Formal specification and analysis of hybrid systems

Formal Specification and Analysis of Hybrid Systems

K.L. Man and R.R.H. Schiffelers

Reproduction: Universiteitsdrukkerij Technische Universiteit Eindhoven

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

ISBN-10: 90-386-2997-4

ISBN-13: 978-90-386-2997-1

Formal Specification and Analysis of Hybrid Systems

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 dinsdag 7 februari 2006 om 15.00 uur

door

Ka Lok Man
geboren te Hong Kong, China

en

Ramon Robert Hubert Schiffelers
geboren te Heerlen

Dit proefschrift is goedgekeurd door de promotoren van K.L. Man:
prof.dr. J.C.M. Baeten
en
prof.dr.ir. J.E. Rooda

Copromotor:
dr.ir. M.A. Reniers

Dit proefschrift is goedgekeurd door de promotoren van ir. R.R.H. Schiffelers:
prof.dr.ir. J.E. Rooda
en
prof.dr. J.C.M. Baeten

Copromotor:
dr.ir. D.A. van Beek

Preface

This thesis is a result of four years of intensive cooperation between the Systems Engineering Group of the Mechanical Engineering Department and the Formal Methods Group of the Mathematics and Computer Science Department at the Eindhoven University of Technology, and most of all, cooperation between the authors.

We would like to thank the following persons. Firstly, we thank our supervisors, prof.dr. J.C.M. Baeten and prof.dr.ir. J.E. Rooda for the opportunities given and support offered. Also, we thank our co-supervisors, dr.ir. D.A. van Beek and dr.ir. M.A. Reniers for the supervision of our Ph.D. research work. Secondly, we thank the members of our committee, prof.dr.ir. C.A. Middelburg, prof.dr. W.C. Rounds and prof.dr. A.J. van der Schaft for reviewing the manuscript of this thesis and giving us valuable comments. Thirdly, we thank our colleagues of the Systems Engineering Group, the Formal Methods Group, the Design and Analysis of Systems Group and the (Eindhoven) Embedded System Institute for contributing to a pleasant working atmosphere. Finally, we thank our families and friends for their support.

All research presented in this thesis is joint work. In particular, we would like to remark that the syntax and semantics of χ, and the formal translation of a subset of χ to hybrid automata and vice versa have been developed by both authors. In addition, K.L. Man has provided proofs for the semantics of χ, the correctness of the translations and tool support, and he developed the elimination theorems for a number of χ operators. The tools and examples have been developed by R.R.H. Schiffelers.

Summary

In this research, the hybrid χ (Chi) formalism has been developed. The hybrid χ formalism is suited to modeling, simulation and verification of hybrid systems. The semantics of hybrid χ is defined by means of deduction rules (in SOS style) that associate a hybrid transition system with a χ process. A set of axioms is presented for a notion of equivalence (bisimilation). The hybrid χ formalism integrates concepts from dynamics and control theory with concepts from computer science, in particular from process algebra and hybrid automata. It integrates ease of modeling with a straightforward semantics. Its 'consistent equation semantics' enforces state changes to be consistent with delay predicates, that combine the invariant and flow clauses of hybrid automata. Ease of modeling is ensured by means of the following concepts: 1) different classes of variables: discrete and continuous, of subclass jumping or non-jumping, and algebraic; 2) strong time determinism of alternative composition in combination with delayable guards; 3) integration of urgent and non-urgent actions; 4) differential algebraic equations as a process term as in mathematics; 5) steady-state initialization; and 6) several user-friendly syntactic extensions. Furthermore, the hybrid χ formalism incorporates several concepts for complex system specification: 1) process terms for scoping that integrate abstraction, local variables, local channels and local recursion definitions; 2) process definition and instantiation that enable process reuse, encapsulation, hierarchical and/or modular composition of processes; and 3) different interaction mechanisms: handshake synchronization and synchronous communication that allow interaction between processes without sharing variables, and shared variables that enable modular composition of continuous-time or hybrid processes.

In process algebra, linearization is a transformation of a recursive specification into a linear representation, i.e., a kind of normal form that is convenient for many forms of analysis. A first step towards the linearization of a reasonable subset of the hybrid χ language has been carried out in the form of elimination theorems for a number of χ operators.

Furthermore, a formal translation of a subset of χ to hybrid automata and vice versa has been defined. It is proved that any transition of a χ model can be mimicked by a transition in the corresponding hybrid automaton model and vice versa, which indicates that the translation as defined is correct. The translation from χ to hybrid automata

enables verification of χ models using existing hybrid automata based verification tools.

For the purpose of simulation and verification of χ models, tools have been developed. The stepper tool generates generalized transitions given a χ process. Based on this stepper, a symbolic simulator has been developed. Finally, the translation from χ to hybrid automata has been automated. The χ formalism is illustrated by means of examples taken from several application domains. Case studies have been conducted to test the developed tools.

Samenvatting

In dit onderzoek is het hybride χ (Chi) formalisme ontworpen. Dit formalisme is geschikt voor het modelleren, simuleren en verifiëren van hybride systemen. De semantiek van hybride χ is definiëerd met behulp van deductieregels (in SOS stijl) die een hybride transitiesysteem associëren met een χ process. Een set van axioma's is gedefinieerd voor een notie van gelijkheid (bisimulatie). Het hybride χ formalisme integreert concepten uit de dynamica en regeltheorie met concepten uit de informatica, in het bijzonder concepten uit de proces algebra en de theorie van hybride automaten. Het formalisme integreert de eenvoud van modelleren met een duidelijke semantiek. De 'consistente vergelijking semantiek' zorgt ervoor dat toestandsveranderingen consistent zijn met delay predicaten, die de invariant en flow clauses van hybride automaten omvatten. De eenvoud van modelleren is gegarandeerd door de volgende concepten: 1) verschillende klassen van variabelen: discreet en continu, met sub-klassen jumping en niet-jumping, en algebraisch; 2) sterk tijddeterminisme van de alternative compositie operator in combinatie met delayable guards; 3) integratie van urgente en niet-urgente acties; 4) algebraische differentiaalvergelijkingen als procestermen zoals in de wiskunde; 5) steady-state initializatie; en 6) verschillende gebruiksvriendelijke syntactische extensies. Verder omvat het hybride χ formalisme verschillende concepten voor de specificatie van complexe systemen: 1) procestermen voor scoping die abstractie, lokale variabelen, lokale kanalen en lokale recursiedefinities integreren; 2) procesdefinitie en procesinstantiatie die het hergebruiken van processen, encapsulatie, hierarchische en/of modulaire compositie van processen mogelijk maken; en 3) verschillende interactiemechanismen: handshake synchronizatie en handshake communicatie die interactie tussen processen zonder shared variabelen mogelijk maken, en shared variabelen die modulaire compositie van continue-tijd of hybride processen mogelijk maken.

