.91}$$

As explained in the last case for $s_1 + \ldots + s_n = t$, the first k time steps of the above composite Transition, such that $s_1 + \ldots + s_k \leq t$, can only be derived from Rule AC-19.
We distinguish between two cases: In one case, there exists a j, with $1 \leq j < n$ such that $s_1 + \ldots + s_j = t$. In the second case, for all i with $1 \leq i \leq n$, $s_1 + \ldots + s_i$ is either strictly less than t or $s_1 + \ldots + s_i$ is strictly greater than t.
The second case arises due to the following reason:
All transitions in F.91 have been derived from rules other than Rule AC-26. The rules allowing a delay of alternative composition are Rules AC-19, AC-20 and AC-21. The premise of these rules contain time transitions which may have been derived by Rule AC-26. The following is an example exhibiting the second case:
Consider the process term $\sigma^1_{\mathsf{rel}}(\sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}})) + \sigma^2_{\mathsf{rel}}(b)$. Both process terms can delay for 2 time units. Hence Rule AC-19 can be applied to derive the following transition:

$$\langle \sigma^1_{\mathsf{rel}}(\sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}})) + \sigma^2_{\mathsf{rel}}(\tilde{\tilde{b}}) \rangle \xmapsto{2} \langle \tilde{\tilde{a}} + \tilde{\tilde{b}} \rangle$$

But one of the prerequisites of the rule, (Transition F.92) is derived from Rule AC-26.

$$\langle \sigma^1_{\mathsf{rel}}(\sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}})) \rangle \xmapsto{1+1} \langle \tilde{\tilde{a}} \rangle \tag{F.92}$$

$$\langle \sigma^2_{\mathsf{rel}}(\tilde{\tilde{b}}) \rangle \xmapsto{2} \langle \tilde{\tilde{b}} \rangle \tag{F.93}$$

In the derivation of time transitions of duration greater than t with a source process term of the form $\sigma^t_{\mathsf{rel}}(z)$ (For example Transition F.36), there always exists a j such that $s_1 + \ldots s_j = t$. Because the rules for the process term $\sigma^t_{\mathsf{rel}}(z)$ do not contain any delay transitions in their premises.

A. **Case 1**

Suppose, there exists a j with $1 \leq j < n$ such that

$$s_1 + \ldots + s_j = t \tag{F.94}$$

Rewriting Transition F.91:

$$\langle \sigma^{s_1+\ldots+s_j}_{\mathsf{rel}}(x) + \sigma^{s_1+\ldots+s_j}_{\mathsf{rel}}(y) \rangle \xmapsto{s_1+\ldots+s_n} \langle z \rangle \tag{F.95}$$

Then the j^{th} time step is as follows:

$$\langle \sigma^{s_j}_{\mathsf{rel}}(x) + \sigma^{s_j}_{\mathsf{rel}}(y) \rangle \xmapsto{s_j} \langle p \rangle \tag{F.96}$$

for some process term p.
Only Rule AC-19 can derive Transition F.96. From the premise of the rule, $p = p' + p''$ and the following holds:

$$\langle \sigma_{\mathsf{rel}}^{s_j}(x) \rangle \xmapsto{s_j} \langle p' \rangle \tag{F.97}$$

$$\langle \sigma_{\mathsf{rel}}^{s_j}(y) \rangle \xmapsto{s_j} \langle p'' \rangle \tag{F.98}$$

The above transitions can only be derived from Rule AC-9. Then $p = x + y$ and

$$\langle \texttt{consistent} \;\; x \rangle \text{ and } \langle \texttt{consistent} \;\; y \rangle$$

which implies:

$$\langle \texttt{consistent} \;\; x + y \rangle \tag{F.99}$$

Rewriting Transition F.96:

$$\langle \sigma_{\mathsf{rel}}^{s_j}(x) + \sigma_{\mathsf{rel}}^{s_j}(y) \rangle \xmapsto{s_j} \langle x + y \rangle \tag{F.100}$$

Partitioning Transition F.95 into two transitions of durations $s_1 + \ldots + s_j$ and $s_{j+1} + \ldots s_n$ respectively:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(x) + \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(y) \rangle \xmapsto{s_1+\ldots+s_j} \langle x + y \rangle \tag{F.101}$$

$$\langle x + y \rangle \xmapsto{s_{j+1}+\ldots+s_n} \langle z \rangle \tag{F.102}$$

From Predicate F.99, Rule AC-9 can be applied to derive the following:

$$\langle \sigma_{\mathsf{rel}}^{t}(x + y) \rangle \xmapsto{t} \langle x + y \rangle \tag{F.103}$$

Apply Rule AC-26 on time steps F.102 and F.103. We get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x + y) \rangle \xmapsto{t+s_{j+1}+\ldots+s_n} \langle z \rangle \tag{F.104}$$

From F.94, $t = s_1 + \ldots + s_j$.
Consider target terms in Transitions F.91 and F.104. The pair (z, z) is in $\mathcal{I}$.

B. **Case 2**

In the second case, for all i with $1 \leq i \leq n$, $s_1 + \ldots + s_i$ is either strictly less than t or $s_1 + \ldots + s_i$ is strictly greater than t.
Let $1 \leq j \leq (n - 1)$, such that:

$$s_1 + \ldots + s_j < t \tag{F.105}$$

$$s_1 + \ldots + s_{j+1} > t \tag{F.106}$$

Let

$$t = s_1 + \ldots + s_j + r_1 \tag{F.107}$$

Rewriting Transition F.91 by writing t as a sum of durations:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j+r_1}(x) + \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j+r_1}(y)\rangle \xmapsto{s_1+\ldots+s_n} \langle z\rangle \tag{F.108}$$

Partitioning the above Time transition into two transitions. One of duration $s_1 + \ldots + s_j$ and the other of duration $s_{j+1} + \ldots s_n$.
Let for some process term p:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j+r_1}(x) + \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j+r_1}(y)\rangle \xmapsto{s_1+\ldots+s_j} \langle p\rangle \tag{F.109}$$

$$\langle p\rangle \xmapsto{s_{j+1}+\ldots+s_n} \langle z\rangle \tag{F.110}$$

It is easy to prove that $p = \sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y)$.
The process term p is the source of the Transition $j+1$ of Transition F.110. Let the target of $j+1$ time step be q. Partitioning Transition F.110 into two transitions of durations s_{j+1} and $s_{j+2} \ldots + s_n$.

$$\langle \sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y)\rangle \xmapsto{s_{j+1}} \langle q\rangle \tag{F.111}$$

$$\langle q\rangle \xmapsto{s_{j+2}+\ldots+s_n} \langle z\rangle \tag{F.112}$$

Again on process term $\sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y)$, only Rule AC-19 is applicable. Then, for some $q_1, q_2 \in P$, q in Transition F.111 is:

$$q = q_1 + q_2 \tag{F.113}$$

From the premise of the rule, the following holds:

$$\langle \sigma_{\mathsf{rel}}^{r_1}(x)\rangle \xmapsto{s_{j+1}} \langle q_1\rangle \tag{F.114}$$

$$\langle \sigma_{\mathsf{rel}}^{r_1}(y)\rangle \xmapsto{s_{j+1}} \langle q_2\rangle \tag{F.115}$$

Rewriting Transitions F.109, F.111 and F.112:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j+r_1}(x) + \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j+r_1}(y)\rangle \xmapsto{s_1+\ldots+s_j} \langle \sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y)\rangle \tag{F.116}$$

$$\langle \sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y)\rangle \xmapsto{s_{j+1}} \langle q_1 + q_2\rangle \tag{F.117}$$

$$\langle q_1 + q_2\rangle \xmapsto{s_{j+2}+\ldots+s_n} \langle z\rangle \tag{F.118}$$

From F.105, F.106 and F.107, we know that $s_{j+1} > r_1$. Then Transitions F.114 and F.115 can only be derived from Rule AC-26.

Let for some $m > 1$:

$$s_{j+1} = u_1 + \ldots + u_m \tag{F.119}$$

Rewriting Transitions F.114 and F.115:

$$\langle \sigma_{\mathsf{rel}}^{r_1}(x) \rangle \xmapsto{u_1+\ldots+u_m} \langle q_1 \rangle \tag{F.120}$$

$$\langle \sigma_{\mathsf{rel}}^{r_1}(y) \rangle \xmapsto{u_1+\ldots+u_m} \langle q_2 \rangle \tag{F.121}$$

By applying the same reasoning as applied for Transition F.36, there exists a k, with $1 \leq k < m$, such that:

$$u_1 + \ldots + u_k = r_1 \tag{F.122}$$

The following constituents of Transitions F.120 and F.121 have been derived by applying Rule AC-9.

$$\langle \sigma_{\mathsf{rel}}^{u_k}(x) \rangle \xmapsto{u_k} \langle x \rangle \tag{F.123}$$

$$\langle \sigma_{\mathsf{rel}}^{u_k}(y) \rangle \xmapsto{u_k} \langle y \rangle \tag{F.124}$$

We can infer from above transitions that the following holds:

$$\langle \texttt{consistent}\ x + y \rangle \tag{F.125}$$

Transition F.120 is obtained by combining m time steps in a sequence. All intermediate time transitions are derivable, other wise Rule AC-26 could not be applied.
We partition Transition F.120 into time transitions of durations '$u_1 + \ldots + u_k$' and '$u_{k+1} + \ldots + u_m$', we get:

$$\langle \sigma_{\mathsf{rel}}^{u_1+\ldots+u_k}(x) \rangle \xmapsto{u_1+\ldots+u_k} \langle x \rangle \tag{F.126}$$

$$\langle x \rangle \xmapsto{u_{k+1}+\ldots+u_m} \langle q_1 \rangle \tag{F.127}$$

Similarly, partitioning Transition F.121 into time transitions of durations '$u_1 + \ldots + u_k$' and '$u_{k+1} + \ldots + u_m$', we get:

$$\langle \sigma_{\mathsf{rel}}^{u_1+\ldots+u_k}(y) \rangle \xmapsto{u_1+\ldots+u_k} \langle y \rangle \tag{F.128}$$

$$\langle y \rangle \xmapsto{u_{k+1}+\ldots+u_m} \langle q_2 \rangle \tag{F.129}$$

Apply Rule AC-19 on Transitions F.127 and F.129:

$$\langle x + y \rangle \xmapsto{u_{k+1}+\ldots+u_m} \langle q_1 + q_2 \rangle \tag{F.130}$$

Using Predicate F.125, Rule AC-9 can derive the following:

$$\langle \sigma_{\mathsf{rel}}^{r_1}(x + y) \rangle \xmapsto{r_1} \langle x + y \rangle \tag{F.131}$$

Apply Rule AC-26 on Transitions F.131 and F.130:

$$\langle \sigma_{\mathsf{rel}}^{r_1}(x + y) \rangle \xmapsto{r_1+u_{k+1}+\ldots+u_m} \langle q_1 + q_2 \rangle \tag{F.132}$$

By F.119, F.122 and the fact that $k < m$, $s_{j+1} = r_1 + u_{k+1} + \ldots + u_m$. Rewriting Transition F.132:

$$\langle \sigma^{r_1}_{\mathsf{rel}}(x + y)\rangle \xmapsto{s_{j+1}} \langle q_1 + q_2\rangle \tag{F.133}$$

Apply Rule AC-26 on Transitions F.118 and F.133.

$$\langle \sigma^{r_1}_{\mathsf{rel}}(x + y)\rangle \xmapsto{s_{j+1}+\ldots+s_n} \langle z\rangle \tag{F.134}$$

For the sum of durations $s_1 + \ldots + s_j$, (from F.105) Rule AC-8 can derive the following:

$$\langle \sigma^{s_1+\ldots+s_j+r_1}_{\mathsf{rel}}(x + y)\rangle \xmapsto{s_1+\ldots+s_j} \langle \sigma^{r_1}_{\mathsf{rel}}(x + y)\rangle \tag{F.135}$$

Again, Apply Rule AC-26 on Transitions F.134 and F.135:

$$\langle \sigma^{s_1+\ldots+s_j+r_1}_{\mathsf{rel}}(x + y)\rangle \xmapsto{s_1+\ldots++s_n} \langle z\rangle \tag{F.136}$$

Consider the target process terms in Transitions F.135 and F.135. The pair (z, z) is in R.

6.

$$\begin{aligned}\langle \sigma^{t}_{\mathsf{rel}}(x + y)\rangle \xmapsto{r} \langle z\rangle \implies \quad & \exists z' \in P : \langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \xmapsto{r} \langle z'\rangle \\ & (z', z) \in R\end{aligned}$$

Suppose,

$$\langle \sigma^{t}_{\mathsf{rel}}(x + y)\rangle \xmapsto{r} \langle z\rangle \tag{F.137}$$

We distinguish between three cases for different values of r.

(a) Case $r < t$

Let $t = r + r_1$, for some r_1 with $0 < r_1 < t$.
Then Transition F.137 can only be derived from Rules AC-8 or AC-26. It is easy to argue that in case of both rules, the process z in Transition F.137 is of the form $\sigma^{r_1}_{\mathsf{rel}}(x + y)$.
From Rule AC-8 the following can be derived:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x)\rangle \xmapsto{r} \langle \sigma^{r_1}_{\mathsf{rel}}(x)\rangle \tag{F.138}$$

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \xmapsto{r} \langle \sigma^{r_1}_{\mathsf{rel}}(y)\rangle \tag{F.139}$$

Apply Rule AC-19 on the above transitions:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) + \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \xmapsto{r} \langle \sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y)\rangle \tag{F.140}$$

Consider the target process terms in Transitions F.137 and F.140. For $0 < r_1 < t$, the pair $(\sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y), \sigma^{r_1}_{\mathsf{rel}}(x + y))$ is in R.

(b) $\underline{\underline{\text{Case } r = t}}$

Then Transition F.137 can only be derived from Rules AC-9 or AC-26. It is easy to argue that in case of Rule AC-26 (transitive closure), the last rule applied is AC-9 and the process z in Transition F.137 is of the form $x + y$.
From premise of Rule AC-9, the following holds:

$$\langle \texttt{consistent } x + y \rangle$$

which can only hold, if:

$$\langle \texttt{consistent } x \rangle \text{ and } \langle \texttt{consistent } y \rangle$$

Then Rule AC-9 can also be applied to derive the following transitions:

$$\langle \sigma^{r}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle x \rangle$$
$$\langle \sigma^{r}_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle y \rangle$$

Apply Rule AC-19 on the above transitions:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) + \sigma^{r+r_1}_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle x + y \rangle \tag{F.141}$$

Consider the target process terms in Transitions F.137 and F.141. The pair $(x + y, x + y)$ is in R.

(c) $\underline{\underline{\text{Case } r > t}}$

Transition F.137 for $r > t$ can only be derived from Rule AC-26.
Let for some $n > 0$,

$$r = s_1 + \ldots s_n$$

Rewriting Transition F.137:

$$\langle \sigma^{t}_{\mathsf{rel}}(x + y) \rangle \xrightarrow{s_1 + \ldots + s_n} \langle z \rangle \tag{F.142}$$

Transition F.142 has been obtained by applying Rule AC-26 on n time transitions. The constituent transitions have been each obtained from application of a single rule other than Rule AC- 26. The duration $s_1 + \ldots + s_n$ of Transitions F.142 is greater than t. Hence, there exists a j, with $1 \leq j < n$ such that:

$$s_1 + \ldots + s_j = t \tag{F.143}$$

Rewriting Transition F.142, (replacing t by the sum $s_1 + \ldots + s_j$):

$$\langle \sigma^{s_1 + \ldots + s_j}_{\mathsf{rel}}(x + y) \rangle \xrightarrow{s_1 + \ldots + s_n} \langle z \rangle \tag{F.144}$$

The jth constituent of Transition F.144 has been derived by applying Rule AC-9.

$$\langle \sigma^{s_j}_{\mathsf{rel}}(x) + \sigma^{s_j}_{\mathsf{rel}}(y) \rangle \overset{s_j}{\longmapsto} \langle x + y \rangle \tag{F.145}$$

Rule AC-9 can only be used to derive the above transitions, if $\langle \texttt{consistent}\ \ x+y\rangle$, which implies:

$$\langle \texttt{consistent}\ \ x\rangle \tag{F.146}$$
$$\langle \texttt{consistent}\ \ y\rangle \tag{F.147}$$

We partition Transition F.144 into time transitions of durations '$s_1 + \ldots + s_j$' and '$s_{j+1} + \ldots + s_n$', we get:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(x+y)\rangle \xmapsto{s_1+\ldots+s_j} \langle x+y\rangle \tag{F.148}$$
$$\langle x+y\rangle \xmapsto{s_{j+1}+\ldots+s_n} \langle z\rangle \tag{F.149}$$

From Predicates F.146 and F.147, Rule AC-9 can derive the following:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(x)\rangle \xmapsto{s_1+\ldots+s_j} \langle x\rangle \tag{F.150}$$
$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(y)\rangle \xmapsto{s_1+\ldots+s_j} \langle y\rangle \tag{F.151}$$

Apply Rule AC-19 on the above transitions:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(x) + \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(y)\rangle \xmapsto{s_1+\ldots+s_j} \langle x+y\rangle \tag{F.152}$$

Apply Rule AC-26 on Transitions F.152 and F.149:

$$\langle \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(x) + \sigma_{\mathsf{rel}}^{s_1+\ldots+s_j}(y)\rangle \xmapsto{s_1+\ldots+s_n} \langle z\rangle \tag{F.153}$$

Consider the target process terms in Transitions F.153 and F.137. The pair (z, z) is in R.

⊠

Appendix G

Theorem 8

Thoerem 8

Axiom SRT3 is sound in the semantics of Section 3.6.3.

$$\sigma^u_{\mathsf{rel}}(x) + \sigma^u_{\mathsf{rel}}(y) = \sigma^u_{\mathsf{rel}}(x + y) \qquad (SRT3)$$

Proof

We prove the soundness of Axiom SRT3 in two steps.

Case $u = 0$

From Rules RI-4, RI-5, RI-6 and RI-7, it easy to prove the following holds in the semantics of $BPA^{srt}_{\perp}$ with modified Relative Delay Operator (Section 3.6.3):

For any process term x,

$$\sigma^0_{\mathsf{rel}}(x) \leftrightarroweq x$$

Since Bisimulation is a congruence therefore, then it becomes trivial to prove that:

$$\sigma^0_{\mathsf{rel}}(x) + \sigma^0_{\mathsf{rel}}(y) \leftrightarroweq \sigma^0_{\mathsf{rel}}(x + y)$$

Case $u > 0$

Let $\mathcal{I}$ be the following relation:

$$\mathcal{I} = \{(p, p) \mid p \in P\}$$

Let R be the following relation:

$$R = \{(\sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(x)), \sigma^t_{\mathsf{rel}}(x + y) \mid 0 < t \leq u, x, y \in P\}$$

We prove that $R \cup \mathcal{I}$ is a bisimulation relation:

For all $a \in A, r > 0, z \in P$:

1.

$$\begin{aligned} \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \langle z \rangle \implies \quad & \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xrightarrow{a} \langle z' \rangle \\ & (z, z') \in R \end{aligned}$$

Trivial.

2.

$$\begin{aligned}\langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \xrightarrow{a} \langle z\rangle \implies \quad & \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xrightarrow{a} \langle z'\rangle \\ & (z', z) \in R\end{aligned}$$

Trivial.

3.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xrightarrow{a} \surd \iff \langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \xrightarrow{a} \surd$$

Trivial.

4.

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \iff \langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(x+y)\rangle$$

Trivial.

5.

$$\begin{aligned}\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \implies \quad & \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle z'\rangle \\ & (z, z') \in R\end{aligned}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \tag{G.1}$$

This can be derived from Rules RI-20, RI-21, RI-22.

(a) Rule RI-20

Then $z = z_1 + z_2$. Rewriting Transtion G.1:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z_1 + z_2\rangle \tag{G.2}$$

And the following must hold:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z_1\rangle \tag{G.3}$$

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z_2\rangle \tag{G.4}$$

We distinguish between three cases:

i. **Case** $r < t$
Let $t = r + r_1$, for some $0 < r_1 < t$.
Then Transtions G.3 and G.4 can only be derived from Rule 8.

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x)\rangle$$
$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(y)\rangle$$

Rule 8 can derive the following:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x+y)\rangle \tag{G.5}$$

For $0 < r_1 < t$, the pair $(\sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y), \sigma^{r_1}_{\mathsf{rel}}(x+y)) \in R$.

ii. **Case** $r = t$
Proof is similar to above. Rule 9 is used which has no conditions like Rule 8.

iii. **Case** $r > t$
Let $r = t + t_1$, for $t_1 > 0$. Then Transtions G.3 and G.4 can only be derived from Rule 10.
From the premise of the rule, the following holds:

$$\langle x \rangle \xmapsto{t_1} \langle z_1 \rangle$$

$$\langle y \rangle \xmapsto{t_1} \langle z_2 \rangle$$

Apply Rule 20 on above transitions:

$$\langle x + y \rangle \xmapsto{t_1} \langle z_1 + z_2 \rangle$$

Apply Rule 10 on above transition:

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{t+t_1} \langle z_1 + z_2 \rangle$$

The pair $(z_1 + z_2, z_1 + z_2) \in \mathcal{I}$.

(b) Rule RI-21

If Transition G.1 given below is derived from this rule:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{r} \langle z \rangle$$

Then the following must holds:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{r} \langle z \rangle \tag{G.6}$$

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(y) \rangle \tag{G.7}$$

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\not\mapsto} \tag{G.8}$$

$$\forall y', \forall s < r(\langle \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{s} \langle y' \rangle \implies \langle \texttt{consistent } y' \rangle) \tag{G.9}$$

We distinguish between three cases:

i. **Case** $r < t$
Not Applicable. Transition G.8 cannot be derived.

ii. **Case** $r = t$
Not Applicable. Transition G.8 cannot be derived.

iii. **Case** $r > t$
Let $r = t + t_1$, for some $t_1 > 0$.
Then Transition G.6 can only be derived from Rule RI-10. From the premise,

$$\langle x \rangle \xmapsto{t_1} \langle z \rangle \tag{G.10}$$

Transition G.8 implies that Rule RI-10 is not applicable. Therefore the premise must not hold:

$$\langle y \rangle \not\xrightarrow{t_1} \tag{G.11}$$

In G.9, for $s = t$, a time transition for $\sigma^t_{\textsf{rel}}(y)$ can only be derived from Rule 9. Then $y' = y$ and y is consistent.

$$\langle \texttt{consistent}\ \ y \rangle \tag{G.12}$$

In G.9, for $s > t$, a time transition for $\sigma^t_{\textsf{rel}}(y)$ can only be derived from Rule 10.
Let $s = v + t$, for $0 < v < t_1$.
Then, the following must hold:

$$\forall y', \forall v < t_1 (\langle y \rangle \xmapsto{v} \langle y' \rangle \implies \langle \texttt{consistent}\ \ y' \rangle) \tag{G.13}$$

Combine Transitions G.10,G.11,G.12 and G.13 and apply Rule RI-21:

$$\langle x + y \rangle \xmapsto{t_1} \langle z \rangle \tag{G.14}$$

Now apply Rule RI-10 on the above trnaistion:

$$\langle \sigma^t_{\textsf{rel}}(x + y) \rangle \xmapsto{t+t_1} \langle z \rangle \tag{G.15}$$

Consider Transitions G.1 and G.15. The pair $(z, z) \in R$.

(c) Rule RI-22

Same as Rule RI-21

6.
$$\langle \sigma^t_{\textsf{rel}}(x + y) \rangle \xmapsto{r} \langle z \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^t_{\textsf{rel}}(x) + \sigma^t_{\textsf{rel}}(y) \rangle \xmapsto{r} \langle z' \rangle \\ (z', z) \in R \end{array}$$

Suppose,

$$\langle \sigma^t_{\textsf{rel}}(x + y) \rangle \xmapsto{r} \langle z \rangle \tag{G.16}$$

We distinguish between three cases:

(a) **Case** $r < t$
Let $t = r + r_1$, for $0 < r_1 < t$.
Rule RI-8 can derive the following transitions:

$$\langle \sigma^{r+r_1}_{\textsf{rel}}(x) \rangle \xmapsto{r} \langle \sigma^{r_1}_{\textsf{rel}}(x) \rangle$$
$$\langle \sigma^{r+r_1}_{\textsf{rel}}(y) \rangle \xmapsto{r} \langle \sigma^{r_1}_{\textsf{rel}}(y) \rangle$$

Apply Rule RI-20 on above transitions:

$$\langle \sigma^{r+r_1}_{\textsf{rel}}(x) + \sigma^{r+r_1}_{\textsf{rel}}(y) \rangle \xmapsto{r} \langle \sigma^{r_1}_{\textsf{rel}}(x) + \sigma^{r_1}_{\textsf{rel}}(y) \rangle \tag{G.17}$$

(b) **Case** $r = t$

Similar to above. Rule RI-9 is used.

(c) **Case** $r > t$

Let $r = t + t_1$, for $t_1 > 0$.

Transition G.16 can only be derived from Rule RI-10. Then the following must hold:

$$\langle x + y\rangle \xmapsto{t_1} \langle z\rangle \tag{G.18}$$

The above transition can be derived from three rules:

i. <u>Rule RI-20</u>

If Transition G.18 is derived from this rule, then $z = z_1 + z_2$.
From the premise of the rule, the following holds:

$$\langle x\rangle \xmapsto{t_1} \langle z_1\rangle$$
$$\langle y\rangle \xmapsto{t_1} \langle z_2\rangle$$

Apply Rule RI-10 on the above transitions:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+t_1} \langle z_1\rangle$$
$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \xmapsto{t+t_1} \langle z_2\rangle$$

Apply Rule RI-20:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xmapsto{t+t_1} \langle z_1 + z_2\rangle$$

ii. <u>Rule RI-21</u>

If Transition G.18 is derived from this rule, then:

$$\langle x\rangle \xmapsto{t_1} \langle z\rangle \tag{G.19}$$
$$\langle \texttt{consistent}\ \ y\rangle \tag{G.20}$$
$$\langle y\rangle \not\xmapsto{t_1} \tag{G.21}$$
$$\forall y', \forall s < t_1(\langle y\rangle \overset{s}{\mapsto} \langle y'\rangle \implies \langle \texttt{consistent}\ \ y'\rangle) \tag{G.22}$$

Apply Rule RI-10 on Transition G.19:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+t_1} \langle z\rangle \tag{G.23}$$

From Rule RI-11,

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(y)\rangle \tag{G.24}$$

From Predicate G.21, Rule RI-10 is not applicable. Then,

$$\langle \sigma^{t}_{\mathsf{rel}}(y) \rangle \not\xrightarrow{t+t_1} \tag{G.25}$$

We know that y is consistent from Predicate G.20.
From Predicate G.20, Rules RI-8, RI-11 and RI-9:

$$\forall y', \forall s \leq t(\langle \sigma^{t}_{\mathsf{rel}}(y) \rangle \xmapsto{s} \langle y' \rangle \implies \langle \texttt{consistent}\ y' \rangle) \tag{G.26}$$

By Rule RI-10:

$$\langle y \rangle \xmapsto{s} \langle y' \rangle \implies \langle \sigma^{t}_{\mathsf{rel}}(y) \rangle \xmapsto{t+s} \langle y' \rangle$$

Using G.22,

$$\forall y', \forall s < t_1 \langle \sigma^{t}_{\mathsf{rel}}(y) \rangle \xmapsto{t+s} \langle y' \rangle \implies \langle \texttt{consistent}\ y' \rangle \tag{G.27}$$

Join G.26 and G.27:

$$\forall y', \forall s < t + t_1 \qquad \langle \sigma^{t}_{\mathsf{rel}}(y) \rangle \xmapsto{s} \langle y' \rangle \implies \langle \texttt{consistent}\ y' \rangle \tag{G.28}$$

Join Transitions G.23, G.24, G.25 and G.28 and apply Rule RI-21:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y) \rangle \xmapsto{t+t_1} \langle z \rangle \tag{G.29}$$

iii. Rule RI-22

Same as the the Rule RI-21.

⊠

Appendix H

Soundness Proofs for Proposal 1

Let $\mathcal{I}$ be a binary relation on process terms defined as follows:

$$\mathcal{I} = \{(x, x) \mid x \in P\}$$

It is obvious that $\mathcal{I}$ is a bisimulation relation. We will use the relation $\mathcal{I}$ frequently in the proofs. We prove that the axioms given in Table 3.10 hold in the semantics given in Section 3.5. Axiom A1 is proved by showing that the transition system satisfies the commutativity format (for the choice operator) given in [Mou05]. The rest of the axioms are proven in the traditional way by proving that left and right hand sides of all axioms are bisimilar. We give complete proofs for some of the axioms including Axioms A2 and axioms for the relative delay operator. For the rest of the proofs, we only give a bisimulation relation. A reader interested in the complete proofs can refer to [KC08] for complete proofs.

The proofs of the soundness theorem use the following two theorems.

H.1 Theorem : Sources of Transitions are Consistent

Theorem 12 *For all closed terms p the following holds:*
For all $p', p'' \in P$, $a, b \in A$, $r, s > 0$:

$$(\langle p\rangle \xrightarrow{a} \langle p'\rangle) \vee (\langle p\rangle \overset{r}{\longmapsto} \langle p''\rangle) \vee (\langle p\rangle \xrightarrow{b} \surd)$$
$$\implies \langle \mathtt{consistent}\ p\rangle$$

<u>Proof</u> We prove the above theorem by structural induction on a process term $p \in P$. The base case of the structural induction comprises of constant process terms, i.e. all undelayable actions in $\mathcal{A}$, the deadlock process term δ and the inconsistent process $\perp$.
<u>Base Case</u>

1. $p = \tilde{\tilde{a}}$.

   From Rule P1-2, $\langle \mathtt{consistent}\ \tilde{\tilde{a}}\rangle$. Hence all conditions of the theorem are trivially satisfied.

2. $p = \tilde{\tilde{\tilde{\delta}}}$

   From Rule P1-1, $\langle \mathtt{consistent}\ \tilde{\tilde{\tilde{\tilde{\delta}}}}\rangle$. Hence all conditions of the theorem are trivially satisfied.

3. $p = \bot$

   There are no rules for an inconsistent process $\bot$ in the semantics of $BPA_{\bot}^{srt}$. Hence all conditions of the theorem are trivially satisfied (as the left hand sides of the implications do not hold.)

By Induction Hypothesis

1. $p = \sigma_{\mathsf{rel}}^{0}(x)$, for a closed term x. We show that if p can perform an action or a time step or a termination predicate holds for p, then $\langle \texttt{consistent}\ p\rangle$ holds.

   (a) *Action Step:*
   Suppose,

   $$\langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \xrightarrow{a} \langle p'\rangle$$

   It can only be derived from Rule P1-6. From the premise of the rule,

   $$\langle x\rangle \xrightarrow{a} \langle p'\rangle$$

   By Induction on the above action step, we get:

   $$\langle \texttt{consistent}\ x\rangle$$

   Apply Rule P1-4. We get:

   $$\langle \texttt{consistent}\ \sigma_{\mathsf{rel}}^{0}(x)\rangle$$

   Hence proved.

   (b) *Time Step:*
   Suppose,

   $$\langle \sigma_{\mathsf{rel}}^{0}(x)\rangle \overset{r}{\longmapsto} \langle p'\rangle$$

   It can only be derived from Rule P1-7. From the premise of the rule,

   $$\langle x\rangle \overset{r}{\longmapsto} \langle p'\rangle$$

   By Induction on the above action step, we get:

   $$\langle \texttt{consistent}\ x\rangle$$

   Apply Rule P1-4. We get:

   $$\langle \texttt{consistent}\ \sigma_{\mathsf{rel}}^{0}(x)\rangle$$

   Hence proved.

(c) *Termination Predicate:*
Suppose,

$$\langle \sigma^0_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \surd$$

It can only be derived from Rule P1-5. From the premise of the rule,

$$\langle x\rangle \xrightarrow{a} \surd$$

By Induction on the above action step, we get:

$$\langle \texttt{consistent}\ x\rangle$$

Apply Rule P1-4. We get:

$$\langle \texttt{consistent}\ \sigma^0_{\mathsf{rel}}(x)\rangle$$

Hence proved.

2. $p = \sigma^t_{\mathsf{rel}}(x) \qquad t > 0$

From Rule P1-8, for a process term $\sigma^t_{\mathsf{rel}}(x)$, with $t > 0$, the following holds:

$$\langle \texttt{consistent}\ \sigma^t_{\mathsf{rel}}(x)\rangle$$

Hence all conditions of the theorem are trivially proved.

3. $p = x \cdot y$.

We prove the four conditions of the theorem one by one.

(a) *Action Step:*
Suppose,

$$\langle x \cdot y\rangle \xrightarrow{a} \langle p'\rangle \tag{H.1}$$

It can only be derived from Rule P1-13 or Rule P1-14.

- *Rule P1-13*
Then for some process term p'', $p' = p'' \cdot y$. From the premise of the rule,

$$\langle x\rangle \xrightarrow{a} \langle p''\rangle$$

By Induction on the above action step, we get:

$$\langle \texttt{consistent}\ x\rangle$$

Apply Rule P1-12. We get:

$$\langle \texttt{consistent}\ x \cdot y\rangle$$

Hence proved.

- *Rule P1-14*
  Then, $p' = y$. From the premise of the rule,

  $$\langle x \rangle \xrightarrow{a} \surd$$

  By Induction on the above predicate, we get:

  $$\langle \texttt{consistent } x \rangle$$

  Apply Rule P1-12. We get:

  $$\langle \texttt{consistent } x \cdot y \rangle$$

  Hence proved.