In de proces algebra is linearizatie een tranformatie van een recursieve specificatie naar een lineaire representatie, ofwel een soort van normaalvorm die handig is voor veel vormen van analyse. Een eerste stap in de richting van linearizatie van een redelijke subset van het hybride χ formalisme is genomen in de vorm van eliminatietheorema's voor een aantal χ operators.

Verder is een formele translatie van een subset van χ naar hybride automaten en visa versa gedefinieerd. Het is bewezen dat iedere transitie van een χ model nagebootst kan

worden door een transitie van het corresponderende hybride automaten model en visa versa. Dit geeft aan dat de gedefinieerde translatie correct is. De translatie van χ naar hybride automaten maakt verificatie van χ modellen gebruik makend van bestaande verificatiegereedschappen voor hybride automaten mogelijk.

Voor simulatie en verificatie van χ modellen zijn gereedschappen ontwikkeld. Het 'stepper gereedschap' genereert gegeneralizeerde transities gegeven een χ proces. Gebaseerd op het stepper gereedschap is een symbolische simulator ontwikkeld. Verder is de translatie van χ naar hybride automaten geautomatiseerd.

Het χ formalisme is geillustreerd met behulp van voorbeelden uit verschillende toepassingsgebieden. Case studies zijn uitgevoerd om de ontwikkelde gereedschappen te testen.

Sommario

In questa ricerca è stato sviluppato il formalismo dell'hybrid χ (Chi). Tale formalismo adatto per modellare, simulare e verificare i sistemi ibridi. Le semantiche dell'hybrid χ sono definite per mezzo di regole deduttive (in stile SOS) che associano un sistema a transizione ibrida con un processo Chi. È stato presentato un insieme di assiomi, per un concetto di bisimilarità. Il formalismo hybrid χ integra concetti della dinamica e della teoria dei controlli con concetti informatici, in particolare dell'algebra dei processi e degli automi ibridi. Presenta facilità di modellizzazione insieme ad una semantica lineare. Le sue semantiche di equazioni consistenti obbligano i cambi di stato ad essere consistenti con i predicati di ritardo, che combinano gli invarianti e le proposizioni di flusso invarianti degli automi ibridi. La facilità di modellizzazione è garantita dai seguenti concetti: 1) differenti classi di variabili: discrete e continue, di jumping e non-jumping di sottoclassi, e algebriche; 2) forte determinismo temporale della composizione alternativa in combinazioni con guards ritardabili; 3) integrazione di azioni urgenti e non urgenti; 4) equazioni algebriche differenziali come un termine di processo, come in matematica; 5) inizializzazione steady-state; e 6) numerose espressioni sintattiche user-friendly. Inoltre, il formalismo dell'hybrid χ incorpora diversi concetti per la specifica di sistemi complessi: 1) termini di processo per scoping che integrano l'astrazione, le variabili locali, canali locali e definizioni di ricorsione locale; 2) definizione ed istanziazione dei processi che permette il riutilizzo dei processi, l'incapsulamento, la composizione gerarchica e/o modulare dei processi; e 3) differenti meccanismi di interazione: sincronizzazione handshake e comunicazione sincrona, che permettono l'interazione tra processi senza condivisione di variabili, e variabili condivise che permettono la composizione modulare dei processi continui nel tempo o ibridi.

Nell'algebra dei processi, la linearizzazione è una trasformazione di una specifica ricorsiva in una rappresentazione lineare, cioè un tipo di forma normale che è vantaggiosa per molte analisi. Un primo passo verso la linearizzazione di un ragionevole sottoinsieme del linguaggio χ è stato messo in pratica nella forma dei teoremi di eliminazione, per alcuni operatori di χ.

Inoltre, è stata definita la traduzione formale di un sottoinsieme di χ agli automi ibridi e vice versa. È stato provato che ogni transizione di un modello χ può essere "mimicked" da una transizione nel corrispondente automa ibrido e vice versa, il che indica la correttezza

della traduzione per come è stata definita. La traduzione del χ agli automi ibridi permette la verifica dei modelli χ usando gli strumenti di verifica esistenti basati sugli automi ibridi. Sono stati sviluppati degli strumenti informatici per la simulazione e verifica dei modelli χ. Lo strumento Stepper genera transizioni generalizzate. Basato Stepper, sono stati sviluppati due simulatori: un simulatore simbolico e un simulatore numerico basato sull'interfaccia Simulink delle funzioni S. Infine, è stata automatizzata la transizione da χ agli automi ibridi. Il formalismo χ è illustrato attraverso esempi tratti da parecchi campi di applicazione. Gli strumenti sviluppati sono stati validati per mezzo di alcuni casi di studio.

CONTENTS

CHAPTER

ONE

Introduction

Hybrid systems related research is based on two, originally different, world views: on the one hand the dynamics and control (DC) world view, and on the other hand the computer science (CS) world view.

The DC world view is that of a predominantly continuous-time system, which is modeled by means of differential (algebraic) equations, or by means of a set of trajectories. Hybrid phenomena are modeled by means of discontinuous functions and/or switched equation systems. The evolution of a hybrid system in the continuous-time domain is considered as a set of piecewise continuous functions of time (one for each variable).

Analysis and synthesis of hybrid systems in the DC domain are done, among others, by means of piecewise affine (PWA) systems, mixed logic dynamical (MLD) systems or linear complementarity (LC) systems, see [HSB01] for an overview relating these different classes, and see Chapter 5 for a translation of PWA systems to hybrid χ (Chi). A different framework to consider hybrid systems are differential (algebraic) equations with discontinuous right-hand sides, the semantics of which can be defined using differential inclusions. Such differential inclusions allow modeling of relays, valves or any kind of on/off switching elements at a high level of abstraction in control systems with so-called sliding modes [Fil88, Utk92].