(b) *Time Step:*
Suppose,

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

It can only be derived from Rule P1-15. From the premise of the rule,

$$\langle x \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

By Induction on the above time step, we get:

$$\langle \texttt{consistent } x \rangle$$

Apply Rule P1-12. We get:

$$\langle \texttt{consistent } x \cdot y \rangle$$

Hence proved.

(c) *Termination Predicate:*
Suppose,

$$\langle x \cdot y \rangle \xrightarrow{a} \surd$$

There are no rules to derive a termination predicate for a sequential composition. Hence the left hand side of the implication does not hold and the implication is trivially satisfied.

4. $p = x + y$.

We prove the four conditions of the theorem one by one.

(a) *Action Step:*
Suppose,

$$\langle x + y \rangle \xrightarrow{a} \langle p' \rangle$$

It can only be derived from Rule P1-19 or Rule P1-20.

- *Rule P1-19*
  From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \langle p' \rangle \tag{H.2}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.3}$$

  By Induction on Transition H.2, we get:

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.4}$$

  Apply Rule P1-16 on Predicates H.3 and H.4. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

  Hence proved.

- *Rule P1-20*
  From the premise of the rule,

$$\langle y \rangle \xrightarrow{a} \langle p' \rangle \tag{H.5}$$

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.6}$$

  By Induction on Transition H.5, we get:

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.7}$$

  Apply Rule P1-16 on Predicates H.6 and H.7. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

  Hence proved.

(b) *Time Step:*
Suppose,

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

It can only be derived from Rule P1-21 or Rule P1-22 or Rule P1-23.

- *Rule P1-21*
  Then for some process terms x_1, y_1, $p' = x_1 + y_1$. From the premise of the rule the following holds:

$$\langle x \rangle \overset{r}{\longmapsto} \langle x_1 \rangle \tag{H.8}$$

$$\langle y \rangle \overset{r}{\longmapsto} \langle y_1 \rangle \tag{H.9}$$

  By Induction on the above time steps, we get:

$$\langle \texttt{consistent}\ \ x \rangle$$

$$\langle \texttt{consistent}\ \ y \rangle$$

  Apply Rule P1-16 on the above Predicates. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

  Hence proved.

- *Rule P1-22*

  From the premise of the rule the following holds:

$$
\begin{align}
\langle x \rangle \xmapsto{r} \langle p' \rangle \tag{H.10}\\
\langle \texttt{consistent}\ \ y \rangle \tag{H.11}\\
\langle y \rangle \overset{r}{\not\mapsto} \tag{H.12}\\
\tag{H.13}
\end{align}
$$

  By Induction on time step H.10, we get:

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.14}$$

  Apply Rule P1-16 on Predicates H.14 and H.11. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

  Hence proved.

- *Rule P1-23*

  From the premise of the rule the following holds:

$$
\begin{align}
\langle y \rangle \xmapsto{r} \langle p' \rangle \tag{H.15}\\
\langle \texttt{consistent}\ \ x \rangle \tag{H.16}\\
\langle x \rangle \overset{r}{\not\mapsto} \tag{H.17}\\
\tag{H.18}
\end{align}
$$

  By Induction on time step H.15, we get:

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.19}$$

  Apply Rule P1-16 on Predicates H.19 and H.16. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

  Hence proved.

(c) *Termination Predicate:*

Suppose,

$$\langle x + y \rangle \xrightarrow{a} \surd$$

It can only be derived from Rule P1-17 or Rule P1-18.

- *Rule P1-17*

  From the premise of the rule,

$$
\begin{align}
\langle x \rangle \xrightarrow{a} \surd \tag{H.20}\\
\langle \texttt{consistent}\ \ y \rangle \tag{H.21}
\end{align}
$$

  By Induction on Predicate H.20, we get:

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.22}$$

  Apply Rule P1-16 on Predicates H.21 and H.22. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

  Hence proved.

- *Rule P1-18*
  From the premise of the rule,

$$\langle y \rangle \xrightarrow{a} \surd \tag{H.23}$$

$$\langle \texttt{consistent}\ x \rangle \tag{H.24}$$

  By Induction on Predicate H.23, we get:

$$\langle \texttt{consistent}\ y \rangle \tag{H.25}$$

  Apply Rule P1-16 on Predicates H.24 and H.25. We get:

$$\langle \texttt{consistent}\ x + y \rangle$$

  Hence proved.

5. $p = \nu_{\mathsf{rel}}(x)$

- *Action Step:*
  Suppose,

$$\langle \nu_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \langle p' \rangle$$

  It can only be derived from Rule P1-26. From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \langle p' \rangle$$

  By Induction on the above action step, we get:

$$\langle \texttt{consistent}\ x \rangle$$

  Apply Rule P1-24. We get:

$$\langle \texttt{consistent}\ \nu_{\mathsf{rel}}(x) \rangle$$

  Hence proved.
- *Time Step:*
  No rule allows a derivation of a time step for the now operator.
- *Termination Predicate:*
  Suppose,

$$\langle \nu_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \surd$$

  It can only be derived from Rule P1-26. From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \surd$$

  By Induction on the above predicate, we get:

$$\langle \texttt{consistent}\ x \rangle$$

  Apply Rule P1-24. We get:

$$\langle \texttt{consistent}\ \nu_{\mathsf{rel}}(x) \rangle$$

  Hence proved.

⊠

H.2 Theorem : Time Determinism

Theorem 13 *For all closed terms p, durations $r > 0$ the following holds:*

$$\begin{aligned} &\langle p \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \wedge \langle p \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \\ &\Longrightarrow p_1 \equiv p_2 \end{aligned}$$

Proof We prove the above theorem by structural induction on a process term $p \in P$. The base case of the structural induction comprises of constant process terms, i.e. all undelayable actions in $\mathcal{A}$, the deadlock process term δ and the inconsistent process $\perp$.
Base Case

1. $p = \tilde{\tilde{a}}$.

   There are no rules to derive a time step for an undelayable action.

2. $p = \tilde{\tilde{\tilde{\delta}}}$

   There are no rules to derive a future Inconsistency predicate for the deadlock constant.

3. $p = \perp$

   There are no rules for an inconsistent process $\perp$.

By Induction Hypothesis

1. $p = \sigma^0_{\mathsf{rel}}(x)$, for a closed term x.

   Suppose,

   $$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.26}$$
   $$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.27}$$

   Only Rule P1-7 allows derivation of a time step for the operator σ^0_{rel}. From the premise of the rule,

   $$\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.28}$$
   $$\langle x \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.29}$$

   By Induction on the above predicate, we get:

   $$p_1 \equiv p_2$$

   Proved.

2. $p = \sigma^t_{\mathsf{rel}}(x) \qquad t > 0$

   Suppose,

   $$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.30}$$
   $$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.31}$$

   We distinguish between three cases depending on the duration r.

(a) Case $r < t$

Let $t = r + r_1$, for some $r_1 > 0$.
Only Rule P1-9 can derive time steps H.30 and H.31. Then the target process terms in both time steps is $\sigma_{\mathsf{rel}}^{r_1}(x)$. I.e.,

$$p_1 = \sigma_{\mathsf{rel}}^{r_1}(x) \wedge p_2 = \sigma_{\mathsf{rel}}^{r_1}(x)$$

Hence

$$p_1 \equiv p_2$$

Proved.

(b) Case $r = t$
Rewriting time steps H.30 and H.31, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \rangle \overset{t}{\longmapsto} \langle p_1 \rangle \tag{H.32}$$

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \rangle \overset{t}{\longmapsto} \langle p_2 \rangle \tag{H.33}$$

Only Rule P1-10 can derive time steps H.32 and H.33. Then the target process terms in both time steps is x. Hence,

$$p_1 \equiv p_2$$

Proved.

(c) Case $r > t$
Let $r = u + t$, for $u > 0$.
Rewriting time steps H.30 and H.31, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \rangle \xmapsto{t+u} \langle p_1 \rangle \tag{H.34}$$

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \rangle \xmapsto{t+u} \langle p_2 \rangle \tag{H.35}$$

Only Rule P1-11 can derive time steps H.34 and H.35. From the premise of the rule, the following must hold:

$$\langle x \rangle \overset{u}{\longmapsto} \langle p_1 \rangle \tag{H.36}$$

$$\langle x \rangle \overset{u}{\longmapsto} \langle p_2 \rangle \tag{H.37}$$

By Induction,

$$p_1 \equiv p_2$$

Proved.

3. $p = x \cdot y$.

Suppose,

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.38}$$

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.39}$$

The above time steps can only be derived from Rule P1-15.

Then for some process term p_1', $p_1 = p_1' \cdot y$.

Rewriting Transition H.38:

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p_1' \cdot y \rangle \tag{H.40}$$

Also for some process term p_2', $p_2 = p_2' \cdot y$.

Rewriting Transition H.39:

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p_2' \cdot y \rangle \tag{H.41}$$

From the premise of Rule P1-15, Transitions H.40 and H.41 can only be derived if the following holds:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1' \rangle \tag{H.42}$$

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_2' \rangle \tag{H.43}$$

By Induction

$$p_1' \equiv p_2'$$

Hence,

$$p_1' \cdot y \equiv p_2' \cdot y \text{ I.e. } p_1 \equiv p_2$$

Proved.

4. $p = x + y$.

Suppose,

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.44}$$

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.45}$$

Rule P1-21, Rule P1-22 or Rule P1-23 can be used to derive the above time steps. We discuss these rules one by one. We show both transitions are derived by the same rule and that only one rule is applicable at a time.

(a) *Rule P1-21*

Suppose Transition H.44 is derived from this rule. Then for some process terms x_1, y_1,

$$p_1 = x_1 + y_1 \tag{H.46}$$

From the premise of the rule the following holds:

$$\langle x \rangle \overset{r}{\longmapsto} \langle x_1 \rangle \tag{H.47}$$

$$\langle y \rangle \overset{r}{\longmapsto} \langle y_1 \rangle \tag{H.48}$$

From Transition H.47, ($\langle x \rangle \overset{r}{\longmapsto} \langle x_1 \rangle$), Rule P1-23 becomes inapplicable to derive a time step for $x + y$.

From Transition H.48, ($\langle y\rangle \overset{r}{\longmapsto} \langle y_1\rangle$), Rule P1-22 becomes inapplicable to derive a time step for $x + y$.

Therefore Transition H.45 can also be only derived by Rule P1-21. From the premise of the rule, for some process terms x_2, y_2,

$$p_2 = x_2 + y_2 \tag{H.49}$$

and the following must hold:

$$\langle x\rangle \overset{r}{\longmapsto} \langle x_2\rangle \tag{H.50}$$

$$\langle y\rangle \overset{r}{\longmapsto} \langle y_2\rangle \tag{H.51}$$

Apply Induction Hypothesis on Transitions H.47 and H.50, and on Transitions H.48 and H.51. We get:

$$x_1 \equiv x_2$$

$$y_1 \equiv y_2$$

which implies

$$x_1 + y_1 \equiv x_2 + y_2$$

From Statements H.46 and H.49,

$$p_1 \equiv p_2$$

Proved.

(b) *Rule P1-22*

Suppose Transition H.44 is derived from this rule. From the premise of the rule the following holds:

$$\langle x\rangle \overset{r}{\longmapsto} \langle p_1\rangle \tag{H.52}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{H.53}$$

$$\langle y\rangle \overset{r}{\not\longmapsto} \tag{H.54}$$

From Transition H.52, ($\langle x\rangle \overset{r}{\longmapsto} \langle p_1\rangle$), Rule P1-23 becomes inapplicable to derive a time step for $x + y$.

From Transition H.54, ($\langle y\rangle \overset{r}{\not\longmapsto}$), Rule P1-21 becomes inapplicable to derive a time step for $x + y$.

Hence Transition H.45 can only be derived from Rule P1-22.

From the premise of the rule, in addition to Predicates H.53 and H.54, the following holds :

$$\langle x\rangle \overset{r}{\longmapsto} \langle p_2\rangle \tag{H.55}$$

Apply Induction Hypothesis on Transition H.52 and Transition H.55, we get:

$$p_1 \equiv p_2$$

Proved.

(c) *Rule P1-23*

Suppose Transition H.44 is derived from this rule. From the premise of the rule the following holds:

$$\langle y\rangle \xmapsto{r} \langle p_1\rangle \tag{H.56}$$

$$\langle \texttt{consistent}\ x\rangle \tag{H.57}$$

$$\langle x\rangle \not\xmapsto{r} \tag{H.58}$$

From Transition H.56, ($\langle y\rangle \xmapsto{r} \langle p_1\rangle$), Rule P1-22 becomes inapplicable to derive a time step for $x + y$.

From Transition H.58, ($\langle x\rangle \not\xmapsto{r}$), Rule P1-21 becomes inapplicable to derive a time step for $x + y$.

Hence Transition H.45 can only be derived from Rule P1-23.

From the premise of the rule, in addition to Predicates H.57 and H.58 the following holds:

$$\langle y\rangle \xmapsto{r} \langle p_2\rangle \tag{H.59}$$

Apply Induction Hypothesis on Transition H.56 and Transition H.59, we get:

$$p_1 \equiv p_2$$

Proved.

5. $p = \nu_{\mathsf{rel}}(x)$

   There are no rules to derive a time step for the now operator. Hence the theorem trivially holds.

⊠

H.3 Axiom A1 (Commutativity)

$$x + y = y + x$$

We prove that the choice operator + is commutative by showing that the TSS $T = (\Sigma, D)$, with Σ the signature of $BPA_{\perp}^{srt}$, and D the set of deduction rules in Table 3.9, is in comm-tyft format (extended with predicates and negative premises) given in [Mou05].

Define the mapping $\hbar$ defined on variables x, y, x', y' as follows:

$$\hbar(x) = y \quad \hbar(y) = x \quad \hbar(x') = y' \quad \hbar(y') = x'$$

Then it easy to prove that Rules P1-16 and P1-21 are commutative mirrors of themselves. Rules P1-17 and P1-18 are commutative mirrors of each other. Similarly, Rules P1-19 and P1-20, and Rules P1-22 and P1-23 are commutative mirrors of each other.

H.4 Axiom A2 (Associativity)

$$x + (y + z) = (x + y) + z$$

We need to prove,

$$x + (y + z) \leftrightarrows (x + y) + z$$

Let

$$R = \{((x + y) + z, x + (y + z)) \mid x, y, z \in P\}$$

be a binary relation on process terms.

We prove that the relation $R \cup \mathcal{I}$ is the witness relation for bisimilarity of $x + (y + z)$ and $(x + y) + z$. We show that all pairs in R satisfy the conditions of bisimulation. For $(x, x) \in \mathcal{I}$, it is trivial that all properties of bisimulation are satisfied.

For all $a \in A, x, y, z, p \in P, r > 0$, the following holds:

1. $\langle (x + y) + z\rangle \xrightarrow{a} \langle p\rangle \implies \exists p' \in P : \langle x + (y + z)\rangle \xrightarrow{a} \langle p'\rangle$ and $(p, p') \in R \cup \mathcal{I}$.

   Suppose,

$$\langle (x + y) + z\rangle \xrightarrow{a} \langle p\rangle \tag{H.60}$$

   The above transition can only be derived using Rules P1-19 or P1 20.

   (a) Rule P1-19

   Then we must have:

$$\langle x + y\rangle \xrightarrow{a} \langle p\rangle \tag{H.61}$$

$$\langle \texttt{consistent}\ z\rangle \tag{H.62}$$

   Again Transition H.61 can be obtained using rules P1-19 or P1 20.

   i. *Rule P1 19*

   Then in Transition H.61, the left most process term must perform the action and the other process term must be consistent. We have:

$$\langle x\rangle \xrightarrow{a} \langle p\rangle \tag{H.63}$$

$$\langle \texttt{consistent}\ y\rangle \tag{H.64}$$

   From Predicates H.62 and H.64:

$$\langle \texttt{consistent}\ y\rangle \wedge \langle \texttt{consistent}\ z\rangle$$

   Hence

$$\langle \texttt{consistent}\ y + z\rangle$$

   Using Rule P1-19:

$$\langle x + (y + z)\rangle \xrightarrow{a} \langle p\rangle \tag{H.65}$$

   Consider the target process terms in Transitions H.60 and H.65. The pair (p, p) is in $\mathcal{I}$.

ii. <u>*Rule P1 20*</u>

Then in Transition H.61, the right most process term must perform the action and the other process term must be consistent. We have:

$$\langle y \rangle \xrightarrow{a} \langle p \rangle, \tag{H.66}$$

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.67}$$

Apply Rule P1-19 on Transition H.66 using Predicate H.62. We get:

$$\langle y + z \rangle \xrightarrow{a} \langle p \rangle$$

Taking $\langle \texttt{consistent}\ \ x \rangle$ from Predicate H.67, apply Rule P1-20 on the above transition, we get:

$$\langle x + (y + z) \rangle \xrightarrow{a} \langle p \rangle \tag{H.68}$$

Consider the target process terms in Transitions H.60 and H.68. The pair (p, p) is in $\mathcal{I}$.

(b) <u>Rule P1-20</u>

If transition H.60 is derived using rule P1 20, then the process term z must perform the action, i.e.:

$$\langle z \rangle \xrightarrow{a} \langle p \rangle, \tag{H.69}$$

$$\langle \texttt{consistent}\ \ x + y \rangle \tag{H.70}$$

$\langle \texttt{consistent}\ \ x + y \rangle$ only holds if:

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.71}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.72}$$

Apply Rule P1 20 on Transition H.69 using Predicate H.72:

$$\langle y + z \rangle \xrightarrow{a} \langle p \rangle$$

Again apply Rule P1 20 on the above transition using Predicate H.71:

$$\langle x + (y + z) \rangle \xrightarrow{a} \langle p \rangle \tag{H.73}$$

Consider the target process terms in Transitions H.60 and H.73. The pair (p, p) is in $\mathcal{I}$.

2. $\langle x + (y + z) \rangle \xrightarrow{a} \langle p \rangle \implies \exists p' \in P : \langle (x + y) + z \rangle \xrightarrow{a} \langle p' \rangle$ and $(p', p) \in R \cup \mathcal{I}$.

Suppose,

$$\langle x + (y + z) \rangle \xrightarrow{a} \langle p \rangle \tag{H.74}$$

The above transition can only be derived from rules P1-19 or P1-20.

(a) Rule P1-19

If Rule P1-19 is used to derive Transition H.74, then the left most process term, i.e. x must perform action a and the other process term $y + z$ must be consistent. Therefore,

$$\langle x \rangle \xrightarrow{a} \langle p \rangle \tag{H.75}$$

$$\langle \texttt{consistent}\ \ y + z \rangle \tag{H.76}$$

The predicate $\langle \texttt{consistent}\ \ y + z \rangle$ can only hold if:

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.77}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{H.78}$$

Apply Rule P1-19 on Transition H.75 and Predicate H.77:

$$\langle x + y \rangle \xrightarrow{a} \langle p \rangle$$

Again apply Rule P1-19 on the above transition with Predicate H.78:

$$\langle (x + y) + z \rangle \xrightarrow{a} \langle p \rangle \tag{H.79}$$

Consider the target process terms in Transitions H.74 and H.79. The pair (p, p) is in $\mathcal{I}$.

(b) Rule P1-20

If Rule P1-20 is used to derive Transition H.74, then the right most process term, i.e. $(y + z)$ must perform action a and the other process term x must be consistent. Therefore,

$$\langle y + z \rangle \xrightarrow{a} \langle p \rangle \tag{H.80}$$

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.81}$$

Transition $H.80$ can only be obtained by using rules P1-19 or P1 20.

i. *Rule P1 19:*

Premise of P1-19:

$$\langle y \rangle \xrightarrow{a} \langle p \rangle \tag{H.82}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{H.83}$$

Apply rule P1-20 on Transition H.82 using Predicate H.81::

$$\langle x + y \rangle \xrightarrow{a} \langle p \rangle \tag{H.84}$$

Again applying P1-19 on the above transition using Predicate H.83::

$$\langle (x + y) + z \rangle \xrightarrow{a} \langle p \rangle \tag{H.85}$$

Consider the target process terms in Transitions H.74 and H.85. The pair (p, p) is in $\mathcal{I}$.

ii. Rule P1 20:

Suppose Transition H.80 has been derived from Rule P1-20. Then from the premise of Rule P1 20:

$$\langle z \rangle \xrightarrow{a} \langle p \rangle \tag{H.86}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.87}$$

Again by Rule P1 20, for any process term q with $\langle \texttt{consistent}\ \ q \rangle$:

$$\langle p_1 + z \rangle \xrightarrow{a} \langle p \rangle \tag{H.88}$$

From Predicates H.81 and H.87:

$$\langle \texttt{consistent}\ \ x \rangle \wedge \langle \texttt{consistent}\ \ y \rangle$$

which implies:$\langle \texttt{consistent}\ \ x + y \rangle$.
Put $q = x + y$ in Transition H.88, we get the desired transition:

$$\langle (x + y) + z \rangle \xrightarrow{a} \langle p \rangle$$

And $(p, p) \in \mathcal{I}$.

3. $\langle x + (y + z) \rangle \xrightarrow{a} \langle \surd \rangle \iff \langle (x + y) + z \rangle \xrightarrow{a} \langle \surd \rangle$.
Reasoning similar to above applies.

4. $\langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p \rangle \implies \exists p' \in P : \langle x + (y + z) \rangle \overset{r}{\longmapsto} \langle p' \rangle$ and $(p, p') \in R \cup \mathcal{I}$.
Suppose,

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.89}$$

The above time transition can be derived from rules P1-21, P1-22 or P1-refprop1rule:alt:delayoneright.

(a) Rule P1-21
Then for some process terms p_1, p_2, the process term p in transition H.89 must be of the following form:

$$p = p_1 + p_2$$

Rewriting Transition H.89:

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p_1 + p_2 \rangle \tag{H.90}$$

From the premise of the rule, following must be derivable:

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.91}$$

$$\langle z \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.92}$$

Again Transition H.91 can be obtained from rules P1-21, P1-22 or P1-23.

i. *Rule P1-21*
Then $p_1 = q_1 + q_2$, for some $q_1, q_2 \in P$.
Rewriting Transitions H.90 and H.91. We get:

$$\langle (x+y)+z \rangle \overset{r}{\longmapsto} \langle (q_1+q_2)+p_2 \rangle \tag{H.93}$$
$$\langle x+y \rangle \overset{r}{\longmapsto} \langle q_1+q_2 \rangle \tag{H.94}$$

From the premise of the rule, the following is derivable:

$$\langle x \rangle \overset{r}{\longmapsto} \langle q_1 \rangle \tag{H.95}$$
$$\langle y \rangle \overset{r}{\longmapsto} \langle q_2 \rangle \tag{H.96}$$

Apply Rule P1-21 on Transitions H.92 and H.96, we get:

$$\langle y+z \rangle \overset{r}{\longmapsto} \langle q_2+p_2 \rangle \tag{H.97}$$

Again apply Rule P1-21 on Transitions H.95 and H.97, we get:

$$\langle x+(y+z) \rangle \overset{r}{\longmapsto} \langle q_1+(q_2+p_2) \rangle \tag{H.98}$$

Consider the target process terms in Transitions H.89 and H.98. The pair $((q_1+q_2)+p_2, q_1+(q_2+p_2))$ is in R.

ii. *Rule P1-22*

Transition H.91 can also be derived using Rule P1 22. Then the following must be derivable:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.99}$$
$$\langle y \rangle \overset{r}{\not\longmapsto} \tag{H.100}$$
$$\langle \texttt{consistent}\ \ y \rangle \tag{H.101}$$

Combine Transition H.92 and Predicates H.100 and H.101. Apply Rule P1-23, we get:

$$\langle y+z \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.102}$$

Now apply Rule P1-21 on Transitions H.102 and H.99. We get:

$$\langle x+(y+z) \rangle \overset{r}{\longmapsto} \langle p_1+p_2 \rangle \tag{H.103}$$

Consider the target process terms in Transitions H.103 and H.90. The pair (p_1+p_2, p_1+p_2) is in $\mathcal{I}$.

iii. *Rule P1-23*
Transition H.91 can also be obtained by Rule P1-23. Then:

$$\langle y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.104}$$
$$\langle x \rangle \overset{r}{\not\longmapsto} \tag{H.105}$$
$$\langle \texttt{consistent}\ \ x \rangle \tag{H.106}$$

From Transition H.92, process term z can delay as follows:

$$\langle z \rangle \xmapsto{r} \langle p_2 \rangle \qquad (H.92)$$

Apply Rule P1-21 on Transitions H.92 and H.104. We get:

$$\langle y + z \rangle \xmapsto{r} \langle p_1 + p_2 \rangle \tag{H.107}$$

On Transition H.107 and Predicates H.105 and H.106, apply Rule P1-23:

$$\langle x + (y + z) \rangle \xmapsto{r} \langle p_1 + p_2 \rangle \tag{H.108}$$

Consider the target process terms in Transitions H.108 and H.90. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

(b) Rule P1-22

$$\langle (x + y) + z \rangle \xmapsto{r} \langle p \rangle \qquad (H.89)$$

Transition H.89 can also be obtained by applying Rule P1-22. Then from the premise of the rule the following must be derivable:

$$\langle x + y \rangle \xmapsto{r} \langle p \rangle \tag{H.109}$$

$$\langle z \rangle \not\xmapsto{r} \tag{H.110}$$

$$\langle \texttt{consistent}\ z \rangle \tag{H.111}$$

Again Transition H.109 can be derived by rules P1-21, P1-22 or P1-23.

i. *Rule P1-21*

If Transition H.109 is derived from Rule P1 21, then for some $p_1, p_2 \in P$, $p = p_1 + p_2$. Rewriting Transitions H.89 and H.109, we get:

$$\langle (x + y) + z \rangle \xmapsto{r} \langle p_1 + p_2 \rangle \tag{H.112}$$

$$\langle x + y \rangle \xmapsto{r} \langle p_1 + p_2 \rangle \tag{H.113}$$

And the following must be derivable:

$$\langle x \rangle \xmapsto{r} \langle p_1 \rangle \tag{H.114}$$

$$\langle y \rangle \xmapsto{r} \langle p_2 \rangle \tag{H.115}$$

Apply Rule P1-22 on Transition H.115 and Predicates H.110 and H.111. We get:

$$\langle y + z \rangle \xmapsto{r} \langle p_2 \rangle \tag{H.116}$$

Combine Transition H.114 and Transition H.116 and apply Rule P1-21. We get:

$$\langle x + (y + z) \rangle \xmapsto{r} \langle p_1 + p_2 \rangle \tag{H.117}$$

Consider the target process terms in Transitions H.112 and H.117. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

ii. *Rule P1-22*

If Transition H.109 is derived from Rule P1-22, then the following must be derivable:

$$\langle x\rangle \xmapsto{r} \langle p\rangle \tag{H.118}$$

$$\langle y\rangle \not\xmapsto{r} \tag{H.119}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{H.120}$$

Combine Predicates H.110 and H.119, Predicates H.111 and H.120. We can infer:

$$\langle y+z\rangle \not\xmapsto{r} \tag{H.121}$$

$$\langle \texttt{consistent}\ \ y+z\rangle \tag{H.122}$$

Combine Transition H.118 and Predicates H.121 and H.122. Apply Rule P1-22. We get:

$$\langle x+(y+z)\rangle \xmapsto{r} \langle p\rangle \tag{H.123}$$

Consider the target process terms in Transitions H.89 and H.123. The pair (p, p) is in $\mathcal{I}$.

iii. *Rule P1-23*

If Transition H.109 is derived from Rule P1 23, then the following must be derivable:

$$\langle y\rangle \xmapsto{r} \langle p\rangle \tag{H.124}$$

$$\langle x\rangle \not\xmapsto{r} \tag{H.125}$$

$$\langle \texttt{consistent}\ \ x\rangle \tag{H.126}$$

Combine Transition H.124 and Predicates H.110 and H.111. Apply Rule P1-22. We get:

$$\langle y+z\rangle \xmapsto{r} \langle p\rangle \tag{H.127}$$

Combine Predicates H.125, H.126 and Transition H.127 and apply Rule P1-23. We get:

$$\langle x+(y+z)\rangle \xmapsto{r} \langle p\rangle \tag{H.128}$$

Consider the target process terms in Transitions H.89 and H.128. The pair (p, p) is in $\mathcal{I}$.

(c) Rule P1-23

$$\langle (x+y)+z\rangle \xmapsto{r} \langle p\rangle \qquad (H.89)$$

Transition H.89 can also be obtained by applying Rule P1-23. Then from the premise of the rule the following must be derivable:

$$\langle z \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.129}$$

$$\langle x + y \rangle \overset{r}{\not\longmapsto} \tag{H.130}$$

$$\langle \texttt{consistent}\ x + y \rangle \tag{H.131}$$

Predicate H.130 can only hold if none of the process terms x and y can do a time step with duration r. Therefore the following holds:

$$\langle x \rangle \overset{r}{\not\longmapsto} \tag{H.132}$$

$$\langle y \rangle \overset{r}{\not\longmapsto} \tag{H.133}$$

From predicate H.131, we can infer the following:

$$\langle \texttt{consistent}\ x \rangle \tag{H.134}$$

$$\langle \texttt{consistent}\ y \rangle \tag{H.135}$$

Combine Transition H.129 and Predicates H.133 and H.135 and apply rule P1-23. We get:

$$\langle y + z \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.136}$$

Again combine Transition H.136 and Predicates H.132 and H.134 and apply rule P1-23. We get:

$$\langle x + (y + z) \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.137}$$

Consider the target process terms in Transitions H.89 and H.137. The pair (p, p) is in $\mathcal{I}$.

5. $\langle x + (y + z) \rangle \overset{r}{\longmapsto} \langle p \rangle \implies \exists p' \in P : \langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p' \rangle$ and $(p', p) \in R \cup \mathcal{I}$.

   Suppose,

$$\langle x + (y + z) \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.138}$$

The above time transition can be derived from rules P1-21, P1-22 or P1-refprop1rule:alt:delayoneright.

(a) <u>Rule P1-21</u>

Then for some process terms p_1, p_2, the process term p (in Transition H.138) can be written as:

$$p = p_1 + p_2 \tag{H.139}$$

Rewriting Transition H.138:

$$\langle x + (y + z) \rangle \overset{r}{\longmapsto} \langle p_1 + p_2 \rangle \tag{H.140}$$

From the premise of Rule P1-21, the following is derivable:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.141}$$

$$\langle y + z \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{H.142}$$

Again Transition H.142 can be obtained from rules P1-21, P1-22 or P1-23.

i. *Rule P1-21*

Then $p_2 = q_1 + q_2$, for some $q_1, q_2 \in P$.
Rewriting Transitions H.138 and H.142:

$$\langle x + (y + z)\rangle \overset{r}{\longmapsto} \langle p_1 + (q_1 + q_2)\rangle \tag{H.143}$$

$$\langle y + z\rangle \overset{r}{\longmapsto} \langle q_1 + q_2\rangle \tag{H.144}$$

and the following is derivable:

$$\langle y\rangle \overset{r}{\longmapsto} \langle q_1\rangle \tag{H.145}$$

$$\langle z\rangle \overset{r}{\longmapsto} \langle q_2\rangle \tag{H.146}$$

Apply Rule P1-21 on Transitions H.141 and H.145, we get:

$$\langle x + y\rangle \overset{r}{\longmapsto} \langle p_1 + q_1\rangle \tag{H.147}$$

Again apply Rule P1-21 on Transitions H.147 and H.146, we get:

$$\langle (x + y) + z\rangle \overset{r}{\longmapsto} \langle (p_1 + q_1) + q_2\rangle \tag{H.148}$$

Consider the target process terms in Transitions H.143 and H.148. The pair $((p_1 + q_1) + q_2, p_1 + (q_1 + q_2))$ is in R.

ii. *Rule P1-22*

If Transition H.142 is derived using Rule P1 22. Then the following must be hold:

$$\langle y\rangle \overset{r}{\longmapsto} \langle p_2\rangle \tag{H.149}$$

$$\langle z\rangle \overset{r}{\not\longmapsto} \tag{H.150}$$

$$\langle \texttt{consistent}\ z\rangle \tag{H.151}$$

Combine Transitions H.141 and H.149 and apply Rule P1-21, we get:

$$\langle x + y\rangle \overset{r}{\longmapsto} \langle p_1 + p_2\rangle \tag{H.152}$$

Now apply Rule P1-22 on Transition H.152 and Predicates H.150 and H.151. We get:

$$\langle (x + y) + z\rangle \overset{r}{\longmapsto} \langle p_1 + p_2\rangle \tag{H.153}$$

Consider the target process terms in Transitions H.140 and H.153. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

iii. *Rule P1-23*

Transition H.142 can also be obtained by Rule P1-23. Then:

$$\langle z\rangle \overset{r}{\longmapsto} \langle p_2\rangle \tag{H.154}$$

$$\langle y\rangle \overset{r}{\not\longmapsto} \tag{H.155}$$

$$\langle \texttt{consistent}\ y\rangle \tag{H.156}$$

From Transition H.141, process term x can delay as follows:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \qquad (H.141)$$

Apply Rule P1-22 on Transition H.141 and Predicates H.155 and H.156. We get:

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{H.157}$$

Joining Transitions H.154 and H.157 and apply Rule P1-21:

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p_1 + p_2 \rangle \tag{H.158}$$

Consider the target process terms in Transitions H.140 and H.158. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

(b) <u>Rule P1-22</u>

Transition H.138 can also be derived from rule P1 22. Then from the premise of the rule:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.159}$$

$$\langle y + z \rangle \overset{r}{\not\longmapsto} \tag{H.160}$$

$$\langle \texttt{consistent}\ \ y + z \rangle \tag{H.161}$$

Predicate H.161 implies:

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.162}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{H.163}$$

Predicate H.160 can only hold if **none of the rules** P1-21, P1-22 or P1-23 can be applied to derive a time transition for $\langle y + z \rangle$ with duration r.
Rule P1-21 cannot be applied only if $\langle y \rangle$ and $\langle z \rangle$ cannot both do a time transition with delay r. Suppose one of them can do the time step and the other cannot. Then if $\langle y \rangle$ can delay, then rule P1-22 can be used to derive a time transition for $\langle y + z \rangle$. If $\langle z \rangle$ can delay, then rule P1-23 can be used to derive a time transition for $\langle y + z \rangle$. Hence predicate H.160 only holds if none of the process term x, y can do a time transition with delay $(r,)$.
Therefore,

$$\langle y \rangle \overset{r}{\not\longmapsto} \tag{H.164}$$

$$\langle z \rangle \overset{r}{\not\longmapsto} \tag{H.165}$$

Apply Rule P1-22 on Transition H.159, Predicate H.164 and Predicate H.162, we get:

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.166}$$

Again apply Rule P1-22 on Transition H.166 and Predicates H.165 and H.163, we get:

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.167}$$

Consider the target process terms in Transitions H.138 and H.167. The pair (p, p) is in $\mathcal{I}$.