The CS world view is that of a predominantly discrete-event system. A well-known model is a (hybrid) automaton, but modeling of discrete-event systems is also based on, among others, process algebra, Petri nets, and data flow languages. For modeling and analysis of hybrid phenomena, discrete-event formalisms are extended in different ways with some form of differential (algebraic) equations. The most influential hybrid system model is that of a hybrid automaton such as defined in [NOSY92, ACH$^+$95, AHH96, Hen00b, vdSS00, LSV03, LJS$^+$03]. An essential difference between such a hybrid automaton and a DC hybrid system model is that where in the DC hybrid model there are no actions, in the hybrid automaton, discontinuities take place mainly by means of (labeled) actions. By means of actions, the hybrid automaton switches from one mode/location to another mode/location.

1.1 The χ formalism

Clearly, hybrid systems represent a domain where the DC and CS world views meet, and we believe that a formalism that integrates the DC and CS world views is a valuable contribution towards integration of the DC and CS methods, techniques, and tools. The hybrid χ formalism is such a formalism. On the one hand, it can deal with continuous-time systems, PWA/MLD/LC systems, and hybrid systems based on sets of ordinary differential equations using discontinuous functions in combination with algebraic constraints (the DC approach). On the other hand, it can deal with discrete-event systems, without continuous variables or differential equations, and with hybrid systems in which discontinuities take place (mainly) by means of actions (the CS approach).

The intended use of hybrid χ is for modeling, simulation, verification, and real-time control. Its application domain ranges from physical phenomena, such as dry friction, to large and complex manufacturing systems. Although the semantics is formally defined, including a solution concept, the straightforward and elegant syntax and semantics is also highly suited to non-computer scientists. In the remainder of this thesis, we usually refer to hybrid χ as χ.

The most important concepts in χ are summarized below:

1. Integration between the DC and CS world views:

   - The DC world view in χ allows modeling of hybrid phenomena by means of discontinuous functions and/or switched equation systems. For this purpose, χ has introduced the category of algebraic variables, the trajectory of which can be discontinuous. Furthermore, the convex equality operator, defined in [vBPNR04], but not explained in detail in this thesis, allows modeling of differential inclusions according to the Filippov solution concept [Fil88]. The solution concept has been formalized in χ.
   - The CS world view in χ allows modeling of hybrid phenomena in a way that is strongly influenced by hybrid automata. In this respect, the new hybrid χ formalism differs considerably from its predecessor defined in [SvBM$^+$03a] which was quite different from hybrid automata. In the χ formalism described in this thesis, the 'consistent equation semantics' enforces changes in the values of variables to be consistent with delay predicates, that combine the invariant and flow clauses of hybrid automata. This is expressed by the property $p \parallel x = e \leftrightarrow p[e/x] \parallel x = e$, that, although not yet proved, we expect to hold. Here, $\parallel$ denotes parallel composition, $x = e$ is a mathematical equation, $p[e/x]$ denotes the process term obtained from p by substituting every free occurrence of variable x by its defining expression e, and $p \leftrightarrow q$ means that the two process terms p and q are bisimilar, that is they have the same behavior. For example: $x := y \parallel y = 1$ is bisimilar to $x := 1 \parallel y = 1$, where $x := y$ denotes an assignment of the value of y to variable x. A difference between the consistent equation semantics and the semantics of hybrid automata is that where the χ semantics

considers $\dot{x} = 1 \wedge \dot{x} = 2$ to be an inconsistent process term, the hybrid automaton can enter the location with flow clause $\dot{x} = 1 \wedge \dot{x} = 2$, but cannot delay in this location. The inconsistent process in a hybrid automaton is a location with invariant false. A translation from the hybrid automaton model defined in [Hen00b] to χ can be found in Chapter 5. This translation assumes that the flow clauses of the hybrid automaton cannot evaluate to false.

2. Integration of a straightforward semantics and ease of modeling.

   An important aspect is the conceptual similarity with hybrid automata as mentioned in the previous item. The concepts from hybrid automata have been extended in several ways to facilitate modeling. Where hybrid automata in general either have locations (e.g. [ACH+95, AHH96, Hen00b]) or discrete variables (e.g. [LJS+03]), and in addition either jumping or non-jumping continuous variables, χ has, among others, the following categories of variables:

   - Discrete variables, which facilitate compact readable specifications. In hybrid automata such variables are sometimes mimicked by real valued variables with a derivative of zero. However, for non-real valued variables, such as variables of type string, the concept of a zero derivative cannot be used.
   - Jumping continuous variables, that correspond to the continuous variables of hybrid automata as defined in, for example, [Hen00b]. The values of these variables are in principle allowed to jump (change) arbitrarily in an action transition, as long as the resulting values satisfy the action (jump) predicate, and the resulting process is consistent. Consider for example a system with three variables: x, y, z. If the value of x should change to 1, and the other variables should remain unchanged, the action (jump) predicate should be $x' = 1 \wedge y' = y \wedge z' = z$, or $x^+ = 1 \wedge y^+ = y^- \wedge z^+ = z^-$, depending on the syntax, where v' and v^+ denote the value of variable v after execution of the action, and v and v^- denote the value of variable v before execution of the action. Restrictions of the type $v^+ = v^-$ clutter the models, and are therefore often omitted in informal hybrid automata specifications. In order to allow fully formal models, without the clutter associated with the restrictions on non-jumping variables, χ has an additional class of variables: the non-jumping continuous variables.
   - Non-jumping continuous variables, that correspond to the continuous variables of hybrid automata as defined by, for example, the input language of the tool HyTech [HHWT95]. The values of these variables are not allowed to change in action transitions, unless their changes are explicitly specified, for example by means of assigning a new value to such a variable.
   - Algebraic variables, that can have discontinuous trajectories, as already discussed in the item on integration between the DC and CS world views.

There are also jumping discrete variables (used for the description of communication), (jumping) dotted continuous variables, and a predefined (non-jumping) variable denoting the current (model) time. For a full overview of the categories of variables in χ and their meaning, the readers is referred to 2.2.

Other concepts that enable integration of a straightforward semantics and ease of modeling are:

- Strong time-deterministic alternative composition operator. Where in many process algebras the passage of time can result in making a choice between the two operands of the choice or alternative composition operator, in χ, the passage of time can never result in such a choice. In the case of weak time-determinism, the alternative composition $\dot{x} = 1 \,[\!]\, x := 1$ (other languages may use the $+$ or $\oplus$ operators instead of $[\!]$) can non-deterministically choose between doing a delay according to and resulting in $\dot{x} = 1$, or doing the (undelayable) action $x := 1$. Strong time deterministic alternative composition means that alternative composition can delay only if both process terms can delay together, so that $\dot{x} = 1 \,[\!]\, x := 1$ can only do the (non-delayable) action $x := 1$, and then terminate. Hybrid automata have a comparable choice mechanism, apart from initialization. In a hybrid automaton, action transitions cannot disappear as a result of time passing. They can only be disabled for the period of time that the associated guard evaluates to false in the valuation prescribed by the trajectory of the variables. Also, time passing cannot result in the choice of a different location. The only changes in a hybrid automaton as a result of time passing are changes in the values of the variables. Only initially, depending on the initial edges and invariants, different initial locations may be selected as a result of time passing. Note that this does not imply that the χ formalism (or a hybrid automaton) is time deterministic. In the case of equations with multiple solutions, such as in $x^2 = 1$, delaying can take place according to any of the allowed solutions.
- Delayable guards. Where many process algebras have non-delayable guards, χ has delayable guards. A non-delayable guard cannot perform a delay when it is false. A delayable guard can delay when it is false until it becomes true, and thus facilitates modeling. Consider for example a valve α that must be switched on when the temperature T exceeds $T_{\max}$. Using a delayable guard, this can be modeled simply by $T \geq T_{\max} \rightarrow \alpha := \text{true}$.

  Delayable guards ensure that in $b \rightarrow h\,!\,b$, the value of expression b that is sent via channel h is always true. Note that $h\,!\,b$ can either do the send action, or delay for an arbitrary period of time. Non-delayable guards may lead to unintuitive behavior, because the value of b that is sent may be false. Consider the process term:

$$x := 0;\ (\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \,[\!]\, \Delta 10) \parallel \Delta 5;\ h\,?\,y),$$

where $\Delta\, s$ can delay for t time-units ($t \leq s$) to $\Delta\, s - t$, and $\Delta\, 0$ can terminate by means of an internal action.

Using non-delayable guards, the process term can perform the assignment, followed by a delay of at most 5, and after an internal action transforms into

$$\dot{x} = 1 \parallel (h\,!\,x \mathbin{[\!]} \Delta 5) \parallel h\,?\,y.$$

The guard that was true has disappeared at the start of the delay. If the communication via channel h takes place now, a value of 5 is sent, which does not conform to $x \leq 3$.

Using delayable guards on the other hand, the process term can do the assignment followed by a delay of at most 3, and transforms into:

$$\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \mathbin{[\!]} \Delta 7) \parallel \Delta 2;\ h\,?\,y,$$

where the value of x is 3. Communication is still not possible. After a delay of 2, followed by an internal action, the process term transforms into:

$$\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \mathbin{[\!]} \Delta 5) \parallel h\,?\,y,$$

where the value of x is 5, and after another delay of 5 it transforms into:

$$\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \mathbin{[\!]} \Delta 0) \parallel h\,?\,y.$$

The time-out takes place, leading to: $\dot{x} = 1 \parallel h\,?\,y$. Due to the delayable guard, that does not disappear while delaying, the communication does not take place, because the guard cannot be satisfied.

- Integrated urgent and non-urgent actions. Where most hybrid automata have non-urgent actions only, the χ formalism has both non-urgent actions and urgent actions. The concept of urgency is defined in a very flexible way: non-delayable actions are by definition urgent and delayable actions are non-urgent. This is achieved without any additional operators. The concept of urgency is built into the individual parallel composition operator, alternative composition operator, and guard operator. Consider the non-delayable action $x := 1$. The following three process terms
  - $\dot{x} = 1 \parallel x := 1$
  - $\dot{x} = 1 \mathbin{[\!]} x := 1$
  - $\dot{x} = 1 \parallel x \leq 0 \rightarrow x := 1$

  can each execute only the action $x := 1$, assuming that the value of x is initially non-positive. Consider now the delayable action $[x := 1]$. The following three process terms
  - $\dot{x} = 1 \parallel [x := 1]$

  - $\dot{x} = 1 \,[\!]\, [x := 1]$
  - $\dot{x} = 1 \parallel x \leq 0 \rightarrow [x := 1]$

  can each execute either the action $x := 1$ or perform a delay, assuming again that the value of x is initially non-positive.

  Communication on channels can also be urgent and non-urgent as in UPPAAL. This is achieved by means of an operator that partitions the set of channels into a set of urgent and a set of non-urgent channels. For the urgent channels, communication must take place as soon as it becomes possible, whereas for the non-urgent channels, no such preference for communication is assumed (see Section 2.4.8).

- Non-causal equations as in mathematics. Differential algebraic equations are process terms in hybrid χ. Therefore, they are modeled in χ in the same way as in mathematics.
- Steady state initialization. Dynamical analysis of physical systems often starts in initial steady-state conditions. This means that the initial state is such that all derivatives are zero. In χ, steady state initialization can be easily expressed by means of the signal emission operator. For example, $\dot{x} = 0 \curvearrowright \dot{x} = -x + 1$ represents the steady state initialization ($\dot{x} = 0$) of model $\dot{x} = -x + 1$. This means that this model only allows behavior for the case that initially $\dot{x} = 0$ holds, which implies that the initial value of x must be 1. In general, steady state initialization is not possible in this way for hybrid automata, because initial edges and invariants are usually predicates over variables, not derivatives. However, when the equations are straightforward enough, the modeler can explicitly calculate steady state conditions. In the example, variable x could be initialized to 1.
- Syntactic extensions. Ease of modeling is further supported in χ by extension of the set of orthogonal core process terms with additional process terms for ease of modeling. These additional process terms are defined by means of a straightforward translation into the core process terms.