(c) Rule P1-23

Transition H.138 can also be derived from rule P1 23. Then from the premise of the rule:

$$\langle y + z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{H.168}$$

$$\langle x\rangle \overset{r}{\not\longmapsto} \tag{H.169}$$

$$\langle \texttt{consistent}\ x\rangle \tag{H.170}$$

Again Transition H.168 can be derived by rules P1-21, P1-22 or P1-23.

i. *Rule P1-21*

If Transition H.168 is derived from Rule P1 21, then for some $p_1, p_2 \in P$, $p = p_1 + p_2$. Rewriting Transitions H.138 and H.168

$$\langle x + (y + z)\rangle \overset{r}{\longmapsto} \langle p_1 + p_2\rangle \tag{H.171}$$

$$\langle y + z\rangle \overset{r}{\longmapsto} \langle p_1 + p_2\rangle \tag{H.172}$$

And the following must be derivable:

$$\langle y\rangle \overset{r}{\longmapsto} \langle p_1\rangle \tag{H.173}$$

$$\langle z\rangle \overset{r}{\longmapsto} \langle p_2\rangle \tag{H.174}$$

Combine Predicates H.169, H.170 and Transition H.173 and apply Rule P1-23. We get:

$$\langle x + y\rangle \overset{r}{\longmapsto} \langle p_1\rangle \tag{H.175}$$

Combine Transitions H.174 and Transitions H.175 and apply Rule P1-21. We get:

$$\langle (x + y) + z\rangle \overset{r}{\longmapsto} \langle p_1 + p_2\rangle \tag{H.176}$$

Consider the target process terms in Transitions H.171 and H.176. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

ii. *Rule P1-22*

If Transition H.168 is derived from Rule P1 22, then the following must be derivable:

$$\langle y\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{H.177}$$

$$\langle z\rangle \overset{r}{\not\longmapsto} \tag{H.178}$$

$$\langle \texttt{consistent}\ z\rangle \tag{H.179}$$

Combine Predicates H.169 and H.170 and Transition H.177 and apply Rule P1-23. We get:

$$\langle x + y\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{H.180}$$

Combine Predicates H.178 and H.179 and Transition H.180 and apply Rule P1-22. We get:

$$\langle (x+y)+z \rangle \xmapsto{r} \langle p \rangle \tag{H.181}$$

Consider the target process terms in Transitions H.168 and H.181. The pair (p, p) is in $\mathcal{I}$.

iii. *Rule P1-23*

If Transition H.168 is derived from Rule P1 23, then the following must be derivable:

$$\langle z \rangle \xmapsto{r} \langle p \rangle \tag{H.182}$$

$$\langle y \rangle \not\xmapsto{r} \tag{H.183}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.184}$$

Combine Predicates H.169 and H.184. None of the rules for delay of an alternative composition can be applied. Hence the following predicate holds:

$$\langle x+y \rangle \not\xmapsto{r} \tag{H.185}$$

Combine Predicates H.170 and H.184. The following predicate holds:

$$\langle \texttt{consistent}\ \ x+y \rangle \tag{H.186}$$

Combine Transition H.182, Predicates H.185 and H.186. Apply Rule P1-23. We get:

$$\langle (x+y)+z \rangle \xmapsto{r} \langle p \rangle \tag{H.187}$$

Consider the target process terms in Transitions H.168 and H.187. The pair (p, p) is in $\mathcal{I}$.

6.

$$\langle \texttt{consistent}\ \ (x+y)+z \rangle \iff \langle \texttt{consistent}\ \ x+(y+z) \rangle$$

Left Implication

Suppose

$$\in [\mathsf{s}((x+y)+z)]$$

This can only be derived from Rule P1-16. From the premise of the rule, the following must hold:

$$\langle \texttt{consistent}\ \ (x+y) \rangle \tag{H.188}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{H.189}$$

Again Predicate H.188 can only be derived from Rule P1-16. Then the following holds:

$$\langle \texttt{consistent}\ \ x \rangle \tag{H.190}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.191}$$

Combine predicates H.191 and H.189 and apply Rule P1-16:

$$\langle \texttt{consistent}\ \ y + z \rangle \tag{H.192}$$

Again combine Predicate H.190 and Predicate H.192 apply Rule P1-16. We get:

$$\langle \texttt{consistent}\ \ x + (y + z) \rangle$$

Left Implication

Suppose

$$\langle \texttt{consistent}\ \ x + (y + z) \rangle$$

By similar reasoning, as given above, the following can be derived.

$$\langle \texttt{consistent}\ \ (x + y) + z \rangle$$

H.5 Axiom SRTD (Conditional Time Determinism)

$$\begin{array}{ll} \sigma^{v}_{\mathsf{rel}}(x) + \sigma^{v}_{\mathsf{rel}}(y) \leftrightarrows \sigma^{v}_{\mathsf{rel}}(x + y) & (SRT3) \\ \text{where } \langle \texttt{consistent}\ \ x \rangle \wedge \langle \texttt{consistent}\ \ y \rangle & \end{array}$$

where $v \geq 0$.

Proof

We prove the soundness of Axiom SRTD in two steps.

Case $v = 0$

By Axiom SRT1, we know that for any process term x,

$$\sigma^{0}_{\mathsf{rel}}(x) \leftrightarrows x$$

Since Bisimulation is a congruence therefore, then it becomes trivial to prove that:

$$\sigma^{0}_{\mathsf{rel}}(x) + \sigma^{0}_{\mathsf{rel}}(y) \leftrightarrows \sigma^{0}_{\mathsf{rel}}(x + y)$$

Case $v > 0$

Let R be the following relation:

$$R = \{(\sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)), \sigma^{t}_{\mathsf{rel}}(x + y) \mid 0 < t \leq v, x, y \in P\}$$

We prove that $R \cup \mathcal{I}$ is a bisimulation relation:

For all $a \in A, r > 0, z \in P$:

1.

$$\begin{array}{lll} \langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \langle z \rangle & \Longrightarrow & \exists z' \in P : \langle \sigma^{t}_{\mathsf{rel}}(x + y) \rangle \xrightarrow{a} \langle z' \rangle \\ & & (z, z') \in R \cup \mathcal{I} \end{array}$$

Trivial.

2.

$$\langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \xrightarrow{a} \langle z\rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xrightarrow{a} \langle z'\rangle \\ (z', z) \in R \cup \mathcal{I} \end{array}$$

Trivial.

3.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \xrightarrow{a} \surd \iff \langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \xrightarrow{a} \surd$$

Trivial.

4.

$$\langle \texttt{consistent}\ \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \iff \langle \texttt{consistent}\ \sigma^t_{\mathsf{rel}}(x+y)\rangle$$

Trivial.

5.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle z'\rangle \\ (z, z') \in R \cup \mathcal{I} \end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \tag{H.193}$$

This can be derived from Rules P1-21, P1-22 and P1-23.

(a) Rule P1-21
If Transition H.193 is derived from this rule, then

$$z = z_1 + z_2$$

And the following holds:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z_1\rangle \tag{H.194}$$

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z_2\rangle \tag{H.195}$$

We distinguish between three cases:

i. Case $r < t$

Let $t = r + r_1$. Then both Transitions H.194 and H.195 are derived from Rule P1-9. Then

$$z_1 = \sigma^{r_1}_{\mathsf{rel}}(x) \text{ and } z_2 = \sigma^{r_1}_{\mathsf{rel}}(y)$$

Rewriting Transition H.193:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) + \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y)\rangle \tag{H.196}$$

By Rule P1-9, the following can be derived:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x+y)\rangle \tag{H.197}$$

ii. Case $r = t$

Then both Transitions H.194 an dH.195 are derived from Rule P1-10. Then

$$z_1 = x \text{ and } z_2 = y$$

From premise of Rule P1-10, the following holds:

$$\langle \texttt{consistent}\ x\rangle \text{ and } \langle \texttt{consistent}\ y\rangle$$

Apply Rule P1-16 on the above predicates, we get:

$$\langle \texttt{consistent}\ x + y\rangle$$

Then Rule P1-10 becomes applicable to derive the following:

$$\langle \sigma^{r}_{\mathsf{rel}}(x + y)\rangle \overset{r}{\longmapsto} \langle x + y\rangle \tag{H.198}$$

iii. Case $r > t$

Let $r = t + t_1$.
Rewriting Transitions H.194 and H.195:

$$\langle \sigma^{t}_{\mathsf{rel}}(x)\rangle \overset{t+t_1}{\longmapsto} \langle z_1\rangle \tag{H.199}$$

$$\langle \sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{t+t_1}{\longmapsto} \langle z_2\rangle \tag{H.200}$$

The above transitions can only be derived from Rule P1-11.
From the premise of the rule, the following holds:

$$\langle x\rangle \overset{t_1}{\longmapsto} \langle z_1\rangle \tag{H.201}$$

$$\langle y\rangle \overset{t_1}{\longmapsto} \langle z_2\rangle \tag{H.202}$$

Apply Rule P1-21 on the above transitions:

$$\langle x + y\rangle \overset{t_1}{\longmapsto} \langle z_1 + z_2\rangle \tag{H.203}$$

Apply Rule P1-11 on the above transitions, we get:

$$\langle \sigma^{t+t_1}_{\mathsf{rel}}(x + y)\rangle \overset{t+t_1}{\longmapsto} \langle z_1 + z_2\rangle \tag{H.204}$$

(b) Rule P1-22
Suppose, Transition H.193 (which is repeated below) is derived from this rule.

$$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z\rangle \qquad (H.193)$$

Then from the premise of the rule, the following holds:

$$\langle \sigma^{t}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z\rangle \tag{H.205}$$

$$\langle \texttt{consistent}\ \sigma^{t}_{\mathsf{rel}}(y)\rangle \tag{H.206}$$

$$\langle \sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{r}{\not\longmapsto} \tag{H.207}$$

Again we distinguish between three cases:

i. Case $r < t$

For $r < t$, Predicate H.207 can not be derived. We conclude that Rule P1-22 cannot be used to derive Transition H.193 for $r < t$.

ii. Case $r = t$

Then Predicate H.207can only be derived if y is inconsistent. I.e.

$$\neg\langle \texttt{consistent } y\rangle$$

The axiom SRTD is conditional and for inconsistent y, we do not need to derive a corresponding transition of $\sigma^t_{\mathsf{rel}}(x + y)$.

iii. Case $r > t$

Let $r = t + t_1$. Rewriting Transitions H.205, H.206 and H.207:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+t_1} \langle z\rangle \tag{H.208}$$

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(y)\rangle \tag{H.209}$$

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{t+t_1} \tag{H.210}$$

Then Transition H.208 can only be derived by Rule P1-11. From the premise of the rule, the following holds:

$$\langle x\rangle \xmapsto{t_1} \langle z\rangle \tag{H.211}$$

Predicate H.210 can hold only if Rule P1-11 is not appllicable on process term $\sigma^t_{\mathsf{rel}}(y)$. Hence, the premise of the rule must not hold. I.e:

$$\langle y\rangle \not\xmapsto{t_1} \tag{H.212}$$

Since, we are proving the axiom for consistent process terms, hence:

$$\langle \texttt{consistent } y\rangle \tag{H.213}$$

Apply Rule P1-22 on Transitions H.211,H.212 and H.213, we get:

$$\langle x + y\rangle \xmapsto{t_1} \langle z\rangle \tag{H.214}$$

Apply Rule P1-11 on the above transition, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x + y)\rangle \xmapsto{t+t_1} \langle z\rangle \tag{H.215}$$

(c) Rule P1-23

Same reasoning as applied for Rule P1-22.

6.
$$\langle \sigma^{t}_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle z\rangle \implies \begin{array}{l}\exists z' \in P : \langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle z'\rangle \\ (z', z) \in R\end{array}$$

Suppose,

$$\langle \sigma^{t}_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle z\rangle \tag{H.216}$$

We distinguish between three cases for different values of r.

(a) $\underline{\text{Case } r < t}$
Let $t = r + r_1$. Then Transition H.216 is derived from Rule P1-9. Then

$$z = \sigma^{r_1}_{\mathsf{rel}}(x+y)$$

Rewriting Transition H.216:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x+y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x+y)\rangle \tag{H.217}$$

By Rule P1-9, the following can be derived:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x)\rangle \tag{H.218}$$

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(y)\rangle \tag{H.219}$$

Apply Rule P1-21 on the above transitions, we get:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) + \sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y)\rangle \tag{H.220}$$

(b) $\underline{\text{Case } r = t}$
Then Transition H.216 is derived from Rule P1-10. Then

$$z = x$$

From the premise of the rule, the folllowing holds:

$$\langle \texttt{consistent}\ x + y\rangle$$

which implies

$$\langle \texttt{consistent}\ x\rangle \text{ and } \langle \texttt{consistent}\ y\rangle$$

Then Rule P1-10 becomes applicable to derive the following:

$$\langle \sigma^{r}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle x\rangle \tag{H.221}$$

$$\langle \sigma^{r}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle y\rangle \tag{H.222}$$

Apply Rule P1-21 on the above transitions, we get:

$$\langle \sigma^{r}_{\mathsf{rel}}(x) + \sigma^{r}_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto} \langle x+y\rangle \tag{H.223}$$

(c) Case $r > t$
Let $r = t + t_1$.
Rewriting Transition H.216:

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{t+t_1} \langle z \rangle \tag{H.224}$$

The above transition can only be derived from Rule P1-11. From the premise of the rule, the following holds:

$$\langle x + y \rangle \xmapsto{t_1} \langle z \rangle \tag{H.225}$$

Transition H.225 can be derived from three rules. Rules P1-21, P1-22 and P1-23.

i. **Rules P1-21**
Then z in Transitions H.224 and H.225 is as follows:

$$z = z_1 + z_2$$

From the premise of the rule, the following holds:

$$\langle x \rangle \xmapsto{t_1} \langle z_1 \rangle \tag{H.226}$$

$$\langle y \rangle \xmapsto{t_1} \langle z_2 \rangle \tag{H.227}$$

Apply Rule P1-11 on the above transitions:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{t+t_1} \langle z_1 \rangle \tag{H.228}$$

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{t+t_1} \langle z_2 \rangle \tag{H.229}$$

Apply Rule P1-21 on the above transitions:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{t+t_1} \langle z_1 + z_2 \rangle \tag{H.230}$$

ii. **Rules P1-22**
If Transition H.225 is derived from this rule, then from the premise of the rule, the following holds:

$$\langle x \rangle \xmapsto{t_1} \langle z_1 \rangle \tag{H.231}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{H.232}$$

$$\langle y \rangle \not\xmapsto{t_1} \tag{H.233}$$

Apply Rule P1-11 on Transition H.231:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{t+t_1} \langle z_1 \rangle \tag{H.234}$$

From Predicate H.233, Rule P1-11 cannot be applied on $\sigma^t_{\mathsf{rel}}(y)$. Since, this is the only rule allowing a delay of length greater than t, hence the following predicate holds:

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{t+t_1} \tag{H.235}$$

From Rule P1-8, the following holds:

$$\langle \texttt{consistent}\ \sigma^{t}_{\mathsf{rel}}(y)\rangle \tag{H.236}$$

for $t > 0$.

Apply Rule P1-22 on transitions H.234, H.235 and H.236. We get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \xmapsto{t+t_1} \langle z_1\rangle \tag{H.237}$$

iii. **Rules P1-23**

Same reasoning as given for Rules P1-22 applies.

⊠

H.6 Axiom SRTD⊥

$$\sigma^{u+r}_{\mathsf{rel}}(x) + \sigma^{r}_{\mathsf{rel}}(\perp) = \sigma^{u+r}_{\mathsf{rel}}(x)$$

where $u \geq, r > 0$.

We need to prove, $\sigma^{u+r}_{\mathsf{rel}}(x) + \sigma^{r}_{\mathsf{rel}}(\perp) \leftrightarrows \sigma^{u+r}_{\mathsf{rel}}(x)$.

Let R be a binary relation on process terms defined as follows:

$$R = \quad \{ \quad (\sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp), \sigma^{u+s}_{\mathsf{rel}}(x)) \mid x, \in P, 0 < s \leq r\}$$

We prove that the relation $R \cup \mathcal{I}$ is a bisimulation relation. Below we prove the conditions that all pairs in R must satisfy in order for $R \cup \mathcal{I}$ to be a bisimulation relation

For all $a \in A$, $t \in R^{>}$,$p \in P$, the following holds:

1.
$$\langle \sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \xrightarrow{a} \langle p\rangle \implies \begin{array}{l}\exists p' \in P : \langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p'\rangle \\ \text{and } (p, p') \in R \cup \mathcal{I}\end{array}$$

Trivial. The left hand side of the implication does not hold.

2.
$$\langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p\rangle \implies \begin{array}{l}\exists p' \in P : \langle \sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \xrightarrow{a} \langle p'\rangle \\ \text{and } (p', p) \in R \cup \mathcal{I}\end{array}$$

Trivial. The left hand side of the implication does not hold.

3.
$$\langle \sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \overset{t}{\mapsto} \langle p\rangle \implies \begin{array}{l}\exists p' \in P : \langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \overset{t}{\mapsto} \langle p'\rangle \\ \text{and } (p, p') \in R \cup \mathcal{I}\end{array}$$

Suppose,

$$\langle \sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \overset{t}{\mapsto} \langle p\rangle \tag{H.238}$$

This can only be derived from three rules.

(a) Rule P1-21

Then $p = p_1 + p_2$.
Rewriting Transition H.238:

$$\langle \sigma_{\mathsf{rel}}^{u+s}(x) + \sigma_{\mathsf{rel}}^{s}(\perp) \rangle \xmapsto{t} \langle p_1 + p_2 \rangle \tag{H.239}$$

From the premise of the rule, the following must hold:

$$\langle \sigma_{\mathsf{rel}}^{u+s}(x) \rangle \xmapsto{t} \langle p_1 \rangle \tag{H.240}$$

$$\langle \sigma_{\mathsf{rel}}^{s}(\perp) \rangle \xmapsto{t} \langle p_2 \rangle \tag{H.241}$$

We distinguish between three cases for different values of t.

i. Case $t < s$

Let $s = t + t_1$, for some $0 < t_1 < s$.
Then Transitions H.240 and H.241 are derived from Rule P1-9. And,

$$p_1 = \sigma_{\mathsf{rel}}^{u+t_1}(x) \text{ and } p_2 = \sigma_{\mathsf{rel}}^{t_1}(\perp)$$

Rewriting Transition H.239:

$$\langle \sigma_{\mathsf{rel}}^{u+t+t_1}(x) + \sigma_{\mathsf{rel}}^{t+t_1}(\perp) \rangle \xmapsto{t} \langle \sigma_{\mathsf{rel}}^{u+t_1}(x) + \sigma_{\mathsf{rel}}^{t_1}(\perp) \rangle \tag{H.242}$$

From Rule P1-9, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{u+t+t_1}(x) \rangle \xmapsto{t} \langle \sigma_{\mathsf{rel}}^{u+t_1}(x) \rangle \tag{H.243}$$

Consider target process terms in Transition H.242 and H.243. For $t_1 > 0$, the pair $(\sigma_{\mathsf{rel}}^{u+t_1}(x) + \sigma_{\mathsf{rel}}^{t_1}(\perp), \sigma_{\mathsf{rel}}^{u+t_1}(x))$ is in R.

ii. Case $t = s$

Then H.241 must be derived from Rule P1-10 And,

$$p_2 = \perp$$

which is not possible.
Hence, Rule P1-21 cannot be used to derive Transition H.238 when $s = t$.

iii. Case $t > s$

Let $t = s + s_1$.
Then Transition H.241 can only be derived from Rule P1-11, which requires that $\perp$ can delay for s_1 time units. Again this is impossible. Hence, Rule P1-21 cannot be used to derive Transition H.238 when $t > s$.

(b) Rule P1-22

If Transition H.238 is derived from this rule, then the following must hold:

$$\langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \overset{t}{\longmapsto} \langle p\rangle \tag{H.244}$$

$$\langle \sigma^{s}_{\mathsf{rel}}(\bot)\rangle \overset{t}{\not\longmapsto} \tag{H.245}$$

$$\langle \texttt{consistent}\ \sigma^{s}_{\mathsf{rel}}(\bot)\rangle \tag{H.246}$$

We distinguish between three cases for different values of t.

i. Case $t < s$
Predicate H.245 cannot be derived for $t < s$. Because from Rule P1-9, a process term $\sigma^{s}_{\mathsf{rel}}(z)$ can always delay for a duration shorter than s.

ii. Cases $t = s$
Rewriting Transitions H.244 and H.245.

$$\langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \overset{s}{\longmapsto} \langle p\rangle \tag{H.247}$$

$$\langle \sigma^{s}_{\mathsf{rel}}(\bot)\rangle \overset{s}{\not\longmapsto} \tag{H.248}$$

The Transition H.247 proves that a transition corresponding to Transition H.238 holds for $\sigma^{u+s}_{\mathsf{rel}}(x)$.
The pair (p, p) is in $R \cup \mathcal{I}$

iii. Cases $t > s$

Reasoning is Similar to above case.

(c) Rule P1-23
Then the following must hold:

$$\langle \sigma^{s}_{\mathsf{rel}}(\bot)\rangle \overset{t}{\longmapsto} \langle p\rangle \tag{H.249}$$

$$\langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \overset{t}{\not\longmapsto} \tag{H.250}$$

$$\langle \texttt{consistent}\ \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \tag{H.251}$$

We distinguish between three cases for different values of t.

i. Case $t < s$

Let $s = t + t_1$.
For $s > t$, the Predicate H.250 cannot be derived.

ii. Case $s = t$
Transition H.249 cannot be derived.
If $u > 0$, then Predicate H.250 can also not be derived.

iii. Case $t > s$

Transition H.249 cannot be derived.

Hence, we conclude that Rule P1-23 cannot be used to derive Transition H.238.

4.

$$\langle \sigma^{u+s}_{\mathsf{rel}}(x)\rangle \overset{t}{\longmapsto} \langle p\rangle \implies \begin{array}{l}\exists p' \in P : \langle \sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\bot)\rangle \overset{t}{\longmapsto} \langle p'\rangle \\ \text{and } (p', p) \in R \cup \mathcal{I}\end{array}$$

Suppose,

$$\langle \sigma_{\mathsf{rel}}^{u+s}(x)\rangle \xmapsto{t} \langle p\rangle \tag{H.252}$$

We distinguish between three cases for different values of t.

(a) $\underline{\underline{\text{Case } t < s}}$

Let $s = t + t_1$, for $t_1 > 0$.
Then Transition H.252 must be derived from Rule P1-9 and $p = \sigma_{\mathsf{rel}}^{u+t_1}(x)$.
Rewriting Transition H.252:

$$\langle \sigma_{\mathsf{rel}}^{u+t+t_1}(x)\rangle \xmapsto{t} \langle \sigma_{\mathsf{rel}}^{u+t_1}(x)\rangle \tag{H.253}$$

By Rule P1-9, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t+t_1}(\perp)\rangle \xmapsto{t} \langle \sigma_{\mathsf{rel}}^{t_1}(\perp)\rangle \tag{H.254}$$

Apply Rule P1-21 on the above transitions:

$$\langle \sigma_{\mathsf{rel}}^{u+t+t_1}(x) + \sigma_{\mathsf{rel}}^{t+t_1}(\perp)\rangle \xmapsto{t} \langle \sigma_{\mathsf{rel}}^{u+t_1}(x) + \sigma_{\mathsf{rel}}^{t_1}(\perp)\rangle \tag{H.255}$$

Consider target process terms in Transition H.253 and H.255. For $t_1 > 0$, the pair $(\sigma_{\mathsf{rel}}^{u+t_1}(x) + \sigma_{\mathsf{rel}}^{t_1}(\perp), \sigma_{\mathsf{rel}}^{u+t_1}(x))$ is in R.

(b) $\underline{\underline{\text{Case } t = s}}$

Rewriting Transition H.252:

$$\langle \sigma_{\mathsf{rel}}^{u+s}(x)\rangle \xmapsto{s} \langle p\rangle \tag{H.256}$$

When $s = t$, the following predicate holds:

$$\langle \sigma_{\mathsf{rel}}^{s}(\perp)\rangle \not\xmapsto{s} \tag{H.257}$$

And also:

$$\langle \texttt{consistent}\ \ \sigma_{\mathsf{rel}}^{s}(\perp)\rangle \tag{H.258}$$

Apply Rule P1-22 on the above transitions:

$$\langle \sigma_{\mathsf{rel}}^{u+s}(x) + \sigma_{\mathsf{rel}}^{s}(\perp)\rangle \xmapsto{s} \langle p\rangle \tag{H.259}$$

Consider target process terms in Transition H.252 and H.259. The pair (p, p) is in $\mathcal{I}$.

(c) $\underline{\underline{\text{Case } t > s}}$

Let $t = s + s_1$, for some $s_1 > 0$
Rewriting Transition H.252:

$$\langle \sigma_{\mathsf{rel}}^{u+s}(x)\rangle \xmapsto{s+s_1} \langle p\rangle \tag{H.260}$$

Let us consider $\sigma^{s}_{\mathsf{rel}}(\perp)$. The Transition

$$\langle\sigma^{s}_{\mathsf{rel}}(\perp)\rangle \xmapsto{s+s_1} \langle p'\rangle$$

cannot be derived for any process term p'.
Hence, the following predicate holds:

$$\langle\sigma^{s}_{\mathsf{rel}}(\perp)\rangle \not\xmapsto{s+s_1} \tag{H.261}$$

Also we know:

$$\langle\texttt{consistent}\ \sigma^{s}_{\mathsf{rel}}(\perp)\rangle$$

Apply Rule P1-22:

$$\langle\sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \xmapsto{s+s_1} \langle p\rangle \tag{H.262}$$

Consider the target process terms in Transitions H.252 and H.262. The pair (p, p) is in $\mathcal{I}$.

5.

$$\langle\sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \xrightarrow{a} \surd \iff \langle\sigma^{u+s}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \surd$$

Trivial.

6.

$$\langle\texttt{consistent}\ \sigma^{u+s}_{\mathsf{rel}}(x) + \sigma^{s}_{\mathsf{rel}}(\perp)\rangle \iff \langle\texttt{consistent}\ \sigma^{u+s}_{\mathsf{rel}}(x)\rangle$$

Trivial.

H.7 Axiom SRT2

$$\sigma^{v}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)) = \sigma^{v+u}_{\mathsf{rel}}(x)$$

where $v, u \geq 0$.

We need to prove, $\sigma^{v}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)) \leftrightarrows \sigma^{v+u}_{\mathsf{rel}}(x)$.

We prove this axiom in four steps:

Case $v = 0, u = 0$

Proof Trivial using Axiom SRT1.

Case $v = 0, u > 0$

Proof Trivial using Axiom SRT1.

Case $v > 0, u = 0$

Proof Trivial using Axiom SRT1.

Case $v > 0, u > 0$

Let R be a binary relation on process terms defined as follows:

$$R = \quad \{ \quad (\sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)), \sigma^{t+u}_{\mathsf{rel}}(x)), \mid x \in P, 0 < t \leq v\}$$

Then the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

For all $a \in A, r > 0, x, y \in P$, the following holds:

1.
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \langle y\rangle \implies \begin{array}{l}\exists z' \in P : \langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle z'\rangle \\ \text{and } (p, z') \in R \cup \mathcal{I}.\end{array}$$

Suppose,
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \langle y\rangle$$

A process term with relative delay operator with duration greater than 0 cannot perform an action step. Hence our supposition doesn't hold.

2.
$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle y\rangle \implies \begin{array}{l}\exists z' \in P : \langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \langle z'\rangle \\ \text{and } (p, z') \in R \cup \mathcal{I}.\end{array}$$

Suppose,
$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle y\rangle$$

A process term with relative delay operator with duration greater than 0 cannot perform an action step. Hence our supposition doesn't hold.

3.
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{r}{\longmapsto} \langle y\rangle \implies \begin{array}{l}\exists z' \in P : \langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z'\rangle \\ \text{and } (p, z') \in R \cup \mathcal{I}.\end{array}$$

Suppose,
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{r}{\longmapsto} \langle y\rangle \tag{H.263}$$

We distinguish between three cases for different values of r.

(a) Case $r < t$

Let $t = r + r_1$ for some r_1 with $0 < r_1 < t$.
Then Transition H.263 is derived from Rule P1-9 and $y = \sigma^{r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))$. Rewriting Transition H.263:
$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \tag{H.264}$$
By Rule P1-9 the following can be derived:
$$\langle \sigma^{r+r_1+u}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1+u}_{\mathsf{rel}}(x)\rangle \tag{H.265}$$
Consider the target process terms in Transitions H.264 and H.265. The pair $(\sigma^{r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)), \sigma^{r_1+u}_{\mathsf{rel}}(x))$, where $0 < r_1 < t$ is in R.

(b) Case $r = t$

Then Transition H.263 is derived from Rule P1-10. Then $y = \sigma^{u}_{\mathsf{rel}}(x)$. Rewriting Transition H.263:
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{t}{\longmapsto} \langle \sigma^{u}_{\mathsf{rel}}(x)\rangle \tag{H.266}$$
By Rule P1-9 the following can be derived:
$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \overset{t}{\longmapsto} \langle \sigma^{u}_{\mathsf{rel}}(x)\rangle \tag{H.267}$$
Consider the target process terms in Transitions H.266 and H.267. The pair $(\sigma^{u}_{\mathsf{rel}}(x), \sigma^{u}_{\mathsf{rel}}(x))$ is in $\mathcal{I}$.

(c) Case $r > t$

Let $r = t + s$, for some $s > 0$. Rewriting Transition H.263,

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+s} \langle y \rangle \tag{H.268}$$

The above transition can only be derived from Rule P1-11. From the premise of the rule, the following holds:

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{s} \langle y \rangle \tag{H.269}$$

We distinguish between three cases depending on different values of the duration s of the time step.

i. Case $s < u$

Let $u = s + s_1$, for some s_1 with $0 < s_1 < s$.
Then Transition H.269 can only be derived from Rule P1-9. Then $y = \sigma_{\mathsf{rel}}^{s_1}(x)$.
Rewriting Transitions H.268 and H.269, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+s} \langle \sigma_{\mathsf{rel}}^{s_1}(x) \rangle \tag{H.270}$$

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{s} \langle \sigma_{\mathsf{rel}}^{s_1}(x) \rangle \tag{H.271}$$

From Rule P1-9, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{t+s} \langle \sigma_{\mathsf{rel}}^{s_1}(x) \rangle \tag{H.272}$$

Consider the target process terms in Transitions H.270 and H.272. The pair $(\sigma_{\mathsf{rel}}^{s_1}(x), \sigma_{\mathsf{rel}}^{s_1}(x))$ is in $\mathcal{I}$.

ii. Case $s = u$

Then Transition H.269 can only be derived from Rule P1-10. Then $y = x$. Rewriting Transitions H.268 and H.269, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+u} \langle x \rangle \tag{H.273}$$

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{u} \langle x \rangle \tag{H.274}$$

From the premise of Rule P1-10, the following must hold:

$$\langle \texttt{consistent}\ x \rangle$$

Applying Rule P1-10 on process term $\sigma_{\mathsf{rel}}^{t+u}(x)$, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{t+u} \langle x \rangle \tag{H.275}$$

Consider the target process terms in Transitions H.273 and H.275. The pair (x, x) is in $\mathcal{I}$.

iii. Case $s > u$

Let $s = u + t_1$, for some $t_1 > 0$. Rewriting Transitions H.268 and H.269, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+u+t_1} \langle y \rangle \tag{H.276}$$

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{u+t_1} \langle y \rangle \tag{H.277}$$

Transition H.277 can only be derived from Rule P1-11. Then from the premise of the rule the following must hold:

$$\langle x \rangle \overset{t_1}{\longmapsto} \langle y \rangle \tag{H.278}$$

Apply Rule P1-11 on the above transition. For any $m > 0$, the following is derivable:

$$\langle \sigma^m_{\mathsf{rel}}(x) \rangle \overset{m+t_1}{\longmapsto} \langle y \rangle$$

In the above transition, m can be $t + u$. Hence, we get:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x) \rangle \overset{t+u+t_1}{\longmapsto} \langle y \rangle \tag{H.279}$$

Consider the target process terms in Transition H.276 and Transition H.279. The pair (y, y) is in $\mathcal{I}$.

4.

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle y \rangle \implies \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \overset{r}{\longmapsto} \langle z' \rangle \text{ and } (p, z') \in R \cup \mathcal{I}.$$

Suppose,

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle y \rangle \tag{H.280}$$

We distinguish between three cases for different values of r.