3. Concepts for complex system specification:

   - Process terms for scoping that integrate abstraction, local variables, local channels and local recursion definitions.
   - Parameterized process definition and process instantiation that enable:
     - process re-use, and
     - encapsulation, hierarchical and/or modular composition of processes.
   - CSP communication and synchronization concepts that allow synchronization and communication without sharing of variables.
   - Shared variables, that enable modular composition of continuous or hybrid processes.

The history of the χ formalism dates back quite some time. It was originally designed as a modeling and simulation language for specification of discrete-event, continuous-time or combined discrete-event/continuous-time models. The first simulator [NA98], however, was suited to discrete-event models only. The simulator was successfully applied to a large number of industrial cases, such as an integrated circuit manufacturing plant, a brewery, and process industry plants [vBvdHR02]. Later, the hybrid language and simulator were developed [Fáb99, vBR00]. For the purpose of verification, the discrete-event part of the language was mapped onto the process algebra χ_σ by means of a syntactical translation. The semantics of χ_σ was defined using a structured operational semantics style (SOS), bisimulation relations were derived, and a model checker was built [BK02]. In this way, verification of discrete-event χ models was made possible [BK00]. The χ formalism defined in this thesis integrates the modeling language and the verification formalism. It integrates, extends and improves the syntax and semantics defined in [SvBM+03b] and [SvBM+03a].

1.2 Analysis of hybrid systems

In literature, many formal techniques for reasoning about the correctness of hybrid systems have been proposed [HHWT97, ABDM00, Fre05]. The goal of these formal techniques is to prove that the hybrid system (described in a formalism) performs as expected. One of the most successful formalisms for hybrid system verification is the theory of hybrid automata. Since the χ formalism is closely related to theory of hybrid automata, formal translations between them (in both directions) have been defined. The translation from hybrid automata to χ aims to show that the χ formalism is at least as expressive as the theory of hybrid automata. The translation from a reasonable subset of χ to hybrid automata enables verification of χ specifications using existing hybrid automata based verification tools. This translation has also been automated.

As an alternative to analyze χ specifications using hybrid automata based verification tools, χ simulators can be used to simulate χ specifications. Recently, a symbolic simulator has been developed for χ.

Like in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ [BM05] and HyPA [CR05], a set of basic terms (in χ) has been defined into which many closed terms can be rewritten using χ properties. This is so-called elimination, which is a useful step for algebraic analysis, because it reduces the complexity of specifications (without recursion variables) by transforming them into simpler forms. The elimination result allows to eliminate the parallel composition from many χ specifications, and it can be regarded as a preprocessing step for the linearization (transformation of a recursive specification into linear form) of χ processes.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 describes the syntax and informal semantics of the χ formalism. In Chapter 3, the semantics of χ is formally

specified, and a notion of equivalence is defined, which is shown to be a congruence for all χ operators. Furthermore, some useful properties of closed χ process terms are given in the same chapter. Several examples in Chapter 4 illustrate the use of the formalism. Translations between χ and other formalisms can be found in Chapter 5. Chapter 6 outlines the architecture and the functionality of the newly developed tools for χ. Some case studies of χ specifications are analyzed in Chapter 7. The elimination result of χ is shown in Chapter 8. Chapter 9 discusses related work, and Chapter 10 terminates with conclusions and points out directions for future work. Full proofs are presented in the appendices.

CHAPTER

TWO

Syntax and informal semantics of the Chi formalism

This section presents a concise definition of the syntax and informal semantics of χ. The syntax definition is incomplete in the sense that the syntax of predicates, expressions, etc. is defined on a high level of abstraction. This is done because different implementations of χ, such as tools for simulation, verification, or real-time control, may impose different syntactical restrictions. The intention of this chapter is to define the χ formalism that encompasses a variety of (future) tools without posing unnecessary syntactical restrictions.

2.1 Syntax of processes

A χ process is a triple $\langle p, \sigma, E \rangle$, where p denotes a process term, σ denotes a valuation, and E denotes an environment. The syntax of process terms is introduced in Section 2.3. A valuation is a partial function from variables to values. Syntactically, a valuation is denoted by a set of pairs $\{x_0 \mapsto c_0, \ldots, x_n \mapsto c_n\}$, where x_i denotes a variable and c_i its value.

An environment E is a tuple (C, J, L, H, R), where C denotes the set of continuous variables, J denotes the set of jumping variables, L denotes the set of algebraic variables, H denotes the set of channels, and R denotes a recursion definition. A recursion definition is a partial function from recursion variables to process terms. Syntactically, a recursive process definition is denoted by a set of pairs $\{X_0 \mapsto p_0, \ldots, X_m \mapsto p_m\}$, where X_i denotes a recursion variable and p_i the process term defining it.

To ensure that the variables, channels and recursion variables occurring in χ processes are defined, each χ process $\langle p, \sigma, (C, J, L, H, R) \rangle$ must satisfy the following requirements:

- All variables occurring free in p or in the range of R must be either in the domain of σ, in set L, or in case of dotted variables $\dot{x}$, their undotted counterparts x must be in C.
- All channels occurring free in p or in the range of R must be in H.
- All recursion variables occurring free in p or in the range of R must be in the domain of R.

- The predefined variable time must be in the domain of σ, and not in any of the sets C, J, and L.

- Finally, continuous variables must have a value: $C \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, jumping variables must be defined: $J \subseteq (\mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}) \cup L$, and algebraic variables, recursion variables and the other variables must be disjoint: $\mathrm{dom}(\sigma) \cap L = \emptyset$ and $(\mathrm{dom}(\sigma) \cup L) \cap \mathrm{dom}(R) = \emptyset$.

2.2 Informal semantics of processes

The behavior of χ processes is defined in terms of actions and delays. Actions define instantaneous changes, where time does not change, to the values of variables. Delays involve the passing of time, where for all variables their trajectory as a function of time is defined. The valuation σ and the environment E, together define the variables that exist in the χ process and the variable classes to which they belong.

The variables are grouped into different classes with respect to the delay behavior and action behavior. With respect to the delay behavior, the variables are divided into the following classes:

- The discrete variables, the values of which remain constant while delaying.

- The continuous variables, the values of which change according to an absolutely continuous function[1] of time while delaying. The values of continuous variables are further restricted by delay predicates, that are usually in the form of differential algebraic equations.

- The dotted continuous variables, the values of which change according to an integrable, possibly discontinuous function of time while delaying. The relation between the dotted continuous variables and the continuous variables is explained in Section 3.3.2.

- The algebraic variables, that behave in a similar way as continuous variables. The differences are that algebraic variables may change according to a discontinuous function of time, and algebraic variables are not allowed to occur as dotted variables.

- The predefined variable 'time', that denotes the current time.

With respect to the action behavior, the variables are divided into two classes:

[1] A function $f(x)$ is *continuous* at $x \in X$ provided that for all $\varepsilon > 0$, there exists $\delta > 0$ so that $|x - y| \leq \delta$ implies $|f(x) - f(y)| \leq \varepsilon$. Roughly speaking, for single-valued functions this means that we can draw the graph of the function without taking the pencil of the paper. The class of absolutely continuous functions consists of continuous functions which are differentiable almost everywhere in Lebesgue sense. This class includes the differentiable functions.

- The non-jumping variables, the values of which by default do not change in actions. The changes of non-jumping variables need to be explicitly specified.

- The jumping variables, the values of which by default can jump to arbitrary values in actions. The values after jumping can be restricted by means of action predicates, send and receive process terms, or delay predicates (equations).

The discrete and continuous variable classes can be divided into jumping and non-jumping versions. For the other classes, such a division is not possible: the dotted continuous and algebraic variables are by definition jumping with respect to the action behavior, and the predefined variable time is by definition non-jumping.

Further explanation on the semantics of the behavior of the different classes of variables is found in Section 3.3.1 on the action predicate, in Section 3.3.2 on the delay predicate, in Section 3.3.3 on the send and receive process terms, and in Section 3.4.6 on parallel composition.

The valuation σ captures the values of those variables that are relevant for determining the future behaviors of a process. The domain of the valuation σ in a χ process $\langle p, \sigma, E\rangle$ consists of the discrete variables, the continuous variables, and the predefined non-jumping variable time. The dotted continuous variables and the algebraic variables are not included in the domain of σ, because their values depend only on the process term p, possibly together with the values of the other variables. The values of the dotted continuous and algebraic variables are included in the so called 'extended valuation'. This extended valuation is required, among others, to ensure consistency of χ processes.

The consistency requirement enforces constraints on χ processes comparable to invariants in hybrid automata. Informally, in χ, the delay predicates (equations) must always hold. Consistency ensures that in $x := 1 \parallel y = x$, assuming that y is a jumping variable, the values of x and y are 1 after assigning 1 to x, independently of the initial value of y. Consistency also ensures that inconsistent processes cannot be reached, e.g. in $x := 1 \parallel x = 2$, the assignment to x cannot be executed. In fact, in χ, only consistent processes can perform action or delay transitions, and the result of an action or delay transition is always a consistent process.

Consistency is related to extended valuations in the following way: a χ process $\langle p, \sigma, E\rangle$ is consistent with extended valuation ξ, where ξ is the valuation σ extended with the (valuation for the) algebraic and dotted variables as defined by environment E, if the delay predicates u in p and the predicates u of signal emission operators in p hold when evaluated in extended valuation ξ.

For a χ process $\langle p, \sigma, (C, J, L, H, R)\rangle$, the combination of the variable classes for the delay and action behavior leads to the following classes of variables:

- The set of discrete variables D is $\mathrm{dom}(\sigma) \setminus (C \cup \{\mathsf{time}\})$,
  - the set of non-jumping discrete variables is $D \setminus J$,
  - the set of jumping discrete variables is $D \cap J$.

- The set of continuous variables is C,
  - the set of non-jumping continuous variables is $C \setminus J$,
  - the set of jumping continuous variables is $C \cap J$.
- The set of (jumping) dotted continuous variables is $\dot{C}$, where $\dot{C}$ denotes the set$\{\dot{x} | x \in C\}$.
- The set of (jumping) algebraic variables is L.
- The predefined (non-jumping) variable denoting the current time is time.

Note that it is possible to have $D \cap J \neq \emptyset$ and $L \cap J \neq \emptyset$. Such jumping discrete or jumping algebraic variables can occur as an artefact of the parallel composition of a send and a receive process term, where the receive process term assigns the received value to a discrete or algebraic variable, see Sections 3.3.3 and 3.4.6. From a modeling perspective, discrete and algebraic variables are in principle never explicitly declared as jumping. Discrete variables are not declared as jumping, because their value is not determined by equations, and therefore their values need not change when the value of a variable occurring in an equation changes due to an action. Algebraic variables are not declared as jumping, because they are by definition jumping. In fact, there is no difference between the behavior of an algebraic variable that is in set J and one that is not in the set.

Consider, for example, the process $\langle\, n := 1 \parallel y = n,\ \{n \mapsto 0, \mathsf{time} \mapsto 0\},\ (\emptyset, \emptyset, \{y\}, \emptyset, \emptyset)\,\rangle$ consisting of the discrete variable n, the predefined variable time, the algebraic variable y, and no continuous variables. Initially, the value of n equals 0, and thus the value of y equals 0. After the assignment of 1 to n, the equation $y = n$ should still hold, and thus the value of y jumps to 1. The process terms and operators used in this model, and their informal semantics are discussed in the Sections 2.3 and 2.4.

2.3 Syntax of process terms

Process terms $p \in P$ (without $p_{\text{ext}} \in P_{\text{ext}}$, see the table below) are the 'core' elements of the χ formalism. In Section 2.5, the syntax of χ process terms is extended with process terms P_{ext} to ensure better readability of χ models. The semantics of those process terms is defined in terms of the core process terms given in this section.