(a) Case $r < (t + u)$

Again we distinguish between three cases:

i. *Case* $r < t$
Let $t = r + r_1$, for some r_1 such that, $0 < r_1 < t$.
Then Transition H.280 can only be derived from Rule P1-9. Then $y = \sigma^{r_1+u}_{\mathsf{rel}}(x)$. Rewriting Transition H.280, we get:

$$\langle \sigma^{r+r_1+u}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1+u}_{\mathsf{rel}}(x) \rangle \tag{H.281}$$

Then from Rule P1-9, the following can be derived:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \tag{H.282}$$

Consider the target process terms in Transitions H.281 and H.282. For $0 < r_1 < t$, the pair $(\sigma^{r_1}_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)), \sigma^{r_1+u}_{\mathsf{rel}}(x))$ is in R.

ii. *Case* $r = t$
Then Transition H.280 can only be derived from Rule P1-9. Then $y = \sigma^u_{\mathsf{rel}}(x)$. Rewriting Transition H.280, we get:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x) \rangle \overset{t}{\longmapsto} \langle \sigma^u_{\mathsf{rel}}(x) \rangle \tag{H.283}$$

From Rule P1-10, the following can be derived:

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \overset{t}{\longmapsto} \langle \sigma^u_{\mathsf{rel}}(x) \rangle \tag{H.284}$$

Consider the target process terms in Transitions H.283 and H.284. The pair $(\sigma^u_{\mathsf{rel}}(x), \sigma^u_{\mathsf{rel}}(x))$ is in $\mathcal{I}$.

iii. *Case $r > t$*

Let $r = t + s$ for some $s > 0$.

Note that $s < u$ because of our assumption that $r < (t + u)$. Let $u = s + s_1$ for some s_1 such that $0 < s_1 < u$.

Rewriting Transition H.280, we get:

$$\langle \sigma^{t+s+s_1}_{\mathsf{rel}}(x)\rangle \xmapsto{t+s} \langle \sigma^{s_1}_{\mathsf{rel}}(x)\rangle \tag{H.285}$$

By Rule P1-9, the following can be derived:

$$\langle \sigma^{s+s_1}_{\mathsf{rel}}(x)\rangle \xmapsto{s} \langle \sigma^{s_1}_{\mathsf{rel}}(x)\rangle \tag{H.286}$$

Apply Rule P1-11 on the above transition. We get:

$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{s+s_1}_{\mathsf{rel}}(x))\rangle \xmapsto{t+s} \langle \sigma^{s_1}_{\mathsf{rel}}(x)\rangle \tag{H.287}$$

Consider the target process terms in Transitions H.285 and H.287. The pair $(\sigma^{s_1}_{\mathsf{rel}}(x), \sigma^{s_1}_{\mathsf{rel}}(x))$ is in $\mathcal{I}$.

(b) Case $r = (t + u)$

Then Transition H.280 can only be derived from Rule P1-10 and $y = x$.

Rewriting Transition H.280, we get:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xmapsto{t+u} \langle x\rangle \tag{H.288}$$

From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ x\rangle$$

Apply Rule P1-10 on the above predicate, we get:

$$\langle \sigma^{u}_{\mathsf{rel}}(x)\rangle \xmapsto{u} \langle x\rangle \tag{H.289}$$

Apply Rule P1-11 on the above transition. We get:

$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xmapsto{t+u} \langle x\rangle \tag{H.290}$$

Consider the target process terms in Transitions H.288 and H.290. The pair (x, x) is in $\mathcal{I}$.

(c) Case $r > (t + u)$

Let $r = t + u + t_1$, for some $t_1 > 0$. Rewriting Transition H.280, we get:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xmapsto{t+u+t_1} \langle y\rangle \tag{H.291}$$

From the premise of the rule the following must hold:

$$\langle x\rangle \xmapsto{t_1} \langle y\rangle \tag{H.292}$$

Apply Rule P1-11 on the above transition. We get:

$$\langle \sigma^{u}_{\mathsf{rel}}(x)\rangle \xmapsto{u+t_1} \langle y\rangle \tag{H.293}$$

Again apply Rule P1-11 on the above transition. We get:

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x))\rangle \xmapsto{t+u+t_1} \langle y \rangle \tag{H.294}$$

Consider the target process terms in Transitions H.291 and H.294. The pair (y, y) is in $\mathcal{I}$.

5.

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \surd \iff \langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \surd$$

Trivial. Both process terms cannot perform an action.

6.

$$\langle \texttt{consistent}\ \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x))\rangle \iff \langle \texttt{consistent}\ \sigma^{t+u}_{\mathsf{rel}}(x)\rangle$$

Trivial. Both are consistent.

H.8 Axiom SRT4

$$\sigma^u_{\mathsf{rel}}(x) \cdot y = \sigma^u_{\mathsf{rel}}(x \cdot y)$$

where $u \geq 0$.

We need to prove, $\sigma^u_{\mathsf{rel}}(x) \cdot y \leftrightarrows \sigma^u_{\mathsf{rel}}(x \cdot y)$.

We do the proof in two steps.

Case $u = 0$

Proof Trivial using Axiom SRT1 and the fact that bisimulation is a congruence.

Case $u > 0$

Let R be a binary relation on process terms defined as follows:

$$R = \{ \ (\sigma^t_{\mathsf{rel}}(x) \cdot y, \sigma^t_{\mathsf{rel}}(x \cdot y)) \mid x, y \in P, 0 < t \leq u\}$$

For all $x, y, p \in P$, $r > 0$, $a \in A$, the following holds:

1.

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xrightarrow{a} \langle p \rangle \implies \exists z \in P : \langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \xrightarrow{a} \langle z \rangle \text{ and } (p, z) \in R \cup \mathcal{I}.$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xrightarrow{a} \langle p \rangle \tag{H.295}$$

The above action step can only be derived from Rule P1-13 or 14. We discuss the two cases one by one:

(a) Rule P1-13

If Transition H.295 is derived from this rule, then for some process term p', $p = p' \cdot y$. And from the premise of the rule, the following must be derivable,

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p' \rangle \tag{H.296}$$

An action step for operator σ^t_{rel} with $t > 0$ cannot be derived from any rules. Hence we conclude that Rule P1-13 cannot be used to derive Transition H.295.

(b) Rule P1-14

If Transition H.295 is derived from this rule, then, $p = y$. And from the premise of the rule, the following must be derivable,

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \surd \tag{H.297}$$

A termination step for operator σ^t_{rel} with $t > 0$ cannot be derived from any rules. Hence we conclude that Rule P1-14 cannot be used to derive Transition H.295.

Transition H.295 cannot be derived from any rules. Since the left hand side of the implication does not hold, therefore the implication holds.

2.

$$\begin{aligned} \langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \xrightarrow{a} \langle p \rangle \implies \quad & \exists z \in P : \langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \xrightarrow{a} \langle z \rangle \\ & \text{and } (z, p) \in R \cup \mathcal{I}. \end{aligned}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \xrightarrow{a} \langle p \rangle \tag{H.298}$$

An action step for operator σ^t_{rel} with $t > 0$ cannot be derived from any rules. Hence our supposition is wrong.

3.

$$\begin{aligned} \langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle p \rangle \implies \quad & \exists z \in P : \langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\longmapsto} \langle z \rangle \\ & \text{and } (p, z) \in R \cup \mathcal{I}. \end{aligned}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{H.299}$$

The above time step can only be derived from Rule P1-15. Then for some process term p', $p = p' \cdot y$. Rewriting Transition H.299:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle p' \cdot y \rangle \tag{H.300}$$

From the premise of the rule the following holds:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p' \rangle \tag{H.301}$$

We distinguish between three cases for different values of r:

(a) Case $r < t$

Let $t = r + r_1$, for some $r_1 > 0$.
Then Transition H.301 can only be derived from Rule P1-9. From the rule, we have $p' = \sigma^{r_1}_{\mathsf{rel}}(x)$. Rewriting Transitions H.300 and H.301:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \cdot y \rangle \tag{H.302}$$

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \rangle \tag{H.303}$$

From Rule P1-9, the following can be

$$\langle \sigma_{\mathsf{rel}}^{r+r_1}(x \cdot y)\rangle \overset{r}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{r_1}(x \cdot y)\rangle \tag{H.304}$$

Consider the target process terms in Transitions H.302 and H.304. For $0 < r_1 < t$, the pair $(\sigma_{\mathsf{rel}}^{r_1}(x) \cdot y, \sigma_{\mathsf{rel}}^{r_1}(x \cdot y))$ is in R.

(b) Case $r = t$

Then Transition H.301 can only be derived from Rule P1-10. From the rule, we have $p' = x$. Rewriting Transitions H.300 and H.301:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \cdot y\rangle \overset{t}{\longmapsto} \langle x \cdot y\rangle \tag{H.305}$$

$$\langle \sigma_{\mathsf{rel}}^{t}(x)\rangle \overset{t}{\longmapsto} \langle x\rangle \tag{H.306}$$

From the premise of Rule P1-10, the following must hold:

$$\langle \texttt{consistent}\ \ x\rangle$$

Apply Rule P1-12 on the above predicate, we get:

$$\langle \texttt{consistent}\ \ x \cdot y\rangle \tag{H.307}$$

Apply Rule P1-10 on process term $\sigma_{\mathsf{rel}}^{t}(x \cdot y)$, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x \cdot y)\rangle \overset{t}{\longmapsto} \langle x \cdot y\rangle \tag{H.308}$$

Consider the target process terms in Transitions H.305 and H.308. The pair $(x \cdot y, x \cdot y)$ is in $\mathcal{I}$.

(c) Case $r > t$

Let $r = t + v$, for some $v > 0$.

Rewriting Transitions H.300 and H.301:

$$\langle \sigma_{\mathsf{rel}}^{t}(x) \cdot y\rangle \xrightarrow{t+v} \langle p' \cdot y\rangle \tag{H.309}$$

$$\langle \sigma_{\mathsf{rel}}^{t}(x)\rangle \xrightarrow{t+v} \langle p'\rangle \tag{H.310}$$

Transition H.310 can only be derived from Rule P1-11. Then from the premise of the rule the following holds:

$$\langle x\rangle \overset{v}{\longmapsto} \langle p'\rangle \tag{H.311}$$

Apply Rule P1-15 on the above transition, we get:

$$\langle x \cdot y\rangle \overset{v}{\longmapsto} \langle p' \cdot y\rangle \tag{H.312}$$

Apply Rule P1-11 on the above transition, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(x \cdot y)\rangle \xrightarrow{t+v} \langle p' \cdot y\rangle \tag{H.313}$$

Consider the target process terms in Transitions H.309 and H.313. The pair $(p' \cdot y, p' \cdot y)$ is in $\mathcal{I}$.

4.
$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \overset{r}{\longmapsto} \langle p\rangle \implies \begin{array}{l}\exists z \in P : \langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \overset{r}{\longmapsto} \langle z\rangle \\ \text{and } (z,p) \in R \cup \mathcal{I}.\end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{H.314}$$

We distinguish between three cases for different values of r.

(a) Case $r < t$

Let $t = r + r_1$, for some $r_1 < t$.
Then Transition H.314 can only be derived from Rule P1-9. From the rule, we have $p = \sigma^{r_1}_{\mathsf{rel}}(x \cdot y)$. Rewriting Transition H.314:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x \cdot y)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x \cdot y)\rangle \tag{H.315}$$

From Rule P1-9, the following can be derived:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x)\rangle$$

Apply Rule P1-15 on the above transition. We get:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \cdot y\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \cdot y\rangle \tag{H.316}$$

Consider the target process terms in Transitions H.315 and H.316. The pair $(\sigma^{r_1}_{\mathsf{rel}}(x) \cdot y, \sigma^{r_1}_{\mathsf{rel}}(x \cdot y))$ is in R.

(b) Case $r = t$
Then Transition H.314 can only be derived from Rule P1-10. From the rule, we have $p = x \cdot y$. Rewriting Transition H.314:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \overset{t}{\longmapsto} \langle x \cdot y\rangle \tag{H.317}$$

The above time step can only be derived from Rule P1-10. From the premise of the rule,

$$\langle \texttt{consistent } x \cdot y\rangle$$

which can only be derived from Rule P1-12. Then the following must hold:

$$\langle \texttt{consistent } x\rangle$$

Apply Rule P1-10 on the above predicate, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{t}{\longmapsto} \langle x\rangle \tag{H.318}$$

Apply Rule P1-15, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \overset{t}{\longmapsto} \langle x \cdot y\rangle \tag{H.319}$$

Consider the target process terms in Transitions H.308 and H.319. The pair $(x \cdot y, x \cdot y)$ is in R.

(c) <u>Case $r > t$</u>
Let $r = v + t$, for some $v > 0$.

Then Transition H.314 can only be derived from Rule P1-11. Rewriting Transition H.314:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \xmapsto{t+v} \langle p \rangle \tag{H.320}$$

From the premise of the rule,

$$\langle x \cdot y \rangle \xmapsto{v} \langle p \rangle \tag{H.321}$$

The above transition can only be derived from Rule P1-15. Then for some process term p', $p = p' \cdot y$. Rewriting Transitions H.320 and H.321, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \xmapsto{t+v} \langle p' \cdot y \rangle \tag{H.322}$$
$$\langle x \cdot y \rangle \xmapsto{v} \langle p' \cdot y \rangle \tag{H.323}$$

From the the premise of the rule,

$$\langle x \rangle \xmapsto{v} \langle p' \rangle \tag{H.324}$$

Apply Rule P1-11 on the above transition, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{t+v} \langle p' \rangle \tag{H.325}$$

Apply Rule P1-15 on the above transition, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \xmapsto{t+v} \langle p' \cdot y \rangle \tag{H.326}$$

Consider the target process terms in Transitions H.320 and H.326. The pair $(p' \cdot y, p' \cdot y)$ is in R.

5.
$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \xrightarrow{a} \surd \iff \langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \xrightarrow{a} \surd$$

6.
$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \iff \langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle$$

From Rule P1-8,

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle$$

From Rule P1-12 and Rule P1-8, it can be derived that:

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle$$

H.9 Bisimulation Relations for other Axioms

1. $x + y = y + x$ **(A1)**

   We need to prove, $x + y \underline{\leftrightarrow} y + x$

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + y, y + x) \mid x, y \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

   The proof is trivial and therefore left.

2. $x + x = x$ (Idempotency-**A3**)

   We need to prove, $x + x \underline{\leftrightarrow} x$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + x, x) \mid x \in P\}$$

   Then using Theorems 12 and 13, it is easy to prove that the relation $R \cup \mathcal{I}$ satisfies all conditions of a bisimulation relation .

3. $(x + y) \cdot z = x \cdot z + y \cdot z$ (Right Distributivity-**A4**).

   We need to prove, $(x + y) \cdot z \underline{\leftrightarrow} x \cdot z + y \cdot z$.

   Let R be a binary relation on process terms defined as follows:

   $$R \;=\; \{ \quad ((x + y) \cdot z, x \cdot z + y \cdot z) \mid x, y, z \in P\}$$

   Then the relation $R \cup \mathcal{I}$ is a bisimulation relation.

4. $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ (Associativity of sequential composition-**A5**).

   We need to prove, $(x \cdot y) \cdot z \underline{\leftrightarrow} x \cdot (y \cdot z)$.

   Let R be a binary relation on process terms defined as follows:

   $$R \;=\; \{ \quad ((x \cdot y) \cdot z, x \cdot (y \cdot z)) \mid x, y, z \in P\}$$

   Then the relation $R \cup \mathcal{I}$ is a bisimulation relation.

5. $x + \tilde{\tilde{\delta}} = x$ **(A6SR)**

   We need to prove, $x + \tilde{\tilde{\delta}} \underline{\leftrightarrow} x$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + \tilde{\tilde{\delta}}, x) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

6. $\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$ **A7SR**

   We need to prove, $\tilde{\tilde{\delta}} \cdot x \,\underline{\leftrightarrow}\, \tilde{\tilde{\delta}}$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(\tilde{\tilde{\delta}} \cdot x, \tilde{\tilde{\delta}}) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

7. $x + \bot = \bot$ **(NE1)**

   We need to prove, $x + \bot \,\underline{\leftrightarrow}\, \bot$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + \bot, \bot) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

8. $\bot \cdot x = \bot$ **(NE2)**

   We need to prove, $\bot \cdot x \,\underline{\leftrightarrow}\, \bot$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(\bot \cdot x, \bot) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

9. $\tilde{\tilde{a}} \cdot \bot = \tilde{\tilde{\delta}}$ **(NE3)**

   We need to prove, $\tilde{\tilde{a}} \cdot \bot \,\underline{\leftrightarrow}\, \tilde{\tilde{\delta}}$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(\tilde{\tilde{a}} \cdot \bot, \tilde{\tilde{\delta}}), (\tilde{\tilde{\delta}}, \tilde{\tilde{a}} \cdot \bot) \mid a \in A\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

10. $\sigma^0_{\mathsf{rel}}(x) = x$ **(SRT1)**

    We need to prove, $\sigma^0_{\mathsf{rel}}(x) \,\underline{\leftrightarrow}\, x$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\sigma^0_{\mathsf{rel}}(x), x) \mid x \in P\}$$

    Then $R \cup \mathcal{I}$ is a bisimulation relation that witnesses $\sigma^0_{\mathsf{rel}}(x) \,\underline{\leftrightarrow}\, x$.

11. $\nu_{\mathsf{rel}}(\tilde{\tilde{a}}) = \tilde{\tilde{a}}$ (**SRU1**)

    We need to prove, $\nu_{\mathsf{rel}}(\tilde{\tilde{a}}) \,\underline{\leftrightarrow}\, \tilde{\tilde{a}}$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\nu_{\mathsf{rel}}(\tilde{\tilde{a}}), \tilde{\tilde{a}}) \mid a \in A\}$$

    The relation $R \cup \mathcal{I}$ is a bisimulation relation.

12. $\nu_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) = \tilde{\tilde{\delta}}$ **SRU2**

    We need to prove, $\nu_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \leftrightarroweq \tilde{\tilde{\delta}}$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\nu_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)), \tilde{\tilde{\delta}}) \mid x \in P, u > 0\}$$

    Then the relation $R \cup \mathcal{I}$ is a bisimulation relation.

13. $\nu_{\mathsf{rel}}(x + y) = \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)$ (**SRU3**)

    We need to prove, $\nu_{\mathsf{rel}}(x + y) \leftrightarroweq \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\nu_{\mathsf{rel}}(x + y), \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)) \mid x, y \in P\}$$

    Then the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

14. $\nu_{\mathsf{rel}}(x \cdot y) = \nu_{\mathsf{rel}}(x) \cdot y$ (**SRU4**)

    We need to prove, $\nu_{\mathsf{rel}}(x \cdot y) \leftrightarroweq \nu_{\mathsf{rel}}(x) \cdot y$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\nu_{\mathsf{rel}}(x \cdot y), \nu_{\mathsf{rel}}(x) \cdot y) \mid x, y \in P\}$$

    Then the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

15. $\nu_{\mathsf{rel}}(\perp) = \perp$ (**NESRU**)

    We need to prove, $\nu_{\mathsf{rel}}(\perp) \leftrightarroweq \perp$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\nu_{\mathsf{rel}}(\perp), \perp)\}$$

    The proof is trivial and therefore left.

Appendix I

Soundness Proofs for Proposal 2

Let $\mathcal{I}$ be a binary relation on process terms defined as follows:

$$\mathcal{I} = \{(x, x) \mid x \in P\}$$

It is obvious that $\mathcal{I}$ is a bisimulation relation. We will use the relation $\mathcal{I}$ frequently in the proofs. We prove that the axioms given in Table 3.14 hold in the semantics given in Section 3.6.4. Axiom A1 is proved by showing that the transition system satisfies the commutativity format (for the choice operator) given in [Mou05]. The rest of the axioms are proven in the traditional way by proving that left and right hand sides of all axioms are bisimilar. We give complete proofs for some of the axioms including Axioms A2 and axioms for the relative delay operator. For the rest of the proofs, we only give a bisimulation relation. A reader interested in the complete proofs can refer to [KC08] for complete proofs.

The proofs of the soundness theorem use the following theorems.

I.1 Theorem : Sources of Transitions are consistent

Theorem 14 *For all closed terms p the following hold:*
For all $p', p'' \in P$, $a, b \in A$, $r, s > 0$:

$$(\langle p \rangle \xrightarrow{a} \langle p' \rangle) \vee (\langle p \rangle \overset{r}{\longmapsto} \langle p'' \rangle) \vee (\langle p \rangle \xrightarrow{b} \surd) \vee (\langle p \rangle \overset{s}{\longmapsto}_{\perp})$$
$$\Longrightarrow \langle \texttt{\textit{consistent}}\ p \rangle$$

<u>**Proof**</u> We prove the above theorem by structural induction on a process term $p \in P$. The base case of the structural induction comprises of constant process terms, i.e. all undelayable actions in $\mathcal{A}$, the deadlock process term δ and the inconsistent process $\perp$.
<u>Base Case</u>

1. $p = \tilde{\tilde{a}}$.

   From Rule P2-2, $\langle \texttt{consistent}\ \tilde{\tilde{a}} \rangle$. Hence all conditions of the theorem are trivially satisfied.

2. $p = \tilde{\tilde{\tilde{\delta}}}$

   From Rule P2-1, $\langle \texttt{consistent}\ \tilde{\tilde{\tilde{\tilde{\delta}}}} \rangle$. Hence all conditions of the theorem are trivially satisfied.

3. $p = \bot$

   There are no rules for an inconsistent process $\bot$ in the semantics of $BPA_{\bot}^{srt}$. Hence all conditions of the theorem are trivially satisfied (as the left hand sides of the implications do not hold.)

By Induction Hypothesis

1. $p = \sigma^0_{\mathsf{rel}}(x)$, for a closed term x. We show that if p can perform an action or a time step or a termination or a future inconsistency predicate holds for p, then $\langle \texttt{consistent}\ p \rangle$ holds.

   (a) *Action Step:*
   Suppose,

   $$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \langle p' \rangle$$

   It can only be derived from Rule P2-4. From the premise of the rule,

   $$\langle x \rangle \xrightarrow{a} \langle p' \rangle$$

   By Induction on the above action step, we get:

   $$\langle \texttt{consistent}\ x \rangle$$

   Apply Rule P2-7. We get:

   $$\langle \texttt{consistent}\ \sigma^0_{\mathsf{rel}}(x) \rangle$$

   Hence proved.

   (b) *Time Step:*
   Suppose,

   $$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

   It can only be derived from Rule P2-6. From the premise of the rule,

   $$\langle x \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

   By Induction on the above action step, we get:

   $$\langle \texttt{consistent}\ x \rangle$$

   Apply Rule P2-7. We get:

   $$\langle \texttt{consistent}\ \sigma^0_{\mathsf{rel}}(x) \rangle$$

   Hence proved.

(c) *Termination Predicate:*
Suppose,

$$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \surd$$

It can only be derived from Rule P2-5. From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \surd$$

By Induction on the above action step, we get:

$$\langle \texttt{consistent}\ x \rangle$$

Apply Rule P2-7. We get:

$$\langle \texttt{consistent}\ \sigma^0_{\mathsf{rel}}(x) \rangle$$

Hence proved.

(d) *A Future Inconsistency Predicate:*
Suppose,

$$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto}_{\perp}$$

It can only be derived from Rule P2-8. From the premise of the rule,

$$\langle x \rangle \overset{r}{\longmapsto}_{\perp}$$

By Induction on the above action step, we get:

$$\langle \texttt{consistent}\ x \rangle$$

Apply Rule P2-7. We get:

$$\langle \texttt{consistent}\ \sigma^0_{\mathsf{rel}}(x) \rangle$$

Hence proved.

2. $p = \sigma^t_{\mathsf{rel}}(x) \qquad t > 0$
From Rule P2-12, for a process term $\sigma^t_{\mathsf{rel}}(x)$, with $t > 0$, the following holds:

$$\langle \texttt{consistent}\ \sigma^t_{\mathsf{rel}}(x) \rangle$$

Hence all conditions of the theorem are trivially proved.

3. $p = x \cdot y$.
We prove the four conditions of the theorem one by one.

(a) *Action Step:*
Suppose,

$$\langle x \cdot y \rangle \xrightarrow{a} \langle p' \rangle$$

It can only be derived from Rule P2-15 or Rule P2-16.

- *Rule P2-15*
  Then for some process term p'', $p' = p'' \cdot y$. From the premise of the rule,

  $$\langle x \rangle \xrightarrow{a} \langle p'' \rangle$$

  By Induction on the above action step, we get:

  $$\langle \texttt{consistent}\ x \rangle$$

  Apply Rule P2-18. We get:

  $$\langle \texttt{consistent}\ x \cdot y \rangle$$

  Hence proved.
- *Rule P2-16*
  Then, $p' = y$. From the premise of the rule,

  $$\langle x \rangle \xrightarrow{a} \surd$$

  By Induction on the above predicate, we get:

  $$\langle \texttt{consistent}\ x \rangle$$

  Apply Rule P2-18. We get:

  $$\langle \texttt{consistent}\ x \cdot y \rangle$$

  Hence proved.

(b) *Time Step:*
Suppose,

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

It can only be derived from Rule P2-17. From the premise of the rule,

$$\langle x \rangle \overset{r}{\longmapsto} \langle p' \rangle$$

By Induction on the above time step, we get:

$$\langle \texttt{consistent}\ x \rangle$$

Apply Rule P2-18. We get:

$$\langle \texttt{consistent}\ x \cdot y \rangle$$

Hence proved.

(c) *Termination Predicate:*
Suppose,

$$\langle x \cdot y \rangle \xrightarrow{a} \surd$$

There are no rules to derive a termination predicate for a sequential composition. Hence the left hand side of the implication does not hold and the implication is trivially satisfied.

(d) *A Future Inconsistency Predicate:*
Suppose,

$$\langle x \cdot y \rangle \overset{r}{\longmapsto}_{\perp}$$

It can only be derived from Rule P2-19. From the premise of the rule,

$$\langle x \rangle \overset{r}{\longmapsto}_{\perp}$$

By Induction on the above predicate, we get:

$$\langle \texttt{consistent}\ \ x \rangle$$

Apply Rule P2-18. We get:

$$\langle \texttt{consistent}\ \ x \cdot y \rangle$$

Hence proved.

4. $p = x + y$.

We prove the four conditions of the theorem one by one.

(a) *Action Step:*
Suppose,

$$\langle x + y \rangle \xrightarrow{a} \langle p' \rangle$$

It can only be derived from Rule P2-20 or Rule P2-21.

- *Rule P2-20*
  From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \langle p' \rangle \tag{I.1}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{I.2}$$

By Induction on Transition I.1, we get:

$$\langle \texttt{consistent}\ \ x \rangle \tag{I.3}$$

Apply Rule P2-27 on Predicates I.2 and I.3. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

Hence proved.

- *Rule P2-21*
  From the premise of the rule,

$$\langle y \rangle \xrightarrow{a} \langle p' \rangle \tag{I.4}$$

$$\langle \texttt{consistent}\ \ x \rangle \tag{I.5}$$

By Induction on Transition I.4, we get:

$$\langle \texttt{consistent}\ \ y \rangle \tag{I.6}$$

Apply Rule P2-27 on Predicates I.5 and I.6. We get:

$$\langle \texttt{consistent}\ \ x + y \rangle$$

Hence proved.

(b) *Time Step:*

Suppose,

$$\langle x + y\rangle \xmapsto{r} \langle p'\rangle$$

It can only be derived from Rule P2-24 or Rule P2-25 or Rule P2-26.

- *Rule P2-24*
  Then for some process terms x_1, y_1, $p' = x_1 + y_1$. From the premise of the rule the following holds:

$$\langle x\rangle \xmapsto{r} \langle x_1\rangle \tag{I.7}$$

$$\langle y\rangle \xmapsto{r} \langle y_1\rangle \tag{I.8}$$

  By Induction on the above time steps, we get:

$$\langle \texttt{consistent}\ \ x\rangle$$

$$\langle \texttt{consistent}\ \ y\rangle$$

  Apply Rule P2-27 on the above Predicates. We get:

$$\langle \texttt{consistent}\ \ x + y\rangle$$

  Hence proved.

- *Rule P2-25*
  From the premise of the rule the following holds:

$$\langle x\rangle \xmapsto{r} \langle p'\rangle \tag{I.9}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.10}$$

$$\langle y\rangle \overset{r}{\not\mapsto} \tag{I.11}$$

$$\forall s \leq r,\ \ \langle y\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.12}$$

  By Induction on time step I.9, we get:

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.13}$$

  Apply Rule P2-27 on Predicates I.13 and I.10. We get:

$$\langle \texttt{consistent}\ \ x + y\rangle$$

  Hence proved.

- *Rule P2-26*
  From the premise of the rule the following holds:

$$\langle y\rangle \xmapsto{r} \langle p'\rangle \tag{I.14}$$

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.15}$$

$$\langle x\rangle \overset{r}{\not\mapsto} \tag{I.16}$$

$$\forall s \leq r,\ \ \langle x\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.17}$$

By Induction on time step I.14, we get:

$$\langle \texttt{consistent}\ y \rangle \tag{I.18}$$

Apply Rule P2-27 on Predicates I.18 and I.15. We get:

$$\langle \texttt{consistent}\ x + y \rangle$$

Hence proved.

(c) *Termination Predicate:*
Suppose,

$$\langle x + y \rangle \xrightarrow{a} \surd$$

It can only be derived from Rule P2-22 or Rule P2-23.

- *Rule P2-22*
  From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \surd \tag{I.19}$$

$$\langle \texttt{consistent}\ y \rangle \tag{I.20}$$

By Induction on Predicate I.19, we get:

$$\langle \texttt{consistent}\ x \rangle \tag{I.21}$$

Apply Rule P2-27 on Predicates I.20 and I.21. We get:

$$\langle \texttt{consistent}\ x + y \rangle$$

Hence proved.

- *Rule P2-23*
  From the premise of the rule,

$$\langle y \rangle \xrightarrow{a} \surd \tag{I.22}$$

$$\langle \texttt{consistent}\ x \rangle \tag{I.23}$$

By Induction on Predicate I.22, we get:

$$\langle \texttt{consistent}\ y \rangle \tag{I.24}$$

Apply Rule P2-27 on Predicates I.23 and I.24. We get:

$$\langle \texttt{consistent}\ x + y \rangle$$

Hence proved.

(d) *A Future Inconsistency Predicate:*
Suppose,

$$\langle x + y \rangle \overset{r}{\longmapsto}_{\perp}$$

It can only be derived from Rule P2-28 or Rule P2-29.

- *Rule P2-28*
  From the premise of the rule the following holds:

$$\langle x\rangle \overset{r}{\longmapsto}_{\bot} \tag{I.25}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.26}$$

$$\forall s < r,\ \ \langle y\rangle \overset{s}{\not\longmapsto}_{\bot} \tag{I.27}$$

  By Induction on predicate I.25, we get:

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.28}$$

  Apply Rule P2-27 on Predicates I.28 and I.26. We get:

$$\langle \texttt{consistent}\ \ x+y\rangle$$

  Hence proved.
- *Rule P2-29*
  From the premise of the rule the following holds:

$$\langle y\rangle \overset{r}{\longmapsto}_{\bot} \tag{I.29}$$

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.30}$$

$$\forall s < r,\ \ \langle x\rangle \overset{s}{\not\longmapsto}_{\bot} \tag{I.31}$$

  By Induction on predicate I.29, we get:

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.32}$$

  Apply Rule P2-27 on Predicates I.32 and I.30. We get:

$$\langle \texttt{consistent}\ \ x+y\rangle$$

  Hence proved.

5. $p = \nu_{\mathsf{rel}}(x)$

   (a) *Action Step:*
   Suppose,

$$\langle \nu_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p'\rangle$$

   It can only be derived from Rule P2-30. From the premise of the rule,

$$\langle x\rangle \xrightarrow{a} \langle p'\rangle$$

   By Induction on the above action step, we get:

$$\langle \texttt{consistent}\ \ x\rangle$$

   Apply Rule P2-32. We get:

$$\langle \texttt{consistent}\ \ \nu_{\mathsf{rel}}(x)\rangle$$

   Hence proved.

(b) *Time Step:*
No rule allows a derivation of a time step for the now operator.

(c) *Termination Predicate:*
Suppose,

$$\langle \nu_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \surd$$

It can only be derived from Rule P2-30. From the premise of the rule,

$$\langle x \rangle \xrightarrow{a} \surd$$

By Induction on the above predicate, we get:

$$\langle \texttt{consistent}\ x \rangle$$

Apply Rule P2-32. We get:

$$\langle \texttt{consistent}\ \nu_{\mathsf{rel}}(x) \rangle$$

Hence proved.

(d) *A future Inconsistency predicate:*
No rule allows a derivation of a future Inconsistency predicate for the now operator.

$\boxtimes$

I.2 Theorem : Future Inconsistency Predicates have Shortest Length

Theorem 15 *For all closed terms p, durations $r > 0$ the following holds:*

$$\langle p \rangle \overset{r}{\longmapsto}_{\perp} \Longrightarrow \forall s < r,\ \langle p \rangle \overset{s}{\not\longmapsto}_{\perp}$$

<u>Proof</u> We prove the above theorem by structural induction on a process term $p \in P$. The base case of the structural induction comprises of constant process terms, i.e. all undelayable actions in $\mathcal{A}$, the deadlock process term δ and the inconsistent process $\perp$.
<u>Base Case</u>

1. $p = \tilde{\tilde{a}}$.

   There are no rules to derive a future Inconsistency predicate for an undelayable action. As the left hand side of the implication does not hold, therefore, the implication holds.

2. $p = \tilde{\tilde{\delta}}$

   There are no rules to derive a future Inconsistency predicate for an undelayable action.

3. $p = \perp$

   There are no rules for an inconsistent process $\perp$ in the semantics of $BPA_{\perp}^{srt}$.

By Induction Hypothesis

1. $p = \sigma^{0}_{\mathsf{rel}}(x)$, for a closed term x.

   Suppose,

   $$\langle \sigma^{0}_{\mathsf{rel}}(x) \rangle \xmapsto{r}_{\perp}$$

   It can only be derived from Rule P2-8. From the premise of the rule,

   $$\langle x \rangle \xmapsto{r}_{\perp}$$

   By Induction on the above predicate, we get:

   $$\forall s < r, \;\; \langle x \rangle \not\xmapsto{s}_{\perp}$$

   Then for all durations $s < r$, the premise of Rule P2-8 is not satisfied. Since this is the only rule allowing a future Inconsistency predicate for the operator $\sigma^{0}_{\mathsf{rel}}$, therefore we conclude:

   $$\forall s < r, \;\; \langle \sigma^{0}_{\mathsf{rel}}(x) \rangle \not\xmapsto{s}_{\perp}$$

   Proved.

2. $p = \sigma^{t}_{\mathsf{rel}}(x) \qquad t > 0$

   Suppose,

   $$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{r}_{\perp} \tag{I.33}$$

   We distinguish between three cases depending on the duration r.

   (a) Case $r < t$

   For a process term $\sigma^{t}_{\mathsf{rel}}(x)$, no future inconsistency predicate of length less than t can be derived.

   (b) Case $r = t$

   Rewriting Predicate I.33:

   $$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{t}_{\perp}$$

   From the case for $r < t$, the following always holds:

   $$\forall s < t, \;\; \langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \not\xmapsto{s}_{\perp} \tag{I.34}$$

   As Predicate I.34 always holds, hence the theorem is proved for $\sigma^{t}_{\mathsf{rel}}(x)$.

   (c) Case $r > t$

   Let $r = u + t$, for $u > 0$.

   Than a future Inconsistency predicate can only be derived from Rule P2-14. From the premise of the rule,

   $$\langle x \rangle \xmapsto{u}_{\perp}$$

By Induction on the above Predicate, we have:

$$\forall s < u, \;\; \langle x \rangle \overset{s}{\not\mapsto}_{\bot}$$

Then Rule P2-14 cannot be applied for deriving a future Inconsistency predicate for $\sigma^{t}_{\mathsf{rel}}(x)$ with length $t + s$. As Rule P2-14 is the only such rule, therefore, we conclude that:

$$\forall s < u, \;\; \langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \overset{t+s}{\not\longmapsto}_{\bot}$$

We can rewrite the above predicate as:

$$\forall s < (t + u), \;\; \langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \overset{s}{\not\mapsto}_{\bot}$$

Hence proved.

3. $p = x \cdot y$.

   Suppose,

   $$\langle x \cdot y \rangle \overset{r}{\mapsto}_{\bot}$$

   It can only be derived from Rule P2-19. From the premise of the rule,

   $$\langle x \rangle \overset{r}{\mapsto}_{\bot}$$

   By Induction on the above predicate, we get:

   $$\forall s < r, \;\; \langle x \rangle \overset{s}{\not\mapsto}_{\bot}$$

   Then Rule P2-19 cannot be applied for deriving a future Inconsistency predicate for $x \cdot y$ with length less than r. As Rule P2-19 is the only such rule, therefore, we conclude that:

   $$\forall s < r, \;\; \langle x \cdot y \rangle \overset{s}{\not\mapsto}_{\bot}$$

   Hence proved.

4. $p = x + y$.

   Suppose,

   $$\langle x + y \rangle \overset{r}{\mapsto}_{\bot} \tag{I.35}$$

   It can only be derived from Rule P2-28 or Rule P2-29.

   - *Rule P2-28* If Predicate I.35 is derived from this rule, then from the premise of the rule the following holds:

     $$\langle x \rangle \overset{r}{\mapsto}_{\bot} \tag{I.36}$$
     $$\langle \texttt{consistent} \;\; y \rangle \tag{I.37}$$
     $$\forall s < r, \;\; \langle y \rangle \overset{s}{\not\mapsto}_{\bot} \tag{I.38}$$

By Induction on Predicate I.36, we get:

$$\forall s < r, \;\; \langle x \rangle \overset{s}{\not\mapsto}_{\bot} \tag{I.39}$$

From Predicate I.38, Rule P2-29 cannot be applied to derive a future Inconsistency Predicate for $x + y$ for a duration less than r.

Similarly, from Predicate I.39, Rule P2-28 cannot be applied to derive a Future Inconsistency Predicate for $x + y$ for a duration less than r. Since rules 28 and 29 are the only such rules, hence we conclude:

$$\forall s < r, \;\; \langle x + y \rangle \overset{s}{\not\mapsto}_{\bot}$$

Hence proved.

- *Rule P2-29* If Predicate I.35 is derived from this rule, then from the premise of the rule the following holds:

$$\langle y \rangle \overset{r}{\mapsto}_{\bot} \tag{I.40}$$

$$\langle \texttt{consistent}\ x \rangle \tag{I.41}$$

$$\forall s < r, \;\; \langle x \rangle \overset{s}{\not\mapsto}_{\bot} \tag{I.42}$$

By Induction on Predicate I.40, we get:

$$\forall s < r, \;\; \langle y \rangle \overset{s}{\not\mapsto}_{\bot} \tag{I.43}$$

From Predicate I.42, Rule P2-28 cannot be applied to derive a future Inconsistency Predicate for $x + y$ for a duration less than r.

Similarly, from Predicate I.43, Rule P2-29 cannot be applied to derive a Future Inconsistency Predicate for $x + y$ for a duration less than r. Since rules 28 and 29 are the only such rules, hence we conclude:

$$\forall s < r, \;\; \langle x + y \rangle \overset{s}{\not\mapsto}_{\bot}$$

Hence proved.

5. $p = \nu_{\mathsf{rel}}(x)$

   There are no rules to derive a Future Inconsistency Predicate for the now operator. Hence the theorem trivially holds.

⊠

I.3 Theorem : Time Determinism

Theorem 16 *For all closed terms p, durations $r > 0$ the following holds:*

$$\langle p \rangle \overset{r}{\mapsto} \langle p_1 \rangle \wedge \langle p \rangle \overset{r}{\mapsto} \langle p_2 \rangle$$
$$\implies p_1 \equiv p_2$$

Proof We prove the above theorem by structural induction on a process term $p \in P$. The base case of the structural induction comprises of constant process terms, i.e. all undelayable actions in $\mathcal{A}$, the deadlock process term δ and the inconsistent process $\bot$.

Base Case

1. $p = \tilde{\tilde{a}}$.

   There are no rules to derive a time step for an undelayable action.

2. $p = \tilde{\tilde{\delta}}$

   There are no rules to derive a future Inconsistency predicate for the deadlock constant.

3. $p = \bot$

   There are no rules for an inconsistent process $\bot$.

By Induction Hypothesis

1. $p = \sigma^0_{\mathsf{rel}}(x)$, for a closed term x.

   Suppose,

$$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{I.44}$$
$$\langle \sigma^0_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{I.45}$$

   Only Rule P2-6 allows derivation of a time step for the operator σ^0_{rel}. From the premise of the rule,

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{I.46}$$
$$\langle x \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{I.47}$$

   By Induction on the above predicate, we get:

$$p_1 \equiv p_2$$

   Proved.

2. $p = \sigma^t_{\mathsf{rel}}(x) \qquad t > 0$

   Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{I.48}$$
$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{I.49}$$

   We distinguish between three cases depending on the duration r.

   (a) Case $r < t$

   Only Rule P2-9 can derive time steps I.48 and I.49. Then the target process terms in both time steps is $\sigma^{t-r}_{\mathsf{rel}}(x)$. I.e.,

$$p_1 = \sigma^{t-r}_{\mathsf{rel}}(x) \wedge p_2 = \sigma^{t-r}_{\mathsf{rel}}(x)$$

Hence

$$p_1 \equiv p_2$$

Proved.

(b) Case $r = t$
Rewriting time steps I.48 and I.49, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{t} \langle p_1 \rangle \tag{I.50}$$

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{t} \langle p_2 \rangle \tag{I.51}$$

Only Rule P2-9 can derive time steps I.50 and I.51. Then the target process terms in both time steps is x. Hence,

$$p_1 \equiv p_2 \equiv x$$

Proved.

(c) Case $r > t$
Let $r = u + t$, for $u > 0$.
Rewriting time steps I.48 and I.49, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{t+u} \langle p_1 \rangle \tag{I.52}$$

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \xmapsto{t+u} \langle p_2 \rangle \tag{I.53}$$

Only Rule P2-9 can derive time steps I.52 and I.53. From the premise of the rule, the following must hold:

$$\langle x \rangle \xmapsto{u} \langle p_1 \rangle \tag{I.54}$$

$$\langle x \rangle \xmapsto{u} \langle p_2 \rangle \tag{I.55}$$

By Induction,

$$p_1 \equiv p_2$$

Proved.

3. $p = x \cdot y$.
Suppose,

$$\langle x \cdot y \rangle \xmapsto{r} \langle p_1 \rangle \tag{I.56}$$

$$\langle x \cdot y \rangle \xmapsto{r} \langle p_2 \rangle \tag{I.57}$$

The above time steps can only be derived from Rule P2-17.

Then for some process term p_1', $p_1 = p_1' \cdot y$.

Rewriting Transition I.56:

$$\langle x \cdot y \rangle \xmapsto{r} \langle p_1' \cdot y \rangle \tag{I.58}$$

Also for some process term p_2', $p_1 = p_2' \cdot y$.

Rewriting Transition I.57:

$$\langle x \cdot y \rangle \overset{r}{\longmapsto} \langle p_2' \cdot y \rangle \tag{I.59}$$

From the premise of Rule P2-17 the following must hold:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1' \rangle \text{ and } \langle x \rangle \overset{r}{\longmapsto} \langle p_2' \rangle \tag{I.60}$$

By Induction

$$p_1' \equiv p_2'$$

Hence,

$$p_1' \cdot y \equiv p_2' \cdot y \text{ I.e. } p_1 \equiv p_2$$

Proved.

4. $p = x + y$.

Suppose,

$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{I.61}$$
$$\langle x + y \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{I.62}$$

Rule P2-24, Rule P2-25 or Rule P2-26 can be used to derive the above time steps. We discuss these rules one by one. We show both transitions are derived by the same rule and that only one rule is applicable at a time.

(a) *Rule P2-24*

Suppose Transition I.61 is derived from this rule. Then for some process terms x_1, y_1,

$$p_1 = x_1 + y_1 \tag{I.63}$$

From the premise of the rule the following holds:

$$\langle x \rangle \overset{r}{\longmapsto} \langle x_1 \rangle \tag{I.64}$$
$$\langle y \rangle \overset{r}{\longmapsto} \langle y_1 \rangle \tag{I.65}$$

From Transition I.64, ($\langle x \rangle \overset{r}{\longmapsto} \langle x_1 \rangle$), Rule P2-26 becomes inapplicable to derive a time step for $x + y$.

From Transition I.65, ($\langle y \rangle \overset{r}{\longmapsto} \langle y_1 \rangle$), Rule P2-25 becomes inapplicable to derive a time step for $x + y$.

Therefore Transition I.62 can also be only derived by Rule P2-24. From the premise of the rule, for some process terms x_2, y_2,

$$p_2 = x_2 + y_2 \tag{I.66}$$

and the following must hold:

$$\langle x \rangle \overset{r}{\longmapsto} \langle x_2 \rangle \tag{I.67}$$
$$\langle y \rangle \overset{r}{\longmapsto} \langle y_2 \rangle \tag{I.68}$$

Apply Induction Hypothesis on Transitions I.64 and I.67, and on Transitions I.65 and I.68. We get:

$$x_1 \equiv x_2$$
$$y_1 \equiv y_2$$

which implies

$$x_1 + y_1 \equiv x_2 + y_2$$

From Statements I.63 and I.66,

$$p_1 \equiv p_2$$

Proved.

(b) *Rule P2-25*

Suppose Transition I.61 is derived from this rule. From the premise of the rule the following holds:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{I.69}$$
$$\langle \texttt{consistent}\ \ y \rangle \tag{I.70}$$
$$\langle y \rangle \overset{r}{\not\longmapsto} \tag{I.71}$$
$$\forall s \leq r,\ \ \langle y \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.72}$$

From Transition I.69, ($\langle x \rangle \overset{r}{\longmapsto} \langle p_1 \rangle$), Rule P2-26 becomes inapplicable to derive a time step for $x + y$.

From Transition I.65, ($\langle y \rangle \overset{r}{\not\longmapsto}$), Rule P2-24 becomes inapplicable to derive a time step for $x + y$.

Hence Transition I.62 can only be derived from Rule P2-25.

From the premise of the rule, in addition to Predicates I.70, I.71 and I.72, the following holds:

$$\langle x \rangle \overset{r}{\longmapsto} \langle p_2 \rangle \tag{I.73}$$

Apply Induction Hypothesis on Transition I.69 and Transition I.73, we get:

$$p_1 \equiv p_2$$

Proved.

(c) *Rule P2-26*

Suppose Transition I.61 is derived from this rule. From the premise of the rule the following holds:

$$\langle y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle \tag{I.74}$$
$$\langle \texttt{consistent}\ \ x \rangle \tag{I.75}$$
$$\langle x \rangle \overset{r}{\not\longmapsto} \tag{I.76}$$
$$\forall s \leq r,\ \ \langle x \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.77}$$

From Transition I.74, ($\langle y \rangle \overset{r}{\longmapsto} \langle p_1 \rangle$), Rule P2-25 becomes inapplicable to derive a time step for $x + y$.

From Transition I.77, ($\langle x \rangle \overset{r}{\not\mapsto}$), Rule P2-24 becomes inapplicable to derive a time step for $x + y$.

Hence Transition I.62 can only be derived from Rule P2-26.

From the premise of the rule, in addition to Predicates I.75, I.76 and I.77, the following holds:

$$\langle y \rangle \overset{r}{\mapsto} \langle p_2 \rangle \tag{I.78}$$

Apply Induction Hypothesis on Transition I.74 and Transition I.78, we get:

$$p_1 \equiv p_2$$

Proved.

5. $p = \nu_{\mathsf{rel}}(x)$

   There are no rules to derive a time step for the now operator. Hence the theorem trivially holds.

⊠

I.4 Axiom A1 (Commutativity)

$$x + y = y + x$$

We prove that the choice operator + is commutative by showing that the TSS $T = (\Sigma, D)$, with Σ the signature of $BPA_{\perp}^{srt}$, and D the set of deduction rules in Table 3.13, is in comm-tyft format (extended with predicates and negative premises) given in [Mou05].

Define the mapping $\hbar$ defined on variables x, y, x', y' as follows:

$$\hbar(x) = y \quad \hbar(y) = x \quad \hbar(x') = y' \quad \hbar(y') = x'$$

Then it easy to prove that Rules P2-24 and P2-27 are commutative mirrors of themselves. Rules P2-20 and P2-21 are commutative mirrors of each other. Similarly, Rules P2-22 and P2-23, Rules P2-25 and P2-26, and Rules P2-28 and P2-29 are commutative mirrors of each other.

I.5 Axiom A2 (Associativity of Choice)

$(x + y) + z = x + (y + z)$ (Associativity of Alternative Composition-A2).

We need to prove, $(x + y) + z \leftrightarrows x + (y + z)$.

Let R be a binary relation on process terms defined as follows:

$$R \;=\; \{ \quad ((x + y) + z, x + (y + z)) \mid x, y, z \in P\}$$

We prove that the relation $R \cup \mathcal{I}$ is a bisimulation relation.

For all $a \in A, r > 0, x, y, z, p \in P$, the following holds:

1.
$$\begin{aligned}\langle (x+y)+z\rangle \xrightarrow{a} \langle p\rangle \implies \quad & \exists p' \in P : \langle x+(y+z)\rangle \xrightarrow{a} \langle p'\rangle \\ & \text{and } (p,p') \in R \cup \mathcal{I}\end{aligned}$$

Suppose,

$$\langle (x+y)+z\rangle \xrightarrow{a} \langle p\rangle \tag{I.79}$$

An action transition for an alternative composition can be derived only from rules P2 20 or P2 21. We discuss them one by one:

(a) Rule P2 20

If Transition I.79 is derived from this rule, then from the premise the following must hold:

$$\langle x+y\rangle \xrightarrow{a} \langle p\rangle \tag{I.80}$$
$$\langle \texttt{consistent}\ \ z\rangle \tag{I.81}$$

Again Transition I.80 can be derived from Rule P2 20 or Rule P2-21.

i. *Rule P2 20:*
If Transition I.80 is derived from this rule, then from the premise the following must hold:

$$\langle x\rangle \xrightarrow{a} \langle p\rangle \tag{I.82}$$
$$\langle \texttt{consistent}\ \ y\rangle \tag{I.83}$$

Apply Rule P2-27 on predicates I.81 and I.83, we get:

$$\langle \texttt{consistent}\ \ y+z\rangle \tag{I.84}$$

By applying Rule 20 on Transition I.82, for any process term q with $\langle \texttt{consistent}\ \ q\rangle$, the following holds:

$$\langle x+q\rangle \xrightarrow{a} \langle p\rangle$$

The term q can be $y+z$. Hence we have,

$$\langle x+(y+z)\rangle \xrightarrow{a} \langle p\rangle \tag{I.85}$$

Consider the target process terms in Transition I.79 and I.85. The pair (p,p) is in $\mathcal{I}$.

ii. *Rule P2 21:*
If Transition I.80 is derived from this rule, then from the premise the following must hold:

$$\langle y\rangle \xrightarrow{a} \langle p\rangle \tag{I.86}$$
$$\langle \texttt{consistent}\ \ x\rangle \tag{I.87}$$

By applying Rule 20 on Transition I.86, using Predicate I.81, we can derive the following transition:

$$\langle y+z\rangle \xrightarrow{a} \langle p\rangle \tag{I.88}$$

By applying Rule 21 on above transition, using Predicate I.87, we can derive the following transition:

$$\langle x + (y + z)\rangle \xrightarrow{a} \langle p\rangle \tag{I.89}$$

Consider the target process terms in Transition I.79 and I.89. The pair (p, p) is in $\mathcal{I}$.

(b) Rule P2 21

If Transition I.79 is derived from this rule, then from the premise the following must hold:

$$\langle z\rangle \xrightarrow{a} \langle p\rangle \tag{I.90}$$
$$\langle \texttt{consistent}\ \ x + y\rangle \tag{I.91}$$

Predicate I.91 can only hold, if

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.92}$$
$$\langle \texttt{consistent}\ \ y\rangle \tag{I.93}$$

By applying Rule 21 on Transition I.90, using Predicate I.93, we can derive the following transition:

$$\langle y + z\rangle \xrightarrow{a} \langle p\rangle \tag{I.94}$$

By applying Rule 21 on above transition, using Predicate I.92, we can derive the following transition:

$$\langle x + (y + z)\rangle \xrightarrow{a} \langle p\rangle \tag{I.95}$$

Consider the target process terms in Transition I.79 and I.95. The pair (p, p) is in $\mathcal{I}$.

2.
$$\langle x + (y + z)\rangle \xrightarrow{a} \langle p\rangle \implies \begin{array}{l}\exists p' \in P : \langle (x + y) + z\rangle \xrightarrow{a} \langle p'\rangle \\ \text{and } (p', p) \in R \cup \mathcal{I}\end{array}$$

Suppose,

$$\langle x + (y + z)\rangle \xrightarrow{a} \langle p\rangle \tag{I.96}$$

An action transition for an alternative composition can be derived only from rules P2 20 or P2 21. We discuss them one by one:

(a) Rule P2-20

If Transition I.96 is derived from this rule, then from the premise the following must hold:

$$\langle x\rangle \xrightarrow{a} \langle p\rangle \tag{I.97}$$
$$\langle \texttt{consistent}\ \ y + z\rangle \tag{I.98}$$

Predicate I.98 can only hold, if

$$\langle \texttt{consistent}\ y\rangle \tag{I.99}$$

$$\langle \texttt{consistent}\ z\rangle \tag{I.100}$$

By applying Rule 20 on Transition I.97, using Predicate I.99, we can derive the following transition:

$$\langle x+y\rangle \xrightarrow{a} \langle p\rangle \tag{I.101}$$

By again applying Rule 20 on above transition, using Predicate I.100, we can derive the following transition:

$$\langle (x+y)+z\rangle \xrightarrow{a} \langle p\rangle \tag{I.102}$$

Consider the target process terms in Transition I.96 and I.102. The pair (p, p) is in $\mathcal{I}$.

(b) Rule P2 21

If Transition I.96 is derived from this rule, then from the premise the following must hold:

$$\langle y+z\rangle \xrightarrow{a} \langle p\rangle \tag{I.103}$$

$$\langle \texttt{consistent}\ x\rangle \tag{I.104}$$

Again Transition I.103 can be derived from Rule P2 20 or Rule P2-21.

i. *Rule P2 20:*
If Transition I.103 is derived from this rule, then from the premise the following must hold:

$$\langle y\rangle \xrightarrow{a} \langle p\rangle \tag{I.105}$$

$$\langle \texttt{consistent}\ z\rangle \tag{I.106}$$

By applying Rule 21 on Transition I.105, using Predicate I.104, we can derive the following transition:

$$\langle x+y\rangle \xrightarrow{a} \langle p\rangle \tag{I.107}$$

By applying Rule 20 on above transition, using Predicate I.106, we can derive the following transition:

$$\langle (x+y)+z\rangle \xrightarrow{a} \langle p\rangle \tag{I.108}$$

Consider the target process terms in Transition I.96 and I.108. The pair (p, p) is in $\mathcal{I}$.

ii. *Rule P2 21:*
If Transition I.103 is derived from this rule, then from the premise the following must hold:

$$\langle z\rangle \xrightarrow{a} \langle p\rangle \tag{I.109}$$

$$\langle \texttt{consistent}\ y\rangle \tag{I.110}$$

Apply Rule P2-27 on predicates I.110 and I.104, we get:

$$\langle \texttt{consistent}\ \ x+y\rangle \tag{I.111}$$

By applying Rule 21 on Transition I.109, for any process term q with $\langle \texttt{consistent}\ \ q\rangle$, the following holds:

$$\langle q+z\rangle \xrightarrow{a} \langle p\rangle$$

The term q can be $x+y$. Hence we have,

$$\langle (x+y)+z\rangle \xrightarrow{a} \langle p\rangle \tag{I.112}$$

Consider the target process terms in Transition I.96 and I.112. The pair (p,p) is in $\mathcal{I}$.

3.
$$\begin{aligned}\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle p\rangle \implies \quad & \exists z' \in P : \langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle z'\rangle \\ & \text{and } (p,z') \in R \cup \mathcal{I}\end{aligned}$$

Suppose,

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.113}$$

Rules P2 24, Rule P2-25 or Rule P2-26 can be used to derive the above transition.

(a) Rule P2-24

If this rule is used to derive Transition I.113, then for some process terms p_1, p_2, $p = p_1 + p_2$. Rewriting Transition I.113:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle p_1+p_2\rangle \tag{I.114}$$

From premise of the rule,

$$\langle x+y\rangle \overset{r}{\longmapsto} \langle p_1\rangle \tag{I.115}$$

$$\langle z\rangle \overset{r}{\longmapsto} \langle p_2\rangle \tag{I.116}$$

Again Transition I.115 can be derived from one of the three rules: Rule P2 24, Rule P2-25 or Rule P2-26.

i. *Rule P2-24*
If Transition I.115 is derived from this rule, then for some process terms x_1, y_1, $p_1 = x_1 + y_1$. Rewriting Transition I.114 and Transition I.115, we get:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle (x_1+y_1)+p_2\rangle \tag{I.117}$$

$$\langle x+y\rangle \overset{r}{\longmapsto} \langle x_1+y_1\rangle \tag{I.118}$$

From the premise of the rule,

$$\langle x\rangle \overset{r}{\longmapsto} \langle x_1\rangle \tag{I.119}$$

$$\langle y\rangle \overset{r}{\longmapsto} \langle y_1\rangle \tag{I.120}$$

Apply Rule P2-24 on Transition I.116 and I.120, we get:

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle y_1+p_2\rangle \tag{I.121}$$

Again apply Rule P2-24 on Transition I.121 and I.119, we get:

$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle x_1+(y_1+p_2)\rangle \tag{I.122}$$

Consider the target process terms in transitions I.117 and I.122. The pair $((x_1+y_1)+p_2, x_1+(y_1+p_2))$ is in R.

ii. *Rule P2-25*
If Transition I.115 is derived from this rule, then from the premise of the rule,

$$\langle x\rangle \overset{r}{\longmapsto} \langle p_1\rangle \tag{I.123}$$
$$\langle \texttt{consistent}\ \ y\rangle \tag{I.124}$$
$$\langle y\rangle \overset{r}{\not\longmapsto} \tag{I.125}$$
$$\forall s \leq r\ \ \langle y\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.126}$$

On Transitions (Predicates) I.116, I.124 , I.125 and I.126, apply Rule P2-26, we get:

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle p_2\rangle \tag{I.127}$$

On Transitions I.123 and I.127, apply Rule P2-24, we get:

$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle p_1+p_2\rangle \tag{I.128}$$

Consider the target process terms in transitions I.114 and I.128. The pair (p_1+p_2, p_1+p_2) is in $\mathcal{I}$.

iii. *Rule P2-26*
If Transition I.115 is derived from this rule, then from the premise of the rule,

$$\langle y\rangle \overset{r}{\longmapsto} \langle p_1\rangle \tag{I.129}$$
$$\langle \texttt{consistent}\ \ x\rangle \tag{I.130}$$
$$\langle x\rangle \overset{r}{\not\longmapsto} \tag{I.131}$$
$$\forall s \leq r\ \ \langle x\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.132}$$

On Transitions I.129 and I.116, apply Rule P2-24, we get:

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle p_1+p_2\rangle \tag{I.133}$$

On Transitions (Predicates) I.133, I.130, I.131 and I.132, apply Rule P2-26, we get:

$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle p_1+p_2\rangle \tag{I.134}$$

Consider the target process terms in transitions I.114 and I.134. The pair (p_1+p_2, p_1+p_2) is in $\mathcal{I}$.

(b) Rule P2-25

If Transition I.113 is derived from this rule, then from the premise:

$$\langle x + y \rangle \xmapsto{r} \langle p \rangle \tag{I.135}$$

$$\langle \texttt{consistent}\ z \rangle \tag{I.136}$$

$$\langle z \rangle \not\xmapsto{r} \tag{I.137}$$

$$\forall s \leq r\ \ \langle z \rangle \not\xmapsto{s}_{\perp} \tag{I.138}$$

Again Transition I.135 can be derived from three rules. They are Rule P2-24, Rule P2-25 and Rule P2-26. We discuss them one by one.

i. *Rule P2-24*

If this rule is used to derive Transition I.135, then for some process terms x_1, y_1, $p = x_1 + y_1$. Rewriting Transitions I.113 and Transition I.135, we get:

$$\langle (x + y) + z \rangle \xmapsto{r} \langle x_1 + y_1 \rangle \tag{I.139}$$

$$\langle x + y \rangle \xmapsto{r} \langle x_1 + y_1 \rangle \tag{I.140}$$

From the premise of the rule:

$$\langle x \rangle \xmapsto{r} \langle x_1 \rangle \tag{I.141}$$

$$\langle y \rangle \xmapsto{r} \langle y_1 \rangle \tag{I.142}$$

On Transitions (Predicates) I.142, I.136, I.137 and I.138, apply Rule P2-25, we get:

$$\langle y + z \rangle \xmapsto{r} \langle y_1 \rangle \tag{I.143}$$

On Transitions I.143, I.141, apply Rule P2-24, we get:

$$\langle x + (y + z) \rangle \xmapsto{r} \langle x_1 + y_1 \rangle \tag{I.144}$$

Consider target process terms in Transitions I.139 and I.144. The pair $(x_1 + y_1, x_1 + y_1)$ is in $\mathcal{I}$.

ii. *Rule P2-25*

If this rule is used to derive Transition I.135, then from the premise of the rule, the following holds:

$$\langle x \rangle \xmapsto{r} \langle p \rangle \tag{I.145}$$

$$\langle \texttt{consistent}\ y \rangle \tag{I.146}$$

$$\langle y \rangle \not\xmapsto{r} \tag{I.147}$$

$$\forall s \leq r\ \ \langle y \rangle \not\xmapsto{s}_{\perp} \tag{I.148}$$

On Predicates I.136 and I.146, apply Rule P2-27, we get:

$$\langle \texttt{consistent}\ y + z \rangle \tag{I.149}$$

A time transition for $y + z$ with duration r can either be derived from Rule P2-24, Rule P2-25 or Rule P2-26. From Predicate I.147, Rules P2 24 and P2

25 cannot be applied. From Predicate I.137, Rule P2 26 cannot be applied. Hence we can conclude,

$$\langle y + z\rangle \overset{r}{\not\mapsto} \qquad (I.150)$$

A future inconsistency predicate for $y + z$ with duration $s \in (0, r]$ can either be derived from Rule P2-28, or Rule P2-29. From Predicate I.148, Rule P2 28 cannot be applied to derive a future inconsistency predicate of length $s \in (0, r]$ for $y + z$. From Predicate I.138, Rule P2 29 cannot be applied to derive a future inconsistency predicate of length $s \in (0, r]$ for $y + z$. Hence we can conclude,

$$\forall s \leq r \;\; \langle y + z\rangle \overset{s}{\not\mapsto}_{\bot} \qquad (I.151)$$

On Transitions (Predicates) I.149, I.150, I.151 and I.145, apply Rule P2 25. We get:

$$\langle x + (y + z)\rangle \overset{r}{\longmapsto} \langle p\rangle \qquad (I.152)$$

Consider target process terms in Transitions I.113 and I.152. The pair (p, p) is in $\mathcal{I}$.

iii. *Rule P2-26*
If this rule is used to derive Transition I.135, then from the premise of the rule, the following holds:

$$\langle y\rangle \overset{r}{\longmapsto} \langle p\rangle \qquad (I.153)$$
$$\langle \texttt{consistent}\;\; x\rangle \qquad (I.154)$$
$$\langle x\rangle \overset{r}{\not\mapsto} \qquad (I.155)$$
$$\forall s \leq r \;\; \langle x\rangle \overset{s}{\not\mapsto}_{\bot} \qquad (I.156)$$

On Transitions (Predicates) I.136, I.137, I.138 and I.153, apply Rule P2-25, we get:

$$\langle y + z\rangle \overset{r}{\longmapsto} \langle p\rangle \qquad (I.157)$$

On Transitions (Predicates) I.157, I.154, I.155 and I.156, apply Rule P2-26, we get:

$$\langle x + (y + z)\rangle \overset{r}{\longmapsto} \langle p\rangle \qquad (I.158)$$

Consider target process terms in Transitions I.113 and I.158. The pair (p, p) is in $\mathcal{I}$.

(c) Rule P2-26
If Transition I.113 is derived from this rule, then from the premise of the rule:

$$\langle z\rangle \overset{r}{\longmapsto} \langle p\rangle \qquad (I.159)$$
$$\langle \texttt{consistent}\;\; x + y\rangle \qquad (I.160)$$
$$\langle x + y\rangle \overset{r}{\not\mapsto} \qquad (I.161)$$
$$\forall s \leq r \;\; \langle x + y\rangle \overset{s}{\not\mapsto}_{\bot} \qquad (I.162)$$

Predicate I.160 can only be derived from Rule P2-27. Hence the premise of the rule must hold:

$$\langle \texttt{consistent}\ x\rangle \tag{I.163}$$

$$\langle \texttt{consistent}\ y\rangle \tag{I.164}$$

From Predicate I.162, we want to prove that the following holds:

$$\forall s \leq r\ \ \langle x\rangle \overset{s}{\not\longmapsto}_{\perp}$$

$$\forall s \leq r\ \ \langle y\rangle \overset{s}{\not\longmapsto}_{\perp}$$

We prove the above predicates by contradiction.
Suppose,

$$\exists_{u,u' \leq r} :\ \ \langle x\rangle \overset{u}{\longmapsto}_{\perp}\ \vee \langle y\rangle \overset{u'}{\longmapsto}_{\perp}$$

The above statement is equivalent to the statement below:

$$\exists_{u \leq r} :\ \ \langle x\rangle \overset{u}{\longmapsto}_{\perp}\ \vee \langle y\rangle \overset{u}{\longmapsto}_{\perp} \tag{I.165}$$

We discuss different cases of the Disjunction Predicate I.165 and show that all cases lead to a contradiction to Predicate I.162.

i. *Case* $\langle x\rangle \overset{u}{\longmapsto}_{\perp} \wedge \langle y\rangle \overset{u}{\longmapsto}_{\perp}$
   Then by Theorem 15,

   $$\forall t_1 < u\ \ \langle y\rangle \overset{t_1}{\not\longmapsto}_{\perp} \tag{I.166}$$

   From Predicate I.164,

   $$\langle \texttt{consistent}\ y\rangle \tag{I.167}$$

   Using Predicates I.165, I.167 and the assumption $\langle x\rangle \overset{u}{\longmapsto}_{\perp}$, apply Rule P2-28, we get:

   $$\langle x + y\rangle \overset{u}{\longmapsto}_{\perp} \tag{I.168}$$

   which is a contradiction to Predicate I.162.

ii. *Case* $\langle x\rangle \overset{u}{\longmapsto}_{\perp} \wedge \langle y\rangle \overset{u}{\not\longmapsto}_{\perp}$ For $v < u$, one of the two cases must hold:

   - *Case* $\langle y\rangle \overset{v}{\longmapsto}_{\perp}$
     Again from Theorem 15 using the assumption $(\langle x\rangle \overset{u}{\longmapsto}_{\perp})$, the following holds:

     $$\forall t_1 < v\ \ \langle x\rangle \overset{t_1}{\not\longmapsto}_{\perp} \tag{I.169}$$

     Using Predicates I.169, I.163 and the assumption $\langle y\rangle \overset{v}{\longmapsto}_{\perp}$, apply Rule P2-28, we get:

     $$\langle x + y\rangle \overset{v}{\longmapsto}_{\perp} \tag{I.170}$$

     which is a contradiction to Predicate I.162.
   - *Case* $\forall v < u :\ \ \langle y\rangle \overset{v}{\not\longmapsto}_{\perp}$
     Using Predicate I.164 and the assumptions $\langle x\rangle \overset{u}{\longmapsto}_{\perp}$ and $\forall v < u :\ \ \langle y\rangle \overset{v}{\not\longmapsto}_{\perp}$, apply Rule P2-28, we get:

     $$\langle x + y\rangle \overset{u}{\longmapsto}_{\perp} \tag{I.171}$$

     which is again a contradiction to Predicate I.162.

iii. *Case* $\langle x\rangle \overset{u}{\not\mapsto}_{\perp} \wedge \langle y\rangle \overset{u}{\longmapsto}_{\perp}$
Similar to as above.