The set of process terms P is defined by the following grammar for the process terms $p \in P$:

$$
\begin{array}{rl}
p ::= & W : r \gg l_{\text{a}} \ \mid\ u \ \mid\ \delta \ \mid\ \bot \\
\mid & [p] \ \mid\ u \curvearrowright p \ \mid\ p;\, p \ \mid\ b \rightarrow p \ \mid\ p \,[\!]\, p \\
\mid & p \parallel p \ \mid\ h \,!!\, \mathbf{e}_n \ \mid\ h \,??\, \mathbf{x}_n \ \mid\ \partial_A(p) \ \mid\ \upsilon_H(p) \\
\mid & X \ \mid\ \iota_{J^+}(p) \\
\mid & [\![_{\text{V}}\, \sigma_\bot, C, L\ \text{`}|\text{'}\, p\, ]\!] \ \mid\ [\![_{\text{H}}\, H\ \text{`}|\text{'}\, p\,]\!] \ \mid\ [\![_{\text{R}}\, R\ \text{`}|\text{'}\, p\,]\!] \\
\mid & p_{\text{ext}}
\end{array}
$$

Here, r is a predicate over variables (including the variable time), dotted continuous variables, and '$^-$' superscripted variables (including the dotted variables, e.g. $\dot{x}$). The action label l_a is taken from a given set A_label which at least contains the special action label τ representing the internal or silent step. Furthermore, u and b are both predicates over variables (including the variable time) and dotted continuous variables; $\mathbf{e}_n$ denotes the expressions $e_1, \ldots, e_n$, and $\mathbf{x}_n$ denotes the (non-dotted) variables $x_1, \ldots, x_n$ such that $\mathsf{time} \notin \{\mathbf{x}_n\}$. For $n = 0$, $h\,!!\,\mathbf{e}_n$ and $h\,??\,\mathbf{x}_n$ can be written $h\,!!$ and $h\,??$, respectively, where h is a channel. Finally, A is a set of actions, H is a set of channels, X is a recursion variable, R is a recursion definition as defined in Section 2.1, W, J^+, C, L are sets of (non-dotted) variables such that $\mathsf{time} \notin W$ and $\mathsf{time} \notin J^+$, and $\sigma_\perp$ is a valuation that also allows the undefined 'value' $\perp$. It is specified as $\{x_0 \mapsto c_0, \ldots, x_n \mapsto c_n\}$, where x_i denotes a variable and c_i a value or $\perp$.

As is common practice in mathematics, the comma in predicates denotes conjunction. E.g. u_1, u_2 denotes the predicate $u_1 \wedge u_2$. Also, both $e_1 \leq \dot{x} \leq e_2$ and $\dot{x} \in [e_1, e_2]$ can be used instead of $\dot{x} \geq e_1$, $\dot{x} \leq e_2$, and likewise for strict inequalities and open intervals.

The operators are listed in descending order of their binding strength as follows $\{\curvearrowright, \rightarrow\}$, ; , $\{\|, [\![\}$. The operators inside the braces have equal binding strength. In addition, operators of equal binding strength associate to the right, and parentheses may be used to group expressions. For example, p; q; r means p; (q; r). An informal, concise explanation of this syntax is given below. Section 3 gives a more detailed account of their meaning.

2.4 Informal semantics of process terms

Strictly speaking, a χ process term p cannot perform actions nor delays. Only the χ process $\langle p, \sigma, E\rangle$, that is obtained by adding a valuation and an environment to p, can, in principle, perform actions and delays. Therefore, when we informally refer to a process term that performs actions or delays, we refer to the process term together with a valuation and environment.

2.4.1 Manipulating the values of variables

In χ, there are several classes of variables, and there are several means to change the value of a variable, depending on the class of variable. The main means for changing the value of a variable are the action predicate, for instantaneous changes, and the delay predicate, for the changes of variables over time.

Action predicates An instantaneous change of the value of a discrete or continuous variable in χ is always connected to the execution of an action. In action predicates, the action is represented by a label. Other types of action are related to communication, which is treated below in the paragraph on parallelism. *Action predicate* $W : r \gg l_\mathrm{a}$ denotes instantaneous changes to the variables from set W, by means of an action labeled l_a, such that predicate r is satisfied. The predefined global variable time cannot be assigned. The non-jumping

variables that are not mentioned in W remain unchanged, and the jumping variables, dotted continuous variables, and algebraic variables may obtain 'arbitrary' values, provided that the predicate r is satisfied and the process remains consistent.

A '$^-$' superscripted occurrence of a variable refers to the value of the variable in the extended valuation prior to execution of the action predicate, and a normal (un-superscripted) occurrence of a variable refers to the value of that variable in the extended valuation that results from the execution of the action predicate. A predicate r is satisfied if evaluating the '$^-$' superscripted variables in the original extended valuation and evaluating the normal occurrences of the variables in the obtained extended valuation means that the predicate is true. The reason to use an extended valuation for evaluating action predicate r is that in such predicates also algebraic and dotted continuous variables may be used. Note that it can be the case that different instantaneous changes satisfy the predicate, this may result in non-determinism.

Note that the (multi-)assignment is not a primitive in χ, as for example in [BK02]. This is because action predicates are more expressive than assignments. An assignment can be expressed as an action predicate (see Section 2.5.2), but not the other way around. Consider for example the action predicate $\{x\} : x \in [0, 1] \gg \tau$, that changes the value of x to a value in the interval $[0, 1]$. Also, the predicate of an action predicate may consist of a conjunction of implicit equations, e.g. $\{\mathbf{x}\} : f_1(\mathbf{x}^-, \mathbf{x}) = 0 \wedge \ldots \wedge f_n(\mathbf{x}^-, \mathbf{x}) = 0 \gg \tau$. The solution of such a system of equations, if present, need not always be expressible in an explicit form. The system may also have multiple solutions.

Delay predicates In principle, continuous and algebraic variables change arbitrarily over time when delaying, although, depending on the class of the variable, they may have to respect some continuity requirements, see Section 3.3.2 for more details. A *delay predicate* u, usually in the form of a differential algebraic equation, restricts the allowed behavior of the continuous and algebraic variables in such a way that the value of the predicate remains true over time. Delay predicates in the form of $x \geq e$, where x is a variable, e an expression, and instead of $\geq$, also $\leq, >, <$ can be used, are comparable to invariants in hybrid automata.