Hence we conclude:

$$\forall s \leq r \;\; \langle x\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.172}$$

$$\forall s \leq r \;\; \langle y\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.173}$$

From Predicate I.161, we conclude that none of the Rules P2 24, P2 25 and P2 26 are applicable. If Rule P2-24 is inapplicable, then x and y cannot delay together for r time units. Suppose one of x and y can delay. Suppose, for some x',

$$\langle x\rangle \overset{r}{\longmapsto} \langle x'\rangle$$
$$\langle y\rangle \overset{r}{\not\mapsto}$$

Now from Transitions (Predicates) I.173,I.164 and the time transition for x' and impossibility of delay for y given above, Rule P2-25 becomes applicable and we can derive,

$$\langle x+y\rangle \overset{r}{\longmapsto} \langle x'\rangle$$

which is a contradiction to Predicate I.161. Similarly, if we suppose y can delay for r time units, then Rule P2 26 becomes applicable.
Hence we conclude that none of the process terms, x and y can delay.

$$\langle x\rangle \overset{r}{\not\mapsto} \tag{I.174}$$

$$\langle y\rangle \overset{r}{\not\mapsto} \tag{I.175}$$

On Transitions (Predicates) I.159, I.164, I.173 and I.175, apply Rule P2-26, we get:

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.176}$$

Again join Transitions (Predicates) I.163, I.172, I.174 and I.176 and apply Rule P2-26, we get:

$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.177}$$

Consider target process terms in transitions I.113 and I.177. The pair (p,p) is in $\mathcal{I}$.

4.
$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle p\rangle \implies \exists z' \in P : \langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle z'\rangle \text{ and } (z',p) \in R \cup \mathcal{I}$$

Suppose,

$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.178}$$

Rules P2 24, Rule P2-25 or Rule P2-26 can be used to derive the above transition.

(a) Rule P2-24

If this rule is used to derive Transition I.178, then for some process terms p_1, p_2, $p = p_1 + p_2$. Rewriting Transition I.178:

$$\langle x + (y + z)\rangle \xmapsto{r} \langle p_1 + p_2\rangle \tag{I.179}$$

From premise of the rule,

$$\langle x\rangle \xmapsto{r} \langle p_1\rangle \tag{I.180}$$

$$\langle y + z\rangle \xmapsto{r} \langle p_2\rangle \tag{I.181}$$

Again Transition I.181 can be derived from one of the three rules: Rule P2 24, Rule P2-25 or Rule P2-26.

i. *Rule P2-24*

If Transition I.181 is derived from this rule, then for some process terms y_2, z_2, $p_2 = y_2 + z_2$. Rewriting Transition I.179 and Transition I.181, we get:

$$\langle x + (y + z)\rangle \xmapsto{r} \langle p_1 + (y_2 + z_2)\rangle \tag{I.182}$$

$$\langle y + z\rangle \xmapsto{r} \langle y_2 + z_2\rangle \tag{I.183}$$

From the premise of the rule,

$$\langle y\rangle \xmapsto{r} \langle y_2\rangle \tag{I.184}$$

$$\langle z\rangle \xmapsto{r} \langle z_2\rangle \tag{I.185}$$

Apply Rule P2-24 on Transition I.180 and I.184, we get:

$$\langle x + y\rangle \xmapsto{r} \langle p_1 + y_2\rangle \tag{I.186}$$

Again apply Rule P2-24 on Transition I.186 and I.185, we get:

$$\langle (x + y) + z)\rangle \xmapsto{r} \langle (p_1 + y_2) + z_2)\rangle \tag{I.187}$$

Consider the target process terms in transitions I.182 and I.187. The pair $(p_1 + (y_2 + z_2), (p_1 + y_2) + z_2))$ is in R.

ii. *Rule P2-25*

If Transition I.181 is derived from this rule, then from the premise of the rule,

$$\langle y\rangle \xmapsto{r} \langle p_2\rangle \tag{I.188}$$

$$\langle \texttt{consistent}\ \ z\rangle \tag{I.189}$$

$$\langle z\rangle \not\xmapsto{r} \tag{I.190}$$

$$\forall s \leq r \quad \langle z\rangle \not\xmapsto{s}_{\perp} \tag{I.191}$$

On Transitions I.180 and I.188, apply Rule P2-24, we get:

$$\langle x + y\rangle \xmapsto{r} \langle p_1 + p_2\rangle \tag{I.192}$$

On Transitions (Predicates) I.192 , I.189, I.190 and I.191, apply Rule P2-25, we get:

$$\langle (x + y) + z\rangle \xmapsto{r} \langle p_1 + p_2\rangle \tag{I.193}$$

Consider the target process terms in transitions I.179 and I.193. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

iii. *Rule P2-26*

If Transition I.181 is derived from this rule, then from the premise of the rule,

$$\langle z\rangle \xmapsto{r} \langle p_2\rangle \tag{I.194}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.195}$$

$$\langle y\rangle \overset{r}{\not\mapsto} \tag{I.196}$$

$$\forall s \leq r \quad \langle y\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.197}$$

On Transitions (Predicates) I.195 , I.196, I.197 and I.180 and apply Rule P2-25, we get:

$$\langle x + y\rangle \xmapsto{r} \langle p_1\rangle \tag{I.198}$$

On Transitions I.194 and I.198, and apply Rule P2-24, we get:

$$\langle x + (y + z)\rangle \xmapsto{r} \langle p_1 + p_2\rangle \tag{I.199}$$

Consider the target process terms in transitions I.179 and I.199. The pair $(p_1 + p_2, p_1 + p_2)$ is in $\mathcal{I}$.

(b) Rule P2-25

If Transition I.178 is derived from this rule, then from the premise of the rule:

$$\langle x\rangle \xmapsto{r} \langle p\rangle \tag{I.200}$$

$$\langle \texttt{consistent}\ \ y + z\rangle \tag{I.201}$$

$$\langle y + z\rangle \overset{r}{\not\mapsto} \tag{I.202}$$

$$\forall s \leq r \ \ \langle y + z\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.203}$$

Predicate I.201 can only be derived from Rule P2-27. Hence the premise of the rule must hold:

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.204}$$

$$\langle \texttt{consistent}\ \ z\rangle \tag{I.205}$$

From Predicates I.203, I.204 ,I.205 and Theorem 15, by employing the same reasoning as given before we conclude:

$$\forall s \leq r \ \ \langle y\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.206}$$

$$\forall s \leq r \ \ \langle z\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.207}$$

From Predicate I.202, we conclude that none of the rules P2 24, P2 25 and P2 26 are applicable. Then y and z cannot both delay for r time units otherwise Rule P2-24 becomes applicable. Suppose one of y and z can delay. Suppose, for some y',

$$\langle y\rangle \xmapsto{r} \langle y'\rangle$$

$$\langle z\rangle \overset{r}{\not\mapsto}$$

Now from Transitions (Predicates) I.207, I.205, the time transition for y and the impossibility of delay predicate for z given above, Rule P2-25 becomes applicable and we can derive,

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle y'\rangle$$

which is a contradiction to Predicate I.202. Similarly, if we suppose, z can delay then Rule P2 26 becomes applicable.

Hence we conclude that none of the process terms, y and z can delay.

$$\langle y\rangle \overset{r}{\not\longmapsto} \tag{I.208}$$

$$\langle z\rangle \overset{r}{\not\longmapsto} \tag{I.209}$$

On Transitions (Predicates) I.200, I.204, I.206 and I.208, apply Rule P2-25, we get:

$$\langle x+y\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.210}$$

Again join Transitions (Predicates) I.205, I.207, I.209 and I.210, apply Rule P2-25, we get:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.211}$$

Consider target process terms in transitions I.178 and I.211. The pair (p,p) is in $\mathcal{I}$.

(c) Rule P2-26

If Transition I.178 is derived from this rule, then from the premise:

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.212}$$

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.213}$$

$$\langle x\rangle \overset{r}{\not\longmapsto} \tag{I.214}$$

$$\forall s \le r\ \ \langle x\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.215}$$

Again Transition I.212 can be derived from three rules. They are Rule P2-24, Rule P2-25 and Rule P2-26. We discuss them one by one.

i. *Rule P2-24*

If this rule is used to derive Transition I.212, then for some process terms y_1, z_1, $p = y_1 + z_1$. Rewriting Transitions I.178 and Transition I.212, we get:

$$\langle x+(y+z)\rangle \overset{r}{\longmapsto} \langle y_1+z_1\rangle \tag{I.216}$$

$$\langle y+z\rangle \overset{r}{\longmapsto} \langle y_1+z_1\rangle \tag{I.217}$$

From the premise of the rule:

$$\langle y\rangle \overset{r}{\longmapsto} \langle y_1\rangle \tag{I.218}$$

$$\langle z\rangle \overset{r}{\longmapsto} \langle z_1\rangle \tag{I.219}$$

On Transitions (Predicates) I.218, I.213, I.214 and I.215, apply Rule P2-26, we get:

$$\langle x+y\rangle \overset{r}{\longmapsto} \langle y_1\rangle \tag{I.220}$$

On Transitions I.220, I.219, apply Rule P2-24, we get:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle y_1+z_1\rangle \tag{I.221}$$

Consider target process terms in Transitions I.216 and I.221. The pair $(y_1 + z_1, y_1 + z_1)$ is in $\mathcal{I}$.

ii. *Rule P2-25*

If this rule is used to derive Transition I.212, then from the premise of the rule, the following holds:

$$\langle y\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.222}$$

$$\langle \texttt{consistent}\ \ z\rangle \tag{I.223}$$

$$\langle z\rangle \overset{r}{\not\longmapsto} \tag{I.224}$$

$$\forall s \leq r \quad \langle z\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.225}$$

On Transitions (Predicates) I.213, I.214, I.215 and I.222, apply Rule P2-26, we get:

$$\langle x+y\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.226}$$

On Transitions (Predicates) I.226, I.223, I.224 and I.225, apply Rule P2-25, we get:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.227}$$

Consider target process terms in Transitions I.178 and I.227. The pair (p, p) is in $\mathcal{I}$.

iii. *Rule P2-26*

If this rule is used to derive Transition I.212, then from the premise of the rule, the following holds:

$$\langle z\rangle \overset{r}{\longmapsto} \langle p\rangle \tag{I.228}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.229}$$

$$\langle y\rangle \overset{r}{\not\longmapsto} \tag{I.230}$$

$$\forall s \leq r \, \langle y\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.231}$$

On Predicates I.213 and I.229, apply Rule P2-27, we get:

$$\langle \texttt{consistent}\ \ x+y\rangle \tag{I.232}$$

A time transition for $x+y$ with duration r can either be derived from Rule P2-24, Rule P2-25 or Rule P2-26. From Predicate I.214, Rules P2 24 and P2 25 cannot be applied. From Predicate I.230, Rule P2 26 cannot be applied. Hence we can conclude,

$$\langle x+y\rangle \overset{r}{\not\longmapsto} \tag{I.233}$$

A future inconsistency predicate for $x+y$ with duration $s \in (0, r]$ can either be derived from Rule P2-28, or Rule P2-29. From Predicate I.215, Rule P2 28 cannot be applied to derive a future inconsistency predicate of length $s \in (0, r]$ for $x + y$. From Predicate I.231, Rule P2 29 cannot be applied to derive a future inconsistency predicate of length $s \in (0, r]$ for $x + y$. Hence we can conclude,

$$\forall s \leq r \;\; \langle x + y \rangle \overset{r}{\not\mapsto}_{\perp} \tag{I.234}$$

On Transitions (Predicates) I.228, I.232, I.233 and I.234, apply Rule P2 26. We get:

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{I.235}$$

Consider target process terms in Transitions I.178 and I.235. The pair (p, p) is in $\mathcal{I}$.

5.

$$\langle (x + y) + z \rangle \xrightarrow{a} \surd \iff \langle x + (y + z) \rangle \xrightarrow{a} \surd$$

Left Implication

Suppose,

$$\langle (x + y) + z \rangle \xrightarrow{a} \surd \tag{I.236}$$

A termination predicate for an alternative composition can be derived only from rules P2 22 or P2 23. We discuss them one by one:

(a) Rule P2 22

If Predicate I.236 is derived from this rule, then from the premise the following must hold:

$$\langle x + y \rangle \xrightarrow{a} \surd \tag{I.237}$$

$$\langle \texttt{consistent}\;\; z \rangle \tag{I.238}$$

Again Predicate I.237 can be derived from Rule P2 22 or Rule P2-23.

i. *Rule P2 22:*
If Predicate I.237 is derived from this rule, then from the premise the following must hold:

$$\langle x \rangle \xrightarrow{a} \surd \tag{I.239}$$

$$\langle \texttt{consistent}\;\; y \rangle \tag{I.240}$$

Apply Rule P2-27 on predicates I.238 and I.240, we get:

$$\langle \texttt{consistent}\;\; y + z \rangle \tag{I.241}$$

By applying Rule 22 on Predicate I.239, for any process term q with $\langle \texttt{consistent}\;\; q \rangle$, the following holds:

$$\langle x + q \rangle \xrightarrow{a} \surd$$

The term q can be $y + z$. Hence we have,

$$\langle x + (y + z) \rangle \xrightarrow{a} \surd \tag{I.242}$$

ii. *Rule P2 23:*
If Predicate I.237 is derived from this rule, then from the premise the following must hold:

$$\langle y \rangle \xrightarrow{a} \surd \tag{I.243}$$

$$\langle \texttt{consistent}\ x \rangle \tag{I.244}$$

By applying Rule 22 on Predicate I.243, using Predicate I.238, we can derive the following predicate:

$$\langle y + z \rangle \xrightarrow{a} \surd \tag{I.245}$$

By applying Rule 23 on above predicate, using Predicate I.244, we can derive the following predicate:

$$\langle x + (y + z) \rangle \xrightarrow{a} \surd \tag{I.246}$$

(b) Rule P2 23

If Predicate I.236 is derived from this rule, then from the premise the following must hold:

$$\langle z \rangle \xrightarrow{a} \surd \tag{I.247}$$

$$\langle \texttt{consistent}\ x + y \rangle \tag{I.248}$$

Predicate I.248 can only hold, if

$$\langle \texttt{consistent}\ x \rangle \tag{I.249}$$

$$\langle \texttt{consistent}\ y \rangle \tag{I.250}$$

By applying Rule 23 on Predicate I.247, using Predicate I.250, we can derive the following predicate:

$$\langle y + z \rangle \xrightarrow{a} \surd \tag{I.251}$$

By applying Rule 22 on above predicate, using Predicate I.249, we can derive the following predicate:

$$\langle x + (y + z) \rangle \xrightarrow{a} \surd \tag{I.252}$$

Right Implication

Suppose,

$$\langle x + (y + z) \rangle \xrightarrow{a} \surd \tag{I.253}$$

A termination predicate for an alternative composition can be derived only from rules P2 22 or P2 23. We discuss them one by one:

(a) Rule P2-22

If Predicate I.253 is derived from this rule, then from the premise the following must hold:

$$\langle x \rangle \xrightarrow{a} \surd \tag{I.254}$$

$$\langle \texttt{consistent}\ \ y + z \rangle \tag{I.255}$$

Predicate I.255 can only hold, if

$$\langle \texttt{consistent}\ \ y \rangle \tag{I.256}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{I.257}$$

By applying Rule 22 on Predicate I.254, using Predicate I.256, we can derive the following predicate:

$$\langle x + y \rangle \xrightarrow{a} \surd \tag{I.258}$$

By again applying Rule 22 on above predicate, using Predicate I.257, we can derive the following predicate:

$$\langle (x + y) + z \rangle \xrightarrow{a} \surd \tag{I.259}$$

(b) Rule P2 23

If Predicate I.253 is derived from this rule, then from the premise the following must hold:

$$\langle y + z \rangle \xrightarrow{a} \surd \tag{I.260}$$

$$\langle \texttt{consistent}\ \ x \rangle \tag{I.261}$$

Again Predicate I.260 can be derived from Rule P2 22 or Rule P2-23.

i. *Rule P2 22:*
If Predicate I.260 is derived from this rule, then from the premise the following must hold:

$$\langle y \rangle \xrightarrow{a} \surd \tag{I.262}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{I.263}$$

By applying Rule 23 on Predicate I.262, using Predicate I.261, we can derive the following predicate:

$$\langle x + y \rangle \xrightarrow{a} \surd \tag{I.264}$$

By applying Rule 22 on above predicate, using Predicate I.263, we can derive the following predicate:

$$\langle (x + y) + z \rangle \xrightarrow{a} \surd \tag{I.265}$$

ii. *Rule P2 23:*
If Predicate I.260 is derived from this rule, then from the premise the following must hold:

$$\langle z \rangle \xrightarrow{a} \surd \tag{I.266}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{I.267}$$

Apply Rule P2-27 on predicates I.267 and I.261, we get:

$$\langle \texttt{consistent}\ \ x + y \rangle \tag{I.268}$$

By applying Rule 23 on Predicate I.266, for any process term q with $\langle \texttt{consistent}\ \ q \rangle$, the following holds:

$$\langle q + z \rangle \xrightarrow{a} \surd$$

The term q can be $x + y$. Hence we have,

$$\langle (x + y) + z \rangle \xrightarrow{a} \surd \tag{I.269}$$

6.

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto}_{\perp} \iff \langle x + (y + z) \rangle \overset{r}{\longmapsto}_{\perp}$$

Left Implication

Suppose,

$$\langle (x + y) + z \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.270}$$

Rule P2-28 or Rule P2-29 can be used to derive the above transition.

(a) Rule P2-28
If Predicate I.270 is derived from this rule, then from the premise:

$$\langle x + y \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.271}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{I.272}$$

$$\forall s < r\, \langle z \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.273}$$

Again Predicate I.271 can be derived from two rules. They are Rule P2-28 and Rule P2-29. We discuss them one by one.

i. *Rule P2-28*
If this rule is used to derive Predicate I.271, then from the premise of the rule, the following holds:

$$\langle x \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.274}$$

$$\langle \texttt{consistent}\ \ y \rangle \tag{I.275}$$

$$\forall s < r \quad \langle y \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.276}$$

On Predicates I.272 and I.275, apply Rule P2-27, we get:

$$\langle \texttt{consistent}\ \ y + z \rangle \tag{I.277}$$

A future inconsistency predicate for $y + z$ with duration $s \in (0, r)$ can either be derived from Rule P2-28, or Rule P2-29. From Predicates I.276 and I.273, none of the rules can be applied. Hence we can conclude,

$$\forall s < r \, \langle y + z \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.278}$$

On Predicates I.277, I.278 and I.274, apply Rule P2 28. We get:

$$\langle x + (y + z) \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.279}$$

ii. *Rule P2-29*
If this rule is used to derive Predicate I.271, then from the premise of the rule, the following holds:

$$\langle y \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.280}$$

$$\langle \texttt{consistent } x \rangle \tag{I.281}$$

$$\forall s < r \quad \langle x \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.282}$$

On Predicates I.272, I.273 and I.280, apply Rule P2-28, we get:

$$\langle y + z \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.283}$$

On Predicates I.283, I.281 and I.282, apply Rule P2-29, we get:

$$\langle x + (y + z) \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.284}$$

(b) Rule P2-29
If Predicate I.270 is derived from this rule, then from the premise of the rule:

$$\langle z \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.285}$$

$$\langle \texttt{consistent } x + y \rangle \tag{I.286}$$

$$\forall s < r \, \langle x + y \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.287}$$

Predicate I.286 can only be derived from Rule P2-27. Hence the premise of the rule must hold:

$$\langle \texttt{consistent } x \rangle \tag{I.288}$$

$$\langle \texttt{consistent } y \rangle \tag{I.289}$$

From Predicates I.287, I.288 and I.289 and Theorem 15, we conclude:

$$\forall s < r \, \langle x \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.290}$$

$$\forall s < r \, \langle y \rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.291}$$

On Predicates I.285, I.289 and I.291, apply Rule P2-29, we get:

$$\langle y + z \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.292}$$

Again join Predicates I.288, I.290 and I.292 and apply Rule P2-29, we get:

$$\langle x + (y + z) \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.293}$$

Right Implication

Suppose,

$$\langle x + (y + z)\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.294}$$

Rule P2-28 or Rule P2-29 can be used to derive the above predicate.

(a) Rule P2-28
If Predicate I.294 is derived from this rule, then from the premise of the rule:

$$\langle x\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.295}$$
$$\langle \texttt{consistent}\ \ y + z\rangle \tag{I.296}$$
$$\forall s < r\,\langle y + z\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.297}$$

Predicate I.296 can only be derived from Rule P2-27. Hence the premise of the rule must hold:

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.298}$$
$$\langle \texttt{consistent}\ \ z\rangle \tag{I.299}$$

From Predicates I.297, I.298 and I.299 and Theorem 15, we conclude:

$$\forall s < r\,\langle y\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.300}$$
$$\forall s < r\,\langle z\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.301}$$

On Predicates I.295, I.298, I.300 , apply Rule P2-28:

$$\langle x + y\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.302}$$

Again join Predicates (Predicates) I.299, I.301, and I.302, apply Rule P2-28, we get:

$$\langle (x + y) + z\rangle \overset{r}{\longmapsto}_{\perp}$$

(b) Rule P2-29
If Predicate I.294 is derived from this rule, then from the premise:

$$\langle y + z\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.303}$$
$$\langle \texttt{consistent}\ \ x\rangle \tag{I.304}$$
$$\forall s < r\,\langle x\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.305}$$

Again Predicate I.303 can be derived from two rules. They are Rule P2-28 and Rule P2-29. We discuss them one by one.

i. *Rule P2-28*
If this rule is used to derive Predicate I.303, then from the premise of the rule, the following holds:

$$\langle y\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.306}$$
$$\langle \texttt{consistent}\ \ z\rangle \tag{I.307}$$
$$\forall s < r\quad \langle z\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.308}$$

On Predicates I.304, I.305 and I.306, apply Rule P2-29, we get:

$$\langle x+y\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.309}$$

On Predicates I.309, I.307 and I.308, apply Rule P2-28, we get:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.310}$$

ii. *Rule P2-29*
If this rule is used to derive Predicate I.303, then from the premise of the rule, the following holds:

$$\langle z\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.311}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.312}$$

$$\forall s < r\,\langle y\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.313}$$

On Predicates I.304 and I.312, apply Rule P2-27, we get:

$$\langle \texttt{consistent}\ \ x+y\rangle \tag{I.314}$$

A future inconsistency predicate for $x+y$ with duration $s \in (0,r)$ can either be derived from Rule P2-28, or Rule P2-29. From Predicates I.305 and I.313 none of the rules can be applied. Hence we can conclude,

$$\forall s < r\,\langle x+y\rangle \overset{r}{\not\longmapsto}_{\perp} \tag{I.315}$$

On Predicates I.311, I.314 and I.315, apply Rule P2 29. We get:

$$\langle (x+y)+z\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.316}$$

7.

$$\langle \texttt{consistent}\ \ (x+y)+z\rangle \iff \langle \texttt{consistent}\ \ x+(y+z)\rangle$$

Left Implication

Suppose,

$$\langle \texttt{consistent}\ \ (x+y)+z\rangle \tag{I.317}$$

The above predicate can only be derived from Rule P2-27. From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ \ x+y\rangle \tag{I.318}$$

$$\langle \texttt{consistent}\ \ z\rangle \tag{I.319}$$

Again Predicate I.319 can only be derived from Rule P2-27. From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ \ x\rangle \tag{I.320}$$

$$\langle \texttt{consistent}\ \ y\rangle \tag{I.321}$$

Apply Rule P2-27 on Predicates I.321 and I.319, we get:

$$\langle \texttt{consistent}\ \ y+z \rangle \tag{I.322}$$

Again apply Rule P2-27 on Predicates I.322 and I.320, we get:

$$\langle \texttt{consistent}\ \ x+(y+z) \rangle$$

Hence the left implication is proved.

Right Implication

Suppose,

$$\langle \texttt{consistent}\ \ x+(y+z) \rangle \tag{I.323}$$

The above predicate can only be derived from Rule P2-27. From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ \ x \rangle \tag{I.324}$$

$$\langle \texttt{consistent}\ \ y+z \rangle \tag{I.325}$$

Again Predicate I.325 can only be derived from Rule P2-27. From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ \ y \rangle \tag{I.326}$$

$$\langle \texttt{consistent}\ \ z \rangle \tag{I.327}$$

Apply Rule P2-27 on Predicates I.326 and I.324, we get:

$$\langle \texttt{consistent}\ \ x+y \rangle \tag{I.328}$$

Again apply Rule P2-27 on Predicates I.328 and I.327, we get:

$$\langle \texttt{consistent}\ \ (x+y)+z \rangle$$

Hence the right implication is proved.

I.6 Axiom SRT2

$\sigma^{v}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)) = \sigma^{v+u}_{\mathsf{rel}}(x) \qquad v, u \geq 0$(SRT2)

We need to prove, $\sigma^{v}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)) \leftrightarrows \sigma^{v+u}_{\mathsf{rel}}(x)$.

We do the proof in four steps:

Case $u = 0, v = 0$

The proof is trivial using Axiom SRT1 and the fact that bisimulation is a congruence.

Case $u > 0, v = 0$

The proof is trivial using Axiom SRT1 and the fact that bisimulation is a congruence.

Case $u = 0, v > 0$

The proof is trivial using Axiom SRT1 and the fact that bisimulation is a congruence.

Case $u > 0, v > 0$

Let R be a binary relation on process terms defined as follows:

$$R = \{ \quad (\sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)), \sigma^{t+u}_{\mathsf{rel}}(x)), \mid x \in P, 0 < t \leq v\}$$

We prove that the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

For all $a \in A, r > 0, x, y \in P$, the following holds:

1.
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \langle y \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle z' \rangle \\ \text{and } (p, z') \in R \cup \mathcal{I}. \end{array}$$

Suppose,
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \langle y \rangle$$

A process term with relative delay operator with duration greater than 0 cannot perform an action step. Hence our supposition doesn't hold.

2.
$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle y \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xrightarrow{a} \langle z' \rangle \\ \text{and } (p, z') \in R \cup \mathcal{I}. \end{array}$$

Suppose,
$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle y \rangle$$

A process term with relative delay operator with duration greater than 0 cannot perform an action step. Hence our supposition doesn't hold.

3.
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{r}{\longmapsto} \langle y \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle z' \rangle \\ \text{and } (p, z') \in R \cup \mathcal{I}. \end{array}$$

Suppose,
$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{r}{\longmapsto} \langle y \rangle \tag{I.329}$$

We distinguish between three cases for different values of r.

(a) Case $r < t$

Let $t = r + r_1$ for some r_1 with $0 < r_1 < t$.

Then Transition I.329 is derived from Rule P2-9 and $y = \sigma^{r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))$. Rewriting Transition I.329:
$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \tag{I.330}$$

By Rule P2-9 the following can be derived:

$$\langle \sigma^{r+r_1+u}_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto} \langle \sigma^{r_1+u}_{\mathsf{rel}}(x)\rangle \tag{I.331}$$

Consider the target process terms in Transitions I.330 and I.331. The pair $(\sigma^{r_1}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)), \sigma^{r_1+u}_{\mathsf{rel}}(x))$, where $0 < r_1 < t$ is in R.

(b) Case $r = t$

Then Transition I.329 is derived from Rule P2-10. Then $y = \sigma^u_{\mathsf{rel}}(x)$. Rewriting Transition I.329:

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \overset{t}{\longmapsto} \langle \sigma^u_{\mathsf{rel}}(x) \rangle \tag{I.332}$$

By Rule P2-9 the following can be derived:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x) \rangle \overset{t}{\longmapsto} \langle \sigma^u_{\mathsf{rel}}(x) \rangle \tag{I.333}$$

Consider the target process terms in Transitions I.332 and I.333. The pair $(\sigma^u_{\mathsf{rel}}(x), \sigma^u_{\mathsf{rel}}(x))$ is in $\mathcal{I}$.

(c) Case $r > t$

Let $r = t + s$, for some $s > 0$. Rewriting Transition I.329,

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \xmapsto{t+s} \langle y \rangle \tag{I.334}$$

The above transition can only be derived from Rule P2-11. From the premise of the rule, the following holds:

$$\langle \sigma^u_{\mathsf{rel}}(x) \rangle \overset{s}{\longmapsto} \langle y \rangle \tag{I.335}$$

We distinguish between three cases depending on different values of the duration s of the time step.

i. Case $s < u$

Let $u = s + s_1$, for some s_1 with $0 < s_1 < s$.
Then Transition I.335 can only be derived from Rule P2-9. Then $y = \sigma^{s_1}_{\mathsf{rel}}(x)$. Rewriting Transitions I.334 and I.335, we get:

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \xmapsto{t+s} \langle \sigma^{s_1}_{\mathsf{rel}}(x) \rangle \tag{I.336}$$

$$\langle \sigma^u_{\mathsf{rel}}(x) \rangle \overset{s}{\longmapsto} \langle \sigma^{s_1}_{\mathsf{rel}}(x) \rangle \tag{I.337}$$

From Rule P2-9, the following can be derived:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x) \rangle \xmapsto{t+s} \langle \sigma^{s_1}_{\mathsf{rel}}(x) \rangle \tag{I.338}$$

Consider the target process terms in Transitions I.336 and I.338. The pair $(\sigma^{s_1}_{\mathsf{rel}}(x), \sigma^{s_1}_{\mathsf{rel}}(x))$ is in $\mathcal{I}$.

ii. Case $s = u$

Then Transition I.335 can only be derived from Rule P2-10. Then $y = x$. Rewriting Transitions I.334 and I.335, we get:

$$\langle \sigma^t_{\mathsf{rel}}(\sigma^u_{\mathsf{rel}}(x)) \rangle \xmapsto{t+u} \langle x \rangle \tag{I.339}$$

$$\langle \sigma^u_{\mathsf{rel}}(x) \rangle \overset{u}{\longmapsto} \langle x \rangle \tag{I.340}$$

From the premise of Rule P2-10, the following must hold:

$$\langle \texttt{consistent}\ x \rangle$$

Applying Rule P2-10 on process term $\sigma_{\mathsf{rel}}^{t+u}(x)$, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{t+u} \langle x \rangle \tag{I.341}$$

Consider the target process terms in Transitions I.339 and I.341. The pair (x, x) is in $\mathcal{I}$.

iii. Case $s > u$

Let $s = u + t_1$, for some $t_1 > 0$. Rewriting Transitions I.334 and I.335, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+u+t_1} \langle y \rangle \tag{I.342}$$

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{u+t_1} \langle y \rangle \tag{I.343}$$

Transition I.343 can only be derived from Rule P2-11. Then from the premise of the rule the following must hold:

$$\langle x \rangle \xmapsto{t_1} \langle y \rangle \tag{I.344}$$

Apply Rule P2-11 on the above transition. For any $m > 0$, the following is derivable:

$$\langle \sigma_{\mathsf{rel}}^{m}(x) \rangle \xmapsto{m+t_1} \langle y \rangle$$

In the above transition, m can be $t + u$. Hence, we get:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{t+u+t_1} \langle y \rangle \tag{I.345}$$

Consider the target process terms in Transition I.342 and Transition I.345. The pair (y, y) is in $\mathcal{I}$.

4.

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{r} \langle y \rangle \implies \exists z' \in P : \langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{r} \langle z' \rangle \text{ and } (p, z') \in R \cup \mathcal{I}.$$

Suppose,

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{r} \langle y \rangle \tag{I.346}$$

We distinguish between three cases for different values of r.

(a) Case $r < (t + u)$

Again we distinguish between three cases:

i. *Case* $r < t$

Let $t = r + r_1$, for some r_1 such that, $0 < r_1 < t$.
Then Transition I.346 can only be derived from Rule P2-9. Then $y = \sigma_{\mathsf{rel}}^{r_1+u}(x)$.
Rewriting Transition I.346, we get:

$$\langle \sigma_{\mathsf{rel}}^{r+r_1+u}(x) \rangle \xmapsto{r} \langle \sigma_{\mathsf{rel}}^{r_1+u}(x) \rangle \tag{I.347}$$

Then from Rule P2-9, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{r+r_1}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{r} \langle \sigma_{\mathsf{rel}}^{r_1}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \tag{I.348}$$

Consider the target process terms in Transitions I.347 and I.348. For $0 < r_1 < t$, the pair $(\sigma_{\mathsf{rel}}^{r_1}(\sigma_{\mathsf{rel}}^{u}(x)), \sigma_{\mathsf{rel}}^{r_1+u}(x))$ is in R.

ii. *Case* $r = t$
Then Transition I.346 can only be derived from Rule P2-9. Then $y = \sigma_{\mathsf{rel}}^{u}(x)$.
Rewriting Transition I.346, we get:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \overset{t}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \tag{I.349}$$

From Rule P2-10, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \overset{t}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \tag{I.350}$$

Consider the target process terms in Transitions I.349 and I.350. The pair $(\sigma_{\mathsf{rel}}^{u}(x), \sigma_{\mathsf{rel}}^{u}(x))$ is in $\mathcal{I}$.

iii. *Case* $r > t$
Let $r = t + s$ for some $s > 0$.
Note that $s < u$ because of our assumption that $r < (t + u)$. Let $u = s + s_1$ for some s_1 such that $0 < s_1 < u$.
Rewriting Transition I.346, we get:

$$\langle \sigma_{\mathsf{rel}}^{t+s+s_1}(x) \rangle \xrightarrow{t+s} \langle \sigma_{\mathsf{rel}}^{s_1}(x) \rangle \tag{I.351}$$

By Rule P2-9, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{s+s_1}(x) \rangle \overset{s}{\longmapsto} \langle \sigma_{\mathsf{rel}}^{s_1}(x) \rangle \tag{I.352}$$

Apply Rule P2-11 on the above transition. We get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{s+s_1}(x)) \rangle \xrightarrow{t+s} \langle \sigma_{\mathsf{rel}}^{s_1}(x) \rangle \tag{I.353}$$

Consider the target process terms in Transitions I.351 and I.353. The pair $(\sigma_{\mathsf{rel}}^{s_1}(x), \sigma_{\mathsf{rel}}^{s_1}(x))$ is in $\mathcal{I}$.

(b) Case $r = (t + u)$

Then Transition I.346 can only be derived from Rule P2-10 and $y = x$.
Rewriting Transition I.346, we get:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xrightarrow{t+u} \langle x \rangle \tag{I.354}$$

From the premise of the rule, the following holds:

$$\langle \texttt{consistent}\ x \rangle$$

Apply Rule P2-10 on the above predicate, we get:

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \overset{u}{\longmapsto} \langle x \rangle \tag{I.355}$$

Apply Rule P2-11 on the above transition. We get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xrightarrow{t+u} \langle x \rangle \tag{I.356}$$

Consider the target process terms in Transitions I.354 and I.356. The pair (x, x) is in $\mathcal{I}$.

(c) Case $r > (t + u)$

Let $r = t + u + t_1$, for some $t_1 > 0$. Rewriting Transition I.346, we get:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{t+u+t_1} \langle y \rangle \tag{I.357}$$

From the premise of the rule the following must hold:

$$\langle x \rangle \xmapsto{t_1} \langle y \rangle \tag{I.358}$$

Apply Rule P2-11 on the above transition. We get:

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{u+t_1} \langle y \rangle \tag{I.359}$$

Again apply Rule P2-11 on the above transition. We get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+u+t_1} \langle y \rangle \tag{I.360}$$

Consider the target process terms in Transitions I.357 and I.360. The pair (y, y) is in $\mathcal{I}$.

5.
$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{r}_{\perp} \iff \langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xmapsto{r}_{\perp}$$

Left Implication

Suppose
$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{r}_{\perp} \tag{I.361}$$

The above predicate can only be derived from Rule P2-13 or Rule P2-14. We discuss them one by one.

(a) Rule P2-13

If Predicate I.361 is derived from this rule, then $r = t$. Rewriting Predicate I.361:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t}_{\perp} \tag{I.362}$$

From the premise of the rule, $\sigma_{\mathsf{rel}}^{u}(x)$ must not be consistent. But from Rule P2-12, a consistency predicate for process term $\sigma_{\mathsf{rel}}^{u}(x)$, with $u > 0$, always holds. We are discussing the case with $u > 0$. Hence Predicate I.362 cannot be derived.

(b) Rule P2-14

Then the length r of future inconsistency predicate I.361 is greater than t. Let $r = t + s$, for some $s > 0$. Rewriting Predicate I.361, we get:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xmapsto{t+s}_{\perp} \tag{I.363}$$

From the premise of Rule P2-14, the following holds:

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xmapsto{s}_{\perp} \tag{I.364}$$

The above predicate can again only be derived from Rule P2-13 or Rule P2-14. We discuss them one by one.

i. Rule P2-13

If Predicate I.364 is derived from this rule, then $s = u$. Rewriting Predicates I.363 and I.364, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xmapsto{t+u} \bot \tag{I.365}$$

$$\langle \sigma^{u}_{\mathsf{rel}}(x)\rangle \xmapsto{u} \bot \tag{I.366}$$

From the premise of the rule, the following holds:

$$\neg\langle \texttt{consistent}\ x\rangle \tag{I.367}$$

Apply Rule P2-13 on the above predicate. For any $m > 0$, the following is derivable:

$$\langle \sigma^{m}_{\mathsf{rel}}(x)\rangle \xmapsto{m} \bot$$

Then m can be $t + u$ and hence the following is derivable:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xmapsto{t+u} \bot \tag{I.368}$$

Consider Predicates I.365 and I.368. The left implication is proved.

ii. Rule P2-14

If Predicate I.364 is derived from this rule, then $s > u$. Let $s = u + t_1$, for some $t_1 > 0$. Rewriting Predicates I.363 and I.364, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x))\rangle \xmapsto{t+u+t_1} \bot \tag{I.369}$$

$$\langle \sigma^{u}_{\mathsf{rel}}(x)\rangle \xmapsto{u+t_1} \bot \tag{I.370}$$

And from the premise of the rule, the following holds:

$$\langle x\rangle \xmapsto{t_1} \bot \tag{I.371}$$

Apply Rule P2-14 on the above predicate. For any $m > 0$, the following is derivable:

$$\langle \sigma^{m}_{\mathsf{rel}}(x)\rangle \xmapsto{m+t_1} \bot \tag{I.372}$$

Then m can be $t + u$ and the following holds:

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xmapsto{t+u+t_1} \bot \tag{I.373}$$

Consider Predicates I.369 and I.373. The left implication is proved. Hence the left implication is proved.

Right Implication

Suppose

$$\langle \sigma^{t+u}_{\mathsf{rel}}(x)\rangle \xmapsto{r} \bot \tag{I.374}$$

The above predicate can only be derived from Rule P2-13 or Rule P2-14. We discuss them one by one.

(a) Rule P2-13

If Predicate I.374 is derived from this rule, then $r = t + u$. Rewriting Predicate I.374:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xrightarrow{t+u}\!\!\!\mapsto \bot \tag{I.375}$$

From the premise of Rule P2-13, the following holds:

$$\neg\langle \texttt{consistent}\ x \rangle$$

Using Rule P2-13 on the above predicate, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xrightarrow{u}\!\!\!\mapsto \bot \tag{I.376}$$

Using Rule P2-14 on the above predicate, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xrightarrow{t+u}\!\!\!\mapsto \bot \tag{I.377}$$

Consider Predicates I.374 and I.377. The right implication is proved.

(b) Rule P2-14

If Predicate I.374 is derived from this rule, then $r > t + u$.
Let $r = t + u + t_1$, for some $t_1 > 0$. Rewriting Predicate I.374, we get:

$$\langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xrightarrow{t+u+t_1}\!\!\!\mapsto \bot \tag{I.378}$$

From the premise of Rule P2-14, the following holds:

$$\langle x \rangle \xrightarrow{t_1}\!\!\!\mapsto \bot \tag{I.379}$$

Apply Rule P2-14 on the above predicate, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{u}(x) \rangle \xrightarrow{u+t_1}\!\!\!\mapsto \bot \tag{I.380}$$

Again apply Rule P2-14 on the above predicate, the following can be derived:

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xrightarrow{t+u+t_1}\!\!\!\mapsto \bot \tag{I.381}$$

Hence the right implication is proved.

6.

$$\langle \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \xrightarrow{a} \surd \iff \langle \sigma_{\mathsf{rel}}^{t+u}(x) \rangle \xrightarrow{a} \surd$$

Trivial. Both process terms cannot perform an action.

7.

$$\langle \texttt{consistent}\ \sigma_{\mathsf{rel}}^{t}(\sigma_{\mathsf{rel}}^{u}(x)) \rangle \iff \langle \texttt{consistent}\ \sigma_{\mathsf{rel}}^{t+u}(x) \rangle$$

Trivial. Both are consistent.

I.7 Axiom SRT3 (Time Determinism)

$\sigma^u_{\mathsf{rel}}(x) + \sigma^u_{\mathsf{rel}}(y) = \sigma^u_{\mathsf{rel}}(x+y) \qquad u \geq 0$ (Time determinism-SRT3).

We prove the soundness of the axiom in two steps:

Case $u = 0$

The proof is trivial using Axiom SRT1.

Case $u > 0$

We need to prove, $\sigma^u_{\mathsf{rel}}(x) + \sigma^u_{\mathsf{rel}}(y) \leftrightarrows \sigma^u_{\mathsf{rel}}(x+y)$.

Let R be a binary relation on process terms defined as follows:

$$R = \quad \{ \quad (\sigma^t_{\mathsf{rel}}(x+y), \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)), \mid x, y \in P, 0 < t \leq u\}$$

We prove that the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

For all $a \in A, r > 0, x, y, z \in P$, the following holds:

1.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \langle z \rangle \implies \begin{array}{l} \exists z' \in P : \langle \langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \rangle \xrightarrow{a} \langle z' \rangle \\ \text{and } (z, z') \in R \cup \mathcal{I} \end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \langle z \rangle \tag{I.382}$$

The above transition can be derived from Rules P2-20 or P2-21.

(a) *Rule P2-20*

From the premise of the rule, the following must hold:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \langle z \rangle \tag{I.383}$$

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(y) \rangle \tag{I.384}$$

Again there are no rules for $\sigma^t_{\mathsf{rel}}(x)$, with $t > 0$ to perform an action. Hence the transition I.382 cannot be derived from Rule P2-20.

(b) *Rule P2-21*

For similar reasons as given above for Rule P2-20, the transition I.382 cannot be derived from Rule P2-21.

2.

$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \xrightarrow{a} \langle z \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^u_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \langle z' \rangle \\ \text{and } (z', z) \in R \cup \mathcal{I} \end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \xrightarrow{a} \langle z \rangle$$

There are no rules allowing a process term $\sigma^r_{\mathsf{rel}}(x)$, with $r > 0$ to perform an action. Hence the above transition with an action step for $\sigma^t_{\mathsf{rel}}(x+y)$ does not exist. Since the left hand side of the implication is impossible, therefore we do not need to show that the right hand side holds.

3.
$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle z \rangle \implies \begin{array}{l} \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \overset{r}{\longmapsto} \langle z' \rangle \\ \text{and } (z, z') \in R \cup \mathcal{I} \end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle z \rangle \tag{I.385}$$

A time step for an alternative composition can be derived from one of the three rules P2 24, P2 25 or P2 26.

(a) Rule P2 24

Then for some process terms x' and y', the process term z in Transition I.385 is $x' + y'$.

Rewriting Transition I.385, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle x' + y' \rangle \tag{I.386}$$

From the premise of Rule P2 24 the following must be derivable:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle x' \rangle, \tag{I.387}$$

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle y' \rangle \tag{I.388}$$

We distinguish between three cases for the derivation of above time steps for different values of duration r.

i. Case $r < t$:

Let $t = r + r_1$, for some $0 < r_1 < t$.
Only Rule P2 9 allows to derive a time step of duration less than t for a process term $\sigma^t_{\mathsf{rel}}(x)$. According to the rule, the target process terms x' and y' in Transitions I.387 and I.388 are $\sigma^{r_1}_{\mathsf{rel}}(x)$ and $\sigma^{r_1}_{\mathsf{rel}}(y)$ respectively. Rewriting Transition I.386, we get:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) + \sigma^{r+r_1}_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y) \rangle \tag{I.389}$$

The following time step can also be derived from Rule P2 9:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x+y) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x+y) \rangle \tag{I.390}$$

Consider the target process terms in Transitions I.389 and I.390. The pair $(\sigma^{r_1}_{\mathsf{rel}}(x) + \sigma^{r_1}_{\mathsf{rel}}(y), \sigma^{r_1}_{\mathsf{rel}}(x+y))$, for $0 < r_1 < t$ is in R.

ii. Case $r = t$:

Then Transitions I.387 and I.388 can only be derived from Rule P2 10. According to the rule,

$$x' = x \text{ and } y' = y$$

Rewriting Transition I.386, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto} \langle x + y \rangle \tag{I.391}$$

From the premise of Rule P2 10, the following must hold:

$$\langle \texttt{consistent } x \rangle \text{ and } \langle \texttt{consistent } y \rangle$$

which implies by rule P2 27

$$\langle \texttt{consistent}\ \ x + y \rangle$$

Consequently, Rule P2 10 becomes applicable on $\sigma^t_{\mathsf{rel}}(x + y)$ to derive the following time step:

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{r} \langle x + y \rangle \tag{I.392}$$

Consider the target process terms in Transitions I.391 and I.392. The pair $(x + y, x + y)$ is in R.

iii. <u>Case $r > t$:</u>

Let $t = r + t_1$, for some t_1 with $0 < t_1 < t$.
Then Transitions I.387 and I.388 can only be derived from Rule P2 11. According to the premise of the rule, the following must be derivable:

$$\langle x \rangle \xmapsto{t_1} \langle x' \rangle, \tag{I.393}$$

$$\langle y \rangle \xmapsto{t_1} \langle y' \rangle \tag{I.394}$$

Joining the two transitions and applying Rule P2-24, we get:

$$\langle x + y \rangle \xmapsto{t_1} \langle x' + y' \rangle \tag{I.395}$$

Apply Rule P2 11 on the above Transition. We get:

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{r+t_1} \langle x' + y' \rangle \tag{I.396}$$

Consider the target process terms in Transitions I.386 and I.396. The pair $(x' + y', x' + y')$ is in R.

(b) <u>Rule P2 25</u>

We now inspect the case when Transition I.385 has been derived from Rule P2 25.

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{r} \langle z \rangle \qquad (I.385)$$

If Rule P2 25 is used to derive Transition I.385, then according to the rule $\sigma^t_{\mathsf{rel}}(x)$ must do the time step of duration r, $\sigma^t_{\mathsf{rel}}(y)$ must be unable to delay for duration r and $\sigma^t_{\mathsf{rel}}(y)$ must remain consistent throughout the delay. Mathematically, the requirements can be written as:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{r} \langle z \rangle, \tag{I.397}$$

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(y) \rangle, \tag{I.398}$$

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{r}, \tag{I.399}$$

$$(\forall s \leq r,\ \ \langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{s} \perp) \tag{I.400}$$

Again we distinguish three cases for different values of r.

i. Case $r < t$:

Let $t = r + r_1$, for some $0 < r_1 < t$.
A time step $\langle\sigma^{r+r_1}_{\mathsf{rel}}(y)\rangle \xmapsto{r} \langle\sigma^{r_1}_{\mathsf{rel}}(y)\rangle$ is always derivable (Rule P2 9). Hence Predicate I.399 does not hold for $r < t$.
We conclude that Transition I.385 cannot be derived from Rule P2 25 for $r < t$.

ii. Case $r = t$:

Rewriting the requirements for Rule P2 25 for $r = t$, we get:

$$\langle\sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t} \langle z\rangle, \tag{I.401}$$

$$\langle\texttt{consistent}\ \sigma^t_{\mathsf{rel}}(y)\rangle, \tag{I.402}$$

$$\langle\sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{t}, \tag{I.403}$$

$$(\forall s \leq t,\ \ \langle\sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{s}_{\perp}) \tag{I.404}$$

The Predicate I.403 indicates that Rule P2 10 is not applicable. Therefore y must be inconsistent. I.e.,

$$\neg\langle\texttt{consistent}\ y\rangle$$

If that is the case, then Rule P2 13 becomes applicable and the following can be derived:

$$\langle\sigma^t_{\mathsf{rel}}(y)\rangle \xmapsto{t}_{\perp} \tag{I.405}$$

which contradicts predicate I.404.
We conclude that Transition I.385 cannot be derived from Rule P2 25 for $r = t$.

iii. Case $r > t$:

Let $r = t + v$, for some $v > 0$.
If Rule P2 25 is used to derive Transition I.385, then according to the rule the following must hold:

$$\langle\sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+v} \langle z\rangle, \tag{I.406}$$

$$\langle\texttt{consistent}\ \sigma^t_{\mathsf{rel}}(y)\rangle, \tag{I.407}$$

$$\langle\sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{t+v}, \tag{I.408}$$

$$(\forall s \leq (t+v),\ \ \langle\sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{s}_{\perp}) \tag{I.409}$$

Transition I.406 can only be derived from Rule P2- 11. Then from the premise of the rule the following must be derivable:

$$\langle x\rangle \xmapsto{v} \langle z\rangle \tag{I.410}$$

From Predicate I.409 we can infer,

$$\langle\sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{t}_{\perp} \tag{I.411}$$

Hence Rule P2 13 must not be applicable. Therefore y must be consistent. I.e.,

$$\langle\texttt{consistent}\ y\rangle \tag{I.412}$$

Predicate I.408 indicates that Rule P2 11 cannot be applied to process term $\sigma^t_{\mathsf{rel}}(y)$. Hence the premise of the rule doesn't hold. I.e.,

$$\langle y \rangle \not\xmapsto{v} \tag{I.413}$$

Consider Predicate I.409. If we weaken the predicate, we have,

$$\forall s : t < s \leq (t+v),\ \ \langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{s}_{\perp} \tag{I.414}$$

(Note we are considering a future inconsistency predicate over a reduced range of s).
The above statement indicates that process term $\sigma^t_{\mathsf{rel}}(y)$ does not have a future inconsistency predicate of length greater than t and less than or equal to $t + v$. This means that Rule P2- 14 is not applicable for any duration in interval $(t, t + v]$. Hence the premise of the rule doesn't hold in the duration $(0, v]$.

$$\forall s \leq v,\ \ \langle y \rangle \not\xmapsto{s}_{\perp} \tag{I.415}$$

Apply Rule P2-25 to Transitions (Predicates) I.410,I.412,I.413 and I.415, we get:

$$\langle x + y \rangle \xmapsto{v} \langle z \rangle \tag{I.416}$$

Apply Rule P2 11 to the above transition. We get:

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{u+v} \langle z \rangle \tag{I.417}$$

Consider the target process terms in Transitions I.385 and I.417. The pair (z, z) is in $\mathcal{I}$.

(c) Rule P2 26

Reasoning similar to above applies.

4.

$$\begin{aligned}\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{r} \langle z \rangle \implies \quad & \exists z' \in P : \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{r} \langle z' \rangle \\ & \text{and } (z', z) \in R \cup \mathcal{I}\end{aligned}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x + y) \rangle \xmapsto{r} \langle z \rangle \tag{I.418}$$

We distinguish three cases depending on the length of duration r.

(a) Case $r < t$:

Let $t = r + r_1$, for some r_1 such that $0 < r_1 < t$.
When $r < t$, then Transition I.418 can only be derived from Rule P2-9. This rule has no premise. It can always be applied. Then, z must be $\sigma^{r_1}_{\mathsf{rel}}(x + y)$.

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x + y) \rangle \xmapsto{r} \langle \sigma^{r_1}_{\mathsf{rel}}(x + y) \rangle \tag{I.419}$$

The Rule P2-9 can be used to derive the following time steps:

$$\langle\sigma_{\mathsf{rel}}^{r+r_1}(x)\rangle \overset{r}{\longmapsto} \langle\sigma_{\mathsf{rel}}^{r_1}(x)\rangle \tag{I.420}$$

$$\langle\sigma_{\mathsf{rel}}^{r+r_1}(y)\rangle \overset{r}{\longmapsto} \langle\sigma_{\mathsf{rel}}^{r_1}(y)\rangle \tag{I.421}$$

Apply Rule P2 24 on the above transitions, we get:

$$\langle\sigma_{\mathsf{rel}}^{r+r_1}(x) + \sigma_{\mathsf{rel}}^{r+r_1}(y)\rangle \overset{r}{\longmapsto} \langle\sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y)\rangle \tag{I.422}$$

Consider the target process terms in transitions I.419 and I.422. For $r_1 < t$, the pair $(\sigma_{\mathsf{rel}}^{r_1}(x) + \sigma_{\mathsf{rel}}^{r_1}(y), \sigma_{\mathsf{rel}}^{r_1}(x+y))$ is in R.

(b) Case $r = t$:

When $r = t$, then Transition I.418 can only be derived from Rule P2 10. Then z must be of the form $x + y$. Rewriting Transition I.418

$$\langle\sigma_{\mathsf{rel}}^{t}(x+y)\rangle \overset{t}{\longmapsto} \langle x+y\rangle \tag{I.423}$$

And from the premise of the rule, the following must hold:

$$\langle\texttt{consistent}\ \ x+y\rangle \tag{I.424}$$

The above predicate can only hold when both x and y are consistent. I.e.,

$$\langle\texttt{consistent}\ \ x\rangle \text{ and } \langle\texttt{consistent}\ \ y\rangle \tag{I.425}$$

Then we can apply rule P2 10 to derive the following transitions for process terms $\sigma_{\mathsf{rel}}^{r}(x)$ and $\sigma_{\mathsf{rel}}^{r}(y)$:

$$\langle\sigma_{\mathsf{rel}}^{t}(x)\rangle \overset{t}{\longmapsto} \langle x\rangle \tag{I.426}$$

$$\langle\sigma_{\mathsf{rel}}^{t}(y)\rangle \overset{t}{\longmapsto} \langle y\rangle \tag{I.427}$$

Apply Rule P2 24 on the above transitions, we get:

$$\langle\sigma_{\mathsf{rel}}^{t}(x) + \sigma_{\mathsf{rel}}^{t}(y)\rangle \overset{t}{\longmapsto} \langle x+y\rangle \tag{I.428}$$

Consider the target process terms in transitions I.423 and I.428. The pair $(x + y, x + y)$ is in $\mathcal{I}$.

(c) Case $r > t$:

Let $r = t + v$, for some $v > 0$.

When $r > t$, then Transition I.418 can only be derived from Rule P2 11. From the premise of the rule, the following must hold:

$$\langle x+y\rangle \overset{v}{\longmapsto} \langle z\rangle \tag{I.429}$$

A time step for an alternative composition can be derived from rules P2 24, P2 25 and P2 26. We discuss each of the rules one by one:

i. *Rule P2 24:*

If this rule is used to derive Transition I.429, then both process terms x and y can do the time step. From the premise of the rule, for some process terms x', y', z must be of the form $x' + y'$. Rewriting Transitions I.418 and I.429:

$$\langle \sigma^{t}_{\mathsf{rel}}(x + y)\rangle \xmapsto{t+v} \langle x' + y'\rangle \tag{I.430}$$

$$\langle x + y\rangle \xmapsto{v} \langle x' + y'\rangle \tag{I.431}$$

And the following must hold:

$$\langle x\rangle \xmapsto{v} \langle x'\rangle \tag{I.432}$$

$$\langle y\rangle \xmapsto{v} \langle y'\rangle \tag{I.433}$$

On each of the above transitions, apply Rule P2 11, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x)\rangle \xmapsto{t+v} \langle x'\rangle \tag{I.434}$$

$$\langle \sigma^{t}_{\mathsf{rel}}(y)\rangle \xmapsto{t+v} \langle y'\rangle \tag{I.435}$$

Apply Rule P2 24 on the above transitions, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \xmapsto{t+v} \langle x' + y'\rangle \tag{I.436}$$

Consider the target process terms in transitions I.430 and I.436. The pair $(x' + y', x' + y')$ is in $\mathcal{I}$.

ii. *Rule P2 25:*

If this rule is used to derive Transition I.429, then according to the rule x must do the time step of duration v, y must be unable to delay for duration v and y must remain consistent throughout the delay. Mathematically, the requirements can be written as:

$$\langle x\rangle \xmapsto{v} \langle z\rangle, \tag{I.437}$$

$$\langle \texttt{consistent}\ \ y\rangle, \tag{I.438}$$

$$\langle y\rangle \overset{v}{\not\mapsto}, \tag{I.439}$$

$$(\forall s \leq v,\ \ \langle y\rangle \overset{s}{\not\mapsto}_{\perp}) \tag{I.440}$$

Apply Rule P2 11 on Transition I.437. We get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x)\rangle \xmapsto{t+v} \langle z\rangle \tag{I.441}$$

A process term $\sigma^{t}_{\mathsf{rel}}(y)$ may have future inconsistency predicates with durations greater than or equal to t. For a process term $\sigma^{t}_{\mathsf{rel}}(y)$, there are no rules allowing a predicate of future inconsistency with a duration s which is strictly less than t. Hence, the following predicate holds:

$$\forall s < t,\ \ \langle \sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.442}$$

For $\sigma^{t}_{\mathsf{rel}}(y)$ to have a future inconsistency with duration t, Rule P2 13 must be applicable. The rule has a premise that the predicate of consistency does not hold for y. But from Predicate I.438, we have: $\langle \texttt{consistent} \ y \rangle$. Hence Rule P2 13 cannot be applied and statement I.442 can be made stronger:

$$\forall s \leq t, \ \langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{s}_{\perp} \tag{I.443}$$

Rule P2 14 is the only rule by which we can derive a future inconsistency predicate of length $t+s$, (where $s > 0$) for the process term $\sigma^t_{\mathsf{rel}}(y)$. The rule requires that a predicate of future inconsistency with length s must hold for y, i.e. the predicate,

$$\langle y \rangle \xmapsto{s}_{\perp} \tag{I.444}$$

must hold.

If there does not hold such a predicate for y, then a future inconsistency predicate for $\sigma^t_{\mathsf{rel}}(y)$ with length $t+s$ cannot hold. Hence from Predicate I.440, $(\forall s \leq v, \ \langle y \rangle \not\xmapsto{s}_{\perp})$

We have,

$$\forall s \leq v, \ \langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{t+s}_{\perp} \tag{I.445}$$

Combining Predicates I.443 and I.445, we get:

$$\forall s \leq (t+v), \ \langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{s}_{\perp} \tag{I.446}$$

From Rule P2-12 the following holds:

$$\langle \texttt{consistent} \ \sigma^t_{\mathsf{rel}}(y) \rangle \tag{I.447}$$

Consider a process term $\sigma^t_{\mathsf{rel}}(z)$. Rule P2 11 is the only rule that allows $\sigma^t_{\mathsf{rel}}(z)$ to delay for a time duration $t+v$. The rule has a premise that z must be delayable for v time units. Hence from Predicate I.439 ($\langle y \rangle \not\xmapsto{v}$), we can infer the following:

$$\langle \sigma^t_{\mathsf{rel}}(y) \rangle \not\xmapsto{t+v} \tag{I.448}$$

Apply Rule P2-25 to Transitions (Predicates) I.441,I.446, I.447 and I.448 we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xmapsto{t+v} \langle z \rangle \tag{I.449}$$

Consider the target process terms in Transitions I.418 and I.449. The pair (z, z) is in R.

iii. *Rule P2 26:*

If this rule is used to derive Transition I.429, then according to the rule y must do the time step of duration r, x must be unable to delay for duration r and x must remain consistent throughout the delay.

Reasoning similar to the Rule P2 25 applies.

5.
$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \xrightarrow{a} \surd \iff \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \surd$$

Left Implication

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \xrightarrow{a} \surd$$

There are no rules allowing a process term $\sigma^r_{\mathsf{rel}}(x)$, with $r > 0$ to perform an action. Hence the above transition with an action step for $\sigma^t_{\mathsf{rel}}(x+y)$ does not exist.

Right Implication

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \xrightarrow{a} \surd \tag{I.450}$$

The above transition can be derived from Rules P2-22 or P2-23.

(a) *Rule P2-22*

From the premise of the rule, the following must hold:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xrightarrow{a} \surd \tag{I.451}$$

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(y) \rangle \tag{I.452}$$

Again there are no rules for $\sigma^t_{\mathsf{rel}}(x)$, with $t > 0$ to perform an action. Hence the transition I.450 cannot be derived from Rule P2-22.

(b) *Rule P2-23*

For similar reasons as given above for Rule P2-22, the transition I.450 cannot be derived from Rule P2-23.

6.

$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \overset{r}{\longmapsto}_{\perp} \Longleftrightarrow \langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y) \rangle \overset{r}{\longmapsto}_{\perp}$$

Left Implication

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \overset{r}{\longmapsto}_{\perp} \tag{I.453}$$

We distinguish three cases depending on the length of duration r.

(a) Case $r < t$:

There are no rules to derive a future inconsistency predicate for a process term $\sigma^t_{\mathsf{rel}}(z)$, with a length r which is less than u. Hence Predicate I.453 cannot be derived for $r < t$.

(b) Case $r = t$:

$$\langle \sigma^t_{\mathsf{rel}}(x+y) \rangle \overset{t}{\longmapsto}_{\perp} \tag{I.454}$$

Only Rule P2 13 can be used to derive the above predicate. From the premise of the Rule, the following holds:

$$\neg\langle \texttt{consistent } x+y \rangle \tag{I.455}$$

A consistency predicate for an alternative composition is derived from Rule P2-27. From Predicate I.455, the premise of the rule must not hold. Hence, at least one of the following holds:

$$\neg\langle \texttt{consistent } x \rangle \text{ and } \neg\langle \texttt{consistent } y \rangle \tag{I.456}$$

i. Case $\neg\langle\texttt{consistent}\ x\rangle$

If $\neg\langle\texttt{consistent}\ x\rangle$, then Rule P2 13 can be applied to derive the following:

$$\langle\sigma^{t}_{\mathsf{rel}}(x)\rangle \overset{t}{\longmapsto}\bot \tag{I.457}$$

There are no rules to derive a predicate of Future Inconsistency for a process term $\sigma^{t}_{\mathsf{rel}}(y)$, with a length r which is less than u. Hence the following holds:

$$\forall s < t,\ \ \langle\sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{s}{\not\longmapsto}\bot \tag{I.458}$$

Also, for $u > 0$, from Rule P2-12, the following holds:

$$\langle\texttt{consistent}\ \sigma^{t}_{\mathsf{rel}}(y)\rangle \tag{I.459}$$

Apply Rule P2 28 on Predicates I.457, I.458 and I.459. We get:

$$\langle\sigma^{t}_{\mathsf{rel}}(x) + \sigma^{u}_{\mathsf{rel}}(y)\rangle \overset{t}{\longmapsto}\bot \tag{I.460}$$

Consider Predicates I.454 and I.460. The Left implication is proved.

ii. Case $\neg\langle\texttt{consistent}\ y\rangle$

If $\neg\langle\texttt{consistent}\ y\rangle$, then by reasoning that is similar to above and by application of Rule P2 29, we get the following predicate:

$$\langle\sigma^{t}_{\mathsf{rel}}(x) + \sigma^{t}_{\mathsf{rel}}(y)\rangle \overset{t}{\longmapsto}\bot$$

(c) Case $r > t$:

Let $r = t + v$, where $v > 0$.

When $r > t$, then Predicate I.453 can only be derived from Rule P2 14. Rewriting Predicate I.453:

$$\langle\sigma^{t}_{\mathsf{rel}}(x + y)\rangle \overset{t+v}{\longmapsto}\bot \tag{I.461}$$

From the premise of the rule, the following must hold:

$$\langle x + y\rangle \overset{v}{\longmapsto}\bot \tag{I.462}$$

A predicate of future inconsistency for an alternative composition can only be derived from rules P2 28 or P2 29. We discuss each of the rules one by one:

i. *Rule P2 28:*

From the premise of the rule the following must hold:

$$\langle x\rangle \overset{v}{\longmapsto}\bot \tag{I.463}$$

$$\langle\texttt{consistent}\ y\rangle \tag{I.464}$$

$$\forall s < v,\ \ \langle y\rangle \overset{s}{\not\longmapsto}\bot \tag{I.465}$$

Apply Rule 14 on Predicate I.463. We get:

$$\langle\sigma^{t}_{\mathsf{rel}}(x)\rangle \overset{t+v}{\longmapsto}\bot \tag{I.466}$$

No rules for deriving a future inconsistency predicate of length less than r for a process term $\sigma^r_{\mathsf{rel}}(z)$. Hence the following predicate holds:

$$\forall s < t,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.467}$$

From Predicate I.464, Rule P2 13 cannot be applied. Hence:

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{t}{\not\longmapsto}_{\perp} \tag{I.468}$$

Combining Predicates I.467 and I.468, we get:

$$\forall s \leq t,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.469}$$

For process term $\sigma^t_{\mathsf{rel}}(y)$, a future inconsistency predicate of duration $t+s$, (where $s > 0$) can only be derived from Rule P2 14. From Predicate I.465, Rule P2 14 cannot be applied on process term $\sigma^t_{\mathsf{rel}}(y)$ for any duration in interval $(t, t+v)$. Hence the following predicate holds:

$$\forall s < v,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{t+s}{\not\longmapsto}_{\perp} \tag{I.470}$$

Combining Predicate I.469 and Predicate I.470, we get:

$$\forall s < t+v,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{s}{\not\longmapsto}_{\perp} \tag{I.471}$$

Also the following holds:

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(y)\rangle \tag{I.472}$$

Apply Rule P2 28 on Transitions (Predicates) I.466, I.472 and I.471. We get:

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{t+v}{\longmapsto}_{\perp}$$

Hence the left implication is proved.

ii. *Rule P2 29:*
Reasoning similar to Rule P2 28 applies.

Right Implication

Suppose

$$\langle \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.473}$$

Rules P2 28 or P2 29 can be applied to derive the above predicate.

(a) *Rule P2 28:*
If this rule is used to derive Predicate I.473, then according to the rule the following must hold:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto}_{\perp}, \tag{I.474}$$

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(y)\rangle, \tag{I.475}$$

$$(\forall s < r,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{s}{\not\longmapsto}_{\perp}) \tag{I.476}$$

We distinguish three cases depending on different values of r:

i. *Case* $r < u$
Then it is not possible to derive Predicate I.474. As the premises of Rule P2 28 are not satisfied, therefore we conclude that Rule P2 28 cannot be used to derive Predicate I.473 for $r < t$.

ii. *Case* $r = t$*:*
Rewriting the requirements of Rule P2 28:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t}_{\perp}, \tag{I.477}$$

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(y)\rangle, \tag{I.478}$$

$$(\forall s < t,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{s}_{\perp}) \tag{I.479}$$

Now Predicate I.477 can only be derived from Rule P2 13. Hence the premise of the rule must hold. I.e.,

$$\neg\langle \texttt{consistent } x\rangle$$

For an alternative composition, both alternatives must be consistent. Only then a consistency predicate for that alternative composition can be derived. Hence, from $\neg\langle \texttt{consistent } x\rangle$, the following holds:

$$\neg\langle \texttt{consistent } x + y\rangle$$

Apply Rule P2 13 on the above predicate:

$$\langle \sigma^u_{\mathsf{rel}}(x + y)\rangle \xmapsto{u}_{\perp} \tag{I.480}$$

Hence left implication is proved.

iii. *Case* $r > t$*:*
Let $r = t + v$, for some $v > 0$. Rewriting premises of Rule 28:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+v}_{\perp}, \tag{I.481}$$

$$\langle \texttt{consistent } \sigma^t_{\mathsf{rel}}(y)\rangle, \tag{I.482}$$

$$(\forall s < (t + v),\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{s}_{\perp}) \tag{I.483}$$

Predicate I.481 can only be derived from Rule P2 14. Hence the following must hold:

$$\langle x\rangle \xmapsto{v}_{\perp} \tag{I.484}$$

Now from Predicate I.483, a future inconsistency predicate for process term $\sigma^t_{\mathsf{rel}}(y)$ with length t does not hold. I.e.,

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{t}_{\perp}$$

That means Rule P2 13 is not applicable. Hence its premise must not hold. I.e.,

$$\langle \texttt{consistent } y\rangle \tag{I.485}$$

By weakening Predicate I.483, the following is inferred:

$$\forall s : t < s < (t + v),\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \not\xmapsto{s}_{\perp}$$

Rewriting the above predicate,

$$\forall s : s < v,\ \ \langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{t+v}{\not\longmapsto}_{\perp}$$

The above Predicate states that Rule P2 14 is not applicable for process term, $\sigma^t_{\mathsf{rel}}(y)$ for any duration in interval $(t, t+v)$. Hence the premise of the rule must not hold:

$$\forall s < v,\ \ \langle y\rangle \overset{s}{\not\mapsto}_{\perp} \tag{I.486}$$

Apply Rule P2 14 on Predicates I.484, I.485 and I.486, we get:

$$\langle x + y\rangle \overset{v}{\longmapsto}_{\perp} \tag{I.487}$$

Apply Rule P2 14 on the above predicate, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x+y)\rangle \overset{t+v}{\longmapsto}_{\perp}$$

(b) *Rule P2 29:*
If this rule is used to derive Predicate I.473, then according to the rule the following must hold:

$$\langle \sigma^t_{\mathsf{rel}}(y)\rangle \overset{r}{\longmapsto}_{\perp}, \tag{I.488}$$

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x)\rangle, \tag{I.489}$$

$$(\forall s < r,\ \ \langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{s}{\not\mapsto}_{\perp}) \tag{I.490}$$

Reasoning similar to given above for Rule P2 28 applies.

7.

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x+y)\rangle \iff \langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle$$

Left Implication

Suppose,

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x+y)\rangle$$

Rule P2-12 indicates that a process term with a relative delay of $t > 0$ time units is always consistent. Hence, the following holds:

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x)\rangle$$
$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(y)\rangle$$

Apply Rule P2-27 on the above two predicates, we get:

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x)\rangle + \langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(y)\rangle$$

Right Implication

Suppose,

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x) + \sigma^t_{\mathsf{rel}}(y)\rangle$$

From Rule P2-12, a process term with a relative delay of $t > 0$ time units is always consistent. Hence, the following holds:

$$\langle \texttt{consistent}\ \ \sigma^t_{\mathsf{rel}}(x+y)\rangle$$

I.8 Axiom SRT4

$\sigma^u_{\mathsf{rel}}(x) \cdot y = \sigma^u_{\mathsf{rel}}(x \cdot y) \qquad u \geq 0$ (SRT4)

We give the proof in two steps:

Case $u = 0$

The proof is trivial using Axiom SRT1.

Case $u > 0$

We need to prove, $\sigma^u_{\mathsf{rel}}(x) \cdot y \leftrightarrows \sigma^u_{\mathsf{rel}}(x \cdot y)$.

Let R be a binary relation on process terms defined as follows:

$$R = \{ \ (\sigma^t_{\mathsf{rel}}(x) \cdot y, \sigma^t_{\mathsf{rel}}(x \cdot y)) \mid x, y \in P, 0 < t \leq u\}$$

For all $x, y, p \in P$, $r > 0$, $a \in A$, the following holds:

1.
$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xrightarrow{a} \langle p\rangle \implies \begin{array}{l} \exists z \in P : \langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \xrightarrow{a} \langle z\rangle \\ \text{and } (p, z) \in R \cup \mathcal{I}. \end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xrightarrow{a} \langle p\rangle \tag{I.491}$$

The above action step can only be derived from Rule P2-15 or 16. We discuss the two cases one by one:

(a) Rule P2-15

If Transition I.491 is derived from this rule, then for some process term p', $p = p' \cdot y$. And from the premise of the rule, the following must be derivable,

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \langle p'\rangle \tag{I.492}$$

An action step for operator σ^t_{rel} with $t > 0$ cannot be derived from any rules. Hence we conclude that Rule P2-15 cannot be used to derive Transition I.491.

(b) Rule P2-16

If Transition I.491 is derived from this rule, then, $p = y$. And from the premise of the rule, the following must be derivable,

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xrightarrow{a} \surd \tag{I.493}$$

A termination step for operator σ^t_{rel} with $t > 0$ cannot be derived from any rules. Hence we conclude that Rule P2-16 cannot be used to derive Transition I.491.

Transition I.491 cannot be derived from any rules. Since the left hand side of the implication does not hold, therefore the implication holds.

2.
$$\langle \sigma^{t}_{\mathsf{rel}}(x \cdot y) \rangle \xrightarrow{a} \langle p \rangle \implies \begin{array}{l} \exists z \in P : \langle \sigma^{t}_{\mathsf{rel}}(x) \cdot y \rangle \xrightarrow{a} \langle z \rangle \\ \text{and } (z, p) \in R \cup \mathcal{I}. \end{array}$$

Suppose,

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \cdot y \rangle \xrightarrow{a} \langle p \rangle \tag{I.494}$$

An action step for operator $\sigma^{t}_{\mathsf{rel}}$ with $t > 0$ cannot be derived from any rules. Hence our supposition is wrong.

3.
$$\langle \sigma^{t}_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle p \rangle \implies \begin{array}{l} \exists z \in P : \langle \sigma^{t}_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\longmapsto} \langle z \rangle \\ \text{and } (p, z) \in R \cup \mathcal{I}. \end{array}$$

Suppose,

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{I.495}$$

The above time step can only be derived from Rule P2-17. Then for some process term p', $p = p' \cdot y$. Rewriting Transition I.495:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle p' \cdot y \rangle \tag{I.496}$$

From the premise of the rule the following holds:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle p' \rangle \tag{I.497}$$

We distinguish between three cases for different values of r:

(a) Case $r < t$

Let $t = r + r_1$, for some $r_1 > 0$.
Then Transition I.497 can only be derived from Rule P2-9. From the rule, we have $p' = \sigma^{r_1}_{\mathsf{rel}}(x)$. Rewriting Transitions I.496 and I.497:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \cdot y \rangle \tag{I.498}$$
$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \rangle \tag{I.499}$$

From Rule P2-9, the following can be

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x \cdot y) \rangle \tag{I.500}$$

Consider the target process terms in Transitions I.498 and I.500. For $0 < r_1 < t$, the pair $(\sigma^{r_1}_{\mathsf{rel}}(x) \cdot y, \sigma^{r_1}_{\mathsf{rel}}(x \cdot y))$ is in R.

(b) <u>Case $r = t$</u>
Then Transition I.497 can only be derived from Rule P2-10. From the rule, we have $p' = x$. Rewriting Transitions I.496 and I.497:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \overset{t}{\longmapsto} \langle x \cdot y \rangle \tag{I.501}$$

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{t}{\longmapsto} \langle x \rangle \tag{I.502}$$

From the premise of Rule P2-10, the following must hold:

$$\langle \texttt{consistent } x \rangle$$

Apply Rule P2-18 on the above predicate, we get:

$$\langle \texttt{consistent } x \cdot y \rangle \tag{I.503}$$

Apply Rule P2-10 on process term $\sigma^t_{\mathsf{rel}}(x \cdot y)$, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{t}{\longmapsto} \langle x \cdot y \rangle \tag{I.504}$$

Consider the target process terms in Transitions I.501 and I.504. The pair $(x \cdot y, x \cdot y)$ is in $\mathcal{I}$.

(c) <u>Case $r > t$</u>
Let $r = t + v$, for some $v > 0$.
Rewriting Transitions I.496 and I.497:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \overset{t+v}{\longmapsto} \langle p' \cdot y \rangle \tag{I.505}$$

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{t+v}{\longmapsto} \langle p' \rangle \tag{I.506}$$

Transition I.506 can only be derived from Rule P2-11. Then from the premise of the rule the following holds:

$$\langle x \rangle \overset{v}{\longmapsto} \langle p' \rangle \tag{I.507}$$

Apply Rule P2-17 on the above transition, we get:

$$\langle x \cdot y \rangle \overset{v}{\longmapsto} \langle p' \cdot y \rangle \tag{I.508}$$

Apply Rule P2-11 on the above transition, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{t+v}{\longmapsto} \langle p' \cdot y \rangle \tag{I.509}$$

Consider the target process terms in Transitions I.505 and I.509. The pair $(p' \cdot y, p' \cdot y)$ is in $\mathcal{I}$.

4.

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\longmapsto} \langle p \rangle \implies \begin{array}{l} \exists z \in P : \langle \sigma^t_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle z \rangle \\ \text{and } (z, p) \in R \cup \mathcal{I}. \end{array}$$

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\longmapsto} \langle p \rangle \tag{I.510}$$

We distinguish between three cases for different values of r.

(a) <u>Case $r < t$</u>

Let $t = r + r_1$, for some $r_1 < t$.

Then Transition I.510 can only be derived from Rule P2-9. From the rule, we have $p = \sigma^{r_1}_{\mathsf{rel}}(x \cdot y)$. Rewriting Transition I.510:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x \cdot y) \rangle \tag{I.511}$$

From Rule P2-9, the following can be derived:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \rangle$$

Apply Rule P2-17 on the above transition. We get:

$$\langle \sigma^{r+r_1}_{\mathsf{rel}}(x) \cdot y \rangle \overset{r}{\longmapsto} \langle \sigma^{r_1}_{\mathsf{rel}}(x) \cdot y \rangle \tag{I.512}$$

Consider the target process terms in Transitions I.511 and I.512. The pair $(\sigma^{r_1}_{\mathsf{rel}}(x) \cdot y), \sigma^{r_1}_{\mathsf{rel}}(x \cdot y))$ is in R.

(b) <u>Case $r = t$</u>

Then Transition I.510 can only be derived from Rule P2-10. From the rule, we have $p = x \cdot y$. Rewriting Transition I.510:

$$\langle \sigma^{t}_{\mathsf{rel}}(x \cdot y) \rangle \overset{t}{\longmapsto} \langle x \cdot y \rangle \tag{I.513}$$

The above time step can only be derived from Rule P2-10. From the premise of the rule,

$$\langle \texttt{consistent}\ x \cdot y \rangle$$

which can only be derived from Rule P2-18. Then the following must hold:

$$\langle \texttt{consistent}\ x \rangle$$

Apply Rule P2-10 on the above predicate, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \rangle \overset{t}{\longmapsto} \langle x \rangle \tag{I.514}$$

Apply Rule P2-17, we get:

$$\langle \sigma^{t}_{\mathsf{rel}}(x) \cdot y \rangle \overset{t}{\longmapsto} \langle x \cdot y \rangle \tag{I.515}$$

Consider the target process terms in Transitions I.504 and I.515. The pair $(x \cdot y, x \cdot y)$ is in R.

(c) <u>Case $r > t$</u>

Let $r = v + t$, for some $v > 0$.

Then Transition I.510 can only be derived from Rule P2-11. Rewriting Transition I.510:

$$\langle \sigma^{t}_{\mathsf{rel}}(x \cdot y) \rangle \xrightarrow{t+v} \langle p \rangle \tag{I.516}$$

From the premise of the rule,

$$\langle x \cdot y\rangle \overset{v}{\longmapsto} \langle p\rangle \tag{I.517}$$

The above transition can only be derived from Rule P2-17. Then for some process term p', $p = p' \cdot y$. Rewriting Transitions I.516 and I.517, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \xmapsto{t+v} \langle p' \cdot y\rangle \tag{I.518}$$

$$\langle x \cdot y\rangle \overset{v}{\longmapsto} \langle p' \cdot y\rangle \tag{I.519}$$

From the premise of the rule,

$$\langle x\rangle \overset{v}{\longmapsto} \langle p'\rangle \tag{I.520}$$

Apply Rule P2-11 on the above transition, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+v} \langle p'\rangle \tag{I.521}$$

Apply Rule P2-17 on the above transition, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xmapsto{t+v} \langle p' \cdot y\rangle \tag{I.522}$$

Consider the target process terms in Transitions I.516 and I.522. The pair $(p' \cdot y, p' \cdot y)$ is in $\mathcal{I}$.

5.

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \xrightarrow{a} \surd \iff \langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xrightarrow{a} \surd$$

Trivial therefore left.

6.

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \overset{r}{\longmapsto}_{\perp} \iff \langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \overset{r}{\longmapsto}_{\perp}$$

Left Implication

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.523}$$

The above predicate can only be derived from Rule P2-19. From the premise of the rule, the following holds:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \overset{r}{\longmapsto}_{\perp} \tag{I.524}$$

We distinguish between three cases for different values of r.

(a) Case $r < t$

For $r < t$, Predicate I.524 cannot be derived. We conclude that Predicate I.523 cannot be derived for $r < t$.

(b) Case $r = t$

Then Predicate I.524 can only be derived from Rule P2-13. Rewriting Predicate I.524, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \overset{t}{\mapsto} \bot \tag{I.525}$$

From the premise of the rule, the following holds:

$$\neg \langle \texttt{consistent}\ x \rangle$$

Then the following also holds:

$$\neg \langle \texttt{consistent}\ x \cdot y \rangle$$

Apply Rule P2-13 on the process term $\sigma^t_{\mathsf{rel}}(x \cdot y)$, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{t}{\mapsto} \bot$$

(c) Case $r > t$

Let $r = t + v$, for some $v > 0$.
Rewriting Predicate I.524:

$$\langle \sigma^t_{\mathsf{rel}}(x) \rangle \xmapsto{t+v} \bot \tag{I.526}$$

The above Predicate can only be derived from Rule P2-14. From the premise of the rule, the following holds:

$$\langle x \rangle \overset{v}{\mapsto} \bot \tag{I.527}$$

Apply Rule P2-19 on the above predicate, we get:

$$\langle x \cdot y \rangle \overset{v}{\mapsto} \bot$$

Apply Rule P2-14 on the above predicate, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \xmapsto{t+v} \bot$$

Right Implication

Suppose,

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y) \rangle \overset{r}{\mapsto} \bot \tag{I.528}$$

We distinguish between three cases for different values of r.

(a) Case $r < t$
A Future Inconsistency predicate for a process term with operator σ^t_{rel} of duration less than t cannot be derived. Hence, for $r < t$, Predicate I.528 cannot hold.

(b) Case $r = t$

Then Predicate I.528 can only be derived from Rule P2-13. From the premise of the rule, the following holds:

$$\neg\langle \texttt{consistent} \;\; x \cdot y\rangle$$

The above predicate can hold only if, $\neg\langle \texttt{consistent} \;\; x\rangle$ holds.

$$\neg\langle \texttt{consistent} \;\; x\rangle \tag{I.529}$$

Apply Rule P2-13 on the above predicate. We get:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t} \bot \tag{I.530}$$

Apply Rule P2-19 on the above predicate, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xmapsto{t} \bot$$

(c) Case $r > t$

Let $r = t + v$, for some $v > 0$.

Rewriting Predicate I.528, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \xmapsto{t+v} \bot \tag{I.531}$$

Predicate I.531 can only be derived from Rule P2-14. Then from the premise of the rule, the following holds:

$$\langle x \cdot y\rangle \xmapsto{v} \bot \tag{I.532}$$

Predicate I.532 can only be derived from Rule P2-19. Then from the premise of the rule, the following holds:

$$\langle x\rangle \xmapsto{v} \bot$$

Apply Rule P2-14 on the above predicate, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x)\rangle \xmapsto{t+v} \bot \tag{I.533}$$

Apply Rule P2-19 on the above predicate, we get:

$$\langle \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle \xmapsto{t+v} \bot \tag{I.534}$$

7.

$$\langle \texttt{consistent} \;\; \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle \iff \langle \texttt{consistent} \;\; \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle$$

From Rule P2-12,

$$\langle \texttt{consistent} \;\; \sigma^t_{\mathsf{rel}}(x \cdot y)\rangle$$

From Rule P2-18 and Rule P2-12, it can be derived that:

$$\langle \texttt{consistent} \;\; \sigma^t_{\mathsf{rel}}(x) \cdot y\rangle$$

I.9 Bisimulation Relations for other Axioms

1. $x + y = y + x$ **(A1)**

   We need to prove, $x + y \mathrel{\underline{\leftrightarrow}} y + x$

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + y, y + x) \mid x, y \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

   The proof is trivial and therefore left.

2. $x + x = x$ (Idempotency-**A3**)

   We need to prove, $x + x \mathrel{\underline{\leftrightarrow}} x$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + x, x) \mid x \in P\}$$

   Then using Theorems 14, 15 and 16, it is easy to prove that the relation $R \cup \mathcal{I}$ satisfies all conditions of a bisimulation relation .

3. $(x + y) \cdot z = x \cdot z + y \cdot z$ (Right Distributivity-**A4**).

   We need to prove, $(x + y) \cdot z \mathrel{\underline{\leftrightarrow}} x \cdot z + y \cdot z$.

   Let R be a binary relation on process terms defined as follows:

   $$R \;=\; \{ \;\; ((x + y) \cdot z, x \cdot z + y \cdot z) \mid x, y, z \in P\}$$

   Then the relation $R \cup \mathcal{I}$ is a bisimulation relation.

4. $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ (Associativity of sequential composition-**A5**).

   We need to prove, $(x \cdot y) \cdot z \mathrel{\underline{\leftrightarrow}} x \cdot (y \cdot z)$.

   Let R be a binary relation on process terms defined as follows:

   $$R \;=\; \{ \;\; ((x \cdot y) \cdot z, x \cdot (y \cdot z)) \mid x, y, z \in P\}$$

   Then the relation $R \cup \mathcal{I}$ is a bisimulation relation.

5. $x + \tilde{\tilde{\delta}} = x$ **(A6SR)**

   We need to prove, $x + \tilde{\tilde{\delta}} \mathrel{\underline{\leftrightarrow}} x$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + \tilde{\tilde{\delta}}, x) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

6. $\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$ **A7SR**

   We need to prove, $\tilde{\tilde{\delta}} \cdot x \,\underline{\leftrightarrow}\, \tilde{\tilde{\delta}}$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(\tilde{\tilde{\delta}} \cdot x, \tilde{\tilde{\delta}}) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

7. $x + \perp = \perp$ **(NE1)**

   We need to prove, $x + \perp \,\underline{\leftrightarrow}\, \perp$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(x + \perp, \perp) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

8. $\perp \cdot x = \perp$ **(NE2)**

   We need to prove, $\perp \cdot x \,\underline{\leftrightarrow}\, \perp$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(\perp \cdot x, \perp) \mid x \in P\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

9. $\tilde{\tilde{a}} \cdot \perp = \tilde{\tilde{\delta}}$ **(NE3)**

   We need to prove, $\tilde{\tilde{a}} \cdot \perp \,\underline{\leftrightarrow}\, \tilde{\tilde{\delta}}$.

   Let R be a binary relation on process terms defined as follows:

   $$R = \{(\tilde{\tilde{a}} \cdot \perp, \tilde{\tilde{\delta}}), (\tilde{\tilde{\delta}}, \tilde{\tilde{a}} \cdot \perp) \mid a \in A\}$$

   The relation $R \cup \mathcal{I}$ is a bisimulation relation.

10. $\sigma^0_{\mathsf{rel}}(x) = x$ **(SRT1)**

    We need to prove, $\sigma^0_{\mathsf{rel}}(x) \,\underline{\leftrightarrow}\, x$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\sigma^0_{\mathsf{rel}}(x), x) \mid x \in P\}$$

    Then $R \cup \mathcal{I}$ is a bisimulation relation that witnesses $\sigma^0_{\mathsf{rel}}(x) \,\underline{\leftrightarrow}\, x$.

11. $\nu_{\mathsf{rel}}(\tilde{\tilde{a}}) = \tilde{\tilde{a}}$ (**SRU1**)

    We need to prove, $\nu_{\mathsf{rel}}(\tilde{\tilde{a}}) \,\underline{\leftrightarrow}\, \tilde{\tilde{a}}$.

    Let R be a binary relation on process terms defined as follows:

    $$R = \{(\nu_{\mathsf{rel}}(\tilde{\tilde{a}}), \tilde{\tilde{a}}) \mid a \in A\}$$

    The relation $R \cup \mathcal{I}$ is a bisimulation relation.

12. $\nu_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)) = \tilde{\tilde{\delta}}$ **SRU2**

We need to prove, $\nu_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)) \leftrightarrows \tilde{\tilde{\delta}}$.

Let R be a binary relation on process terms defined as follows:

$$R = \{(\nu_{\mathsf{rel}}(\sigma^{u}_{\mathsf{rel}}(x)), \tilde{\tilde{\delta}}) \mid x \in P, u > 0\}$$

Then the relation $R \cup \mathcal{I}$ is a bisimulation relation.

13. $\nu_{\mathsf{rel}}(x + y) = \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)$ (**SRU3**)

We need to prove, $\nu_{\mathsf{rel}}(x + y) \leftrightarrows \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)$.

Let R be a binary relation on process terms defined as follows:

$$R = \{(\nu_{\mathsf{rel}}(x + y), \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)) \mid x, y \in P\}$$

Then the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

14. $\nu_{\mathsf{rel}}(x \cdot y) = \nu_{\mathsf{rel}}(x) \cdot y$ (**SRU4**)

We need to prove, $\nu_{\mathsf{rel}}(x \cdot y) \leftrightarrows \nu_{\mathsf{rel}}(x) \cdot y$.

Let R be a binary relation on process terms defined as follows:

$$R = \{(\nu_{\mathsf{rel}}(x \cdot y), \nu_{\mathsf{rel}}(x) \cdot y) \mid x, y \in P\}$$

Then the relation $R \cup \mathcal{I}$ satisfies all conditions of bisimulation.

15. $\nu_{\mathsf{rel}}(\perp) = \perp$ (**NESRU**)

We need to prove, $\nu_{\mathsf{rel}}(\perp) \leftrightarrows \perp$.

Let R be a binary relation on process terms defined as follows:

$$R = \{(\nu_{\mathsf{rel}}(\perp), \perp)\}$$

The proof is trivial and therefore left.

Summary

Our research is about formal specification and analysis of hybrid systems. The formalism used is process algebra. Hybrid systems are systems that exhibit both discrete and continuous behaviour. An example of a hybrid system is a digital controller controlling a physical device such as present in thermostats, fire-alarms, microwave ovens etc. Formal Methods is a branch of computer science that deals with the application of mathematics in software development. Using formal methods helps in removing ambiguities and errors from a specification and increases the developer's confidence in correctness and reliability of a system. Process algebras aim to represent processes algebraically. A process algebra can be compared to a *group* in mathematics. A process algebra has a signature with a set of operators and a set of constants. Complex processes can be built by combining simpler ones using operators. A process algebra has a set of rules called *axioms* that all processes must satisfy. Using axioms, different representations of a process can be examined.

In this thesis, we study the extensions of process algebras for description, analysis and verification of hybrid systems. During our four year research, we undertook the following projects:

1. A Comparative Study of Process Algebras for Hybrid Systems.

   We include four process algebras for hybrid systems in our study. We compare the process algebras for expressiveness, ease of modelling and strength of axiomatic reasoning. We also modelled a well-known case study of a train gate controller in all four process algebras and used the tools and techniques available for each process algebra to verify a safety condition.

2. Basic Timed Process Algebra with Non-existence ($BPA_{\perp}^{\mathrm{srt}}$).

   Process Algebra for Hybrid Systems (ACP_{hs}^{srt}) is a well-known formalism for hybrid system specification and analysis. Recently, a number of errors were found in it, namely the choice operator is not associative and a few axioms are unsound. ACP_{hs}^{srt} is built hierarchically from a number of simpler process algebraic theories. One of its basic components is $BPA_{\perp}^{srt}$. The unsoundness of the time determinism axiom can be traced down to this basic component. We think that fixing $BPA_{\perp}^{srt}$ is essential in rectifying the errors in ACP_{hs}^{srt}. Accordingly, we present two proposals for a sound process algebra $BPA_{\perp}^{\mathrm{srt}}$ and accompany our results with proofs.

3. Linearization of Hybrid χ.

   We developed an algorithm for linearization of specifications written in Hybrid χ language. Linearization is a procedure of rewriting a process term into a simpler form called a *linear* form. A linear form consists of only basic operators of an algebra such as

actions, choice and sequential composition. In particular, a linear form does not contain a parallel operator. The challenge in linearization is to keep the size of the resulting linear term small. In our algorithm, we use discrete counters to model interleaving of actions when a parallel operator is removed from a specification. Using these counters reduces the increase in the size of a specification during linearization.

The work on extending different process algebras to represent hybrid systems is rather recent. A reason for the relative unpopularity of process algebras for hybrid systems is their complex notation and lack of user friendly tool support. There are several ways in which improvements can be made in this area. At the end of our thesis, we give some suggestions for future research in process algebras for hybrid systems.

Curriculum Vitae

Uzma Khadim was born on the 22nd of September 1978 in Gujranwala, Pakistan. In 1998, she did her bachelors from Kinnaird College for Women, Lahore, Pakistan, with a major in Mathematics and Physics. In 2001, she received her Masters degree in Computer Science from Quaid-i-Azam University, Islamabad, Pakistan. Her masters thesis was titled “Interactive Algorithm Animation System”. From April 2002 till August 2002, she was a research fellow at the International Institute for Software Technology (UNU/IIST), UN University, Macau, China. There she worked on a project titled “Developing an XML-Enabled Serials Management System using RAISE”. In January 2004, she started her PhD studies at the Formal Methods Group in the Department of Mathematics and Computer Science at Eindhoven University of Technology, Netherlands. The research conducted during her PhD is related to formal specification and analysis of hybrid systems. The formalism studied is process algebra. The research resulted in a number of reports and this thesis. Uzma currently lives in Eindhoven with her husband Faisal Javed.

Titles in the IPA Dissertation Series since 2002

M.C. van Wezel. *Neural Networks for Intelligent Data Analysis: theoretical and experimental aspects.* Faculty of Mathematics and Natural Sciences, UL. 2002-01

V. Bos and J.J.T. Kleijn. *Formal Specification and Analysis of Industrial Systems.* Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2002-02

T. Kuipers. *Techniques for Understanding Legacy Software Systems.* Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2002-03

S.P. Luttik. *Choice Quantification in Process Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-04

R.J. Willemen. *School Timetable Construction: Algorithms and Complexity.* Faculty of Mathematics and Computer Science, TU/e. 2002-05

M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-time and Parametric Systems.* Faculty of Science, Mathematics and Computer Science, KUN. 2002-06

N. van Vugt. *Models of Molecular Computing.* Faculty of Mathematics and Natural Sciences, UL. 2002-07

A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems.* Faculty of Science, Mathematics and Computer Science, KUN. 2002-08

R. van Stee. *On-line Scheduling and Bin Packing.* Faculty of Mathematics and Natural Sciences, UL. 2002-09

D. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms.* Faculty of Mathematics and Natural Sciences, UL. 2002-10

M.B. van der Zwaag. *Models and Logics for Process Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-11

J.I. den Hartog. *Probabilistic Extensions of Semantical Models.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2002-12

L. Moonen. *Exploring Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-13

J.I. van Hemert. *Applying Evolutionary Computation to Constraint Satisfaction and Data Mining.* Faculty of Mathematics and Natural Sciences, UL. 2002-14

S. Andova. *Probabilistic Process Algebra.* Faculty of Mathematics and Computer Science, TU/e. 2002-15

Y.S. Usenko. *Linearization in μCRL.* Faculty of Mathematics and Computer Science, TU/e. 2002-16

J.J.D. Aerts. *Random Redundant Storage for Video on Demand.* Faculty of Mathematics and Computer Science, TU/e. 2003-01

M. de Jonge. *To Reuse or To Be Reused: Techniques for component composition and construction.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-02

J.M.W. Visser. *Generic Traversal over Typed Source Code Representations.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-03

S.M. Bohte. *Spiking Neural Networks.* Faculty of Mathematics and Natural Sciences, UL. 2003-04

T.A.C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing.* Faculty of Mathematics and Computer Science, TU/e. 2003-05

S.V. Nedea. *Analysis and Simulations of Catalytic Reactions*. Faculty of Mathematics and Computer Science, TU/e. 2003-06

M.E.M. Lijding. *Real-time Scheduling of Tertiary Storage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-07

H.P. Benz. *Casual Multimedia Process Annotation – CoMPAs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-08

D. Distefano. *On Modelchecking the Dynamics of Object-based Software: a Foundational Approach*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-09

M.H. ter Beek. *Team Automata – A Formal Approach to the Modeling of Collaboration Between System Components*. Faculty of Mathematics and Natural Sciences, UL. 2003-10

D.J.P. Leijen. *The λ Abroad – A Functional Approach to Software Components*. Faculty of Mathematics and Computer Science, UU. 2003-11

W.P.A.J. Michiels. *Performance Ratios for the Differencing Method*. Faculty of Mathematics and Computer Science, TU/e. 2004-01

G.I. Jojgov. *Incomplete Proofs and Terms and Their Use in Interactive Theorem Proving*. Faculty of Mathematics and Computer Science, TU/e. 2004-02

P. Frisco. *Theory of Molecular Computing – Splicing and Membrane systems*. Faculty of Mathematics and Natural Sciences, UL. 2004-03

S. Maneth. *Models of Tree Translation*. Faculty of Mathematics and Natural Sciences, UL. 2004-04

Y. Qian. *Data Synchronization and Browsing for Home Environments*. Faculty of Mathematics and Computer Science and Faculty of Industrial Design, TU/e. 2004-05

F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-06

L. Cruz-Filipe. *Constructive Real Analysis: a Type-Theoretical Formalization and Applications*. Faculty of Science, Mathematics and Computer Science, KUN. 2004-07

E.H. Gerding. *Autonomous Agents in Bargaining Games: An Evolutionary Investigation of Fundamentals, Strategies, and Business Applications*. Faculty of Technology Management, TU/e. 2004-08

N. Goga. *Control and Selection Techniques for the Automated Testing of Reactive Systems*. Faculty of Mathematics and Computer Science, TU/e. 2004-09

M. Niqui. *Formalising Exact Arithmetic: Representations, Algorithms and Proofs*. Faculty of Science, Mathematics and Computer Science, RU. 2004-10

A. Löh. *Exploring Generic Haskell*. Faculty of Mathematics and Computer Science, UU. 2004-11

I.C.M. Flinsenberg. *Route Planning Algorithms for Car Navigation*. Faculty of Mathematics and Computer Science, TU/e. 2004-12

R.J. Bril. *Real-time Scheduling for Media Processing Using Conditionally Guaranteed Budgets*. Faculty of Mathematics and Computer Science, TU/e. 2004-13

J. Pang. *Formal Verification of Distributed Systems*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-14

F. Alkemade. *Evolutionary Agent-Based Economics*. Faculty of Technology Management, TU/e. 2004-15

E.O. Dijk. *Indoor Ultrasonic Position Estimation Using a Single Base Station.* Faculty of Mathematics and Computer Science, TU/e. 2004-16

S.M. Orzan. *On Distributed Verification and Verified Distribution.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-17

M.M. Schrage. *Proxima - A Presentation-oriented Editor for Structured Documents.* Faculty of Mathematics and Computer Science, UU. 2004-18

E. Eskenazi and A. Fyukov. *Quantitative Prediction of Quality Attributes for Component-Based Software Architectures.* Faculty of Mathematics and Computer Science, TU/e. 2004-19

P.J.L. Cuijpers. *Hybrid Process Algebra.* Faculty of Mathematics and Computer Science, TU/e. 2004-20

N.J.M. van den Nieuwelaar. *Supervisory Machine Control by Predictive-Reactive Scheduling.* Faculty of Mechanical Engineering, TU/e. 2004-21

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support- .* Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics.* Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems.* Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication.* Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell.* Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model.* Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems.* Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems.* Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality.* Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications.* Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting.* Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs.* Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations.* Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data.* Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space.* Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization.* Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML.* Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models.* Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications.* Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols.* Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition.* Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems.* Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation.* Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures.* Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains.* Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use.* Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems.* Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML.* Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery.* Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding.* Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks.* Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data.* Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy.* Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS.* Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems.* Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates.* Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines.* Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras.* Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit.* Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks.* Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments.* Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design.* Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems.* Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms.* Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calam. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation.* Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems.* Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25