2.4.2 Deadlock and inconsistency

In χ, only consistent processes can do action or delay transitions, and the result of an action or delay transition is always a consistent process. Some process terms are consistent for certain valuations and inconsistent for other valuations. E.g. the delay predicate process term $x \geq 0$ is consistent for all values of x greater or equal to zero, and inconsistent otherwise. There are also process terms that are consistent or inconsistent for all valuations. The *inconsistent process term* $\perp$ is inconsistent for all valuations. It cannot perform any transition.

The *deadlock process term* δ cannot perform actions or delays. It is however consistent. Both process terms are needed for the specification of properties only.

2.4.3 Any delay operator

Besides the specification of delay by means of delay predicates, arbitrary delay can be described by means of the *any delay operator* $[p]$. The resulting behavior is such that arbitrary delays are allowed. When $[p]$ delays, p remains unchanged and its delay behavior is ignored. The action behavior of p remains unchanged in $[p]$.

2.4.4 Signal emission

Signal emission operator process term $u \curvearrowright p$ behaves as p for those extended valuations where u holds. The process term is inconsistent with extended valuations for which u does not hold.

2.4.5 Sequential composition

The *sequential composition* of process terms p and q behaves as process term p until p terminates, and then continues to behave as process term q.

2.4.6 Conditional

The *guarded process term* $b \rightarrow p$ can do whatever actions p can do under the condition that the guard b evaluates to true using the current extended valuation. The guarded process term can delay according to p under the condition that for the intermediate extended valuations during the delay, the guard b holds. The guarded process term can perform arbitrary delays under the condition that for the intermediate valuations during the delay, possibly excluding the first and last valuation, the guard b does not hold.

2.4.7 Choice

The *alternative composition operator* $[]$ allows a non-deterministic choice between different actions of a process. With respect to time behavior, the participants in the alternative composition have to synchronize. This means that the trajectories of the variables have to be agreed upon by both participants. This means that $[]$ is a strong time-deterministic choice operator.

2.4.8 Parallelism

Parallelism can be specified by means of the *parallel composition operator* $\|$. Parallel processes interact by means of shared variables or by means of synchronous point-to-point communication/synchronization via a channel. Channels are denoted as labels (identifiers). The parallel composition $p \parallel q$ synchronizes the time behavior of p and q, interleaves the action behavior (including the instantaneous changes of variables) of p and q, and synchronizes matching send and receive actions. The synchronization of time behavior

means that only the time behaviors that are allowed by both p and q are allowed by their parallel composition. The consistent equation semantics of χ enforces that actions by p (or q) are allowed only if the values of the variables before and after the actions are consistent with the other process term q (or p). This means, among others, that the delay predicates of q must hold before and after execution of an action by p.

By means of the *send process term* $h\,!!\,e_1, \ldots, e_n$, for $n \geq 1$, the values of expressions $e_1, \ldots, e_n$ (evaluated w.r.t. the extended valuation) are sent via channel h. For $n = 0$, this reduces to $h\,!!$ and nothing is sent via the channel. By means of the *receive process term* $h\,??\,x_1, \ldots, x_n$, for $n \geq 1$, values for $x_1, \ldots, x_n$ are received from channel h. We assume that all variables in the sequence $\mathbf{x}_n$ are syntactically different: $x_i \equiv x_j \implies i = j$. For $n = 0$, this reduces to $h\,??$, and nothing is received via the channel. Communication in χ is the sending of values by one parallel process via a channel to another parallel process, where the received values (if any) are stored in variables. For communication, the acts of sending and receiving (values) have to take place in different parallel processes at the same moment in time. In case no values are sent and received, we refer to synchronization instead of communication.

In order to be able to model open systems (i.e. systems that interface with the environment), it is necessary not to enforce communication via the external channels of the model (e.g. the channels that send or receive from the environment). For communication via internal channels, however, the communication of matching send and receive actions, often is not only an option, but an obligation. In such models, the separate occurrence of the send action and the receive action via an internal channel is undesired. The *encapsulation operator* ∂_A, where $A \subseteq \mathcal{A} \setminus \{\tau\}$ is a set of actions ($\mathcal{A}$ is the set of all possible actions and τ is the predefined internal action), is introduced to block the actions from the set A. In order to assure that, for internal channels, only the synchronous execution of matching send and receive actions takes place, one can simply put all send and receive actions via internal channels in the set A.

In principle the channels in χ are non-urgent. This means that communication does not necessarily take place as soon as possible. In order to describe also urgent channels, the *urgent communication operator* $\upsilon_H(p)$, where $H \subseteq \mathcal{H}$ is a set of channel labels, ensures that p can only delay in case no communication or synchronization of send and receive actions via a channel from H is possible.

Note that a different kind of urgency can be achieved by means of undelayable process terms. The χ semantics ensures that actions of undelayable process terms have priority over delays. For example in $\dot{x} = 1 \parallel x := 1$ and $\dot{x} = 1 \,[\!]\, x := 1$, the assignment cannot delay. Therefore, it must be executed before a delay is possible. Also in $h\,!! \parallel \dot{x} = 1$, or $h\,!! \parallel h\,??$, or $h\,!! \parallel [h\,??]$, the parallel composition cannot delay because $h\,!!$ cannot delay. Therefore, a send action must be executed before a delay may be possible. Process term $[h\,!!] \parallel [h\,??]$, however, can do a communication action (or send or receive action), but it can also delay. To enforce the synchronization, the encapsulation operator is used; to enforce this as soon as possible, the urgent communication operator is used: $\upsilon_{\{h\}}([h\,!!] \parallel [h\,??])$.

2.4.9 Recursive definitions

Process term X denotes a recursion variable (identifier) that is defined either in the environment of the process, or in a recursion scope operator process term $[\![_{\mathrm{R}} \ldots \mid p \,]\!]$, see below. Among others, it is used to model repetition. Recursion variable X can do whatever the process term of its definition can do.

2.4.10 Jump enabling operator

Jump enabling operator $\iota_{J^+}(p)$, where J^+ denotes a set of variables, is used to (re)define the variables in set J^+ as jumping variables.

2.4.11 Hierarchical modeling

Thus far, it has been assumed that all variables that are allowed to occur in a χ process term are either declared in the valuation or in the environment (in the set L). To support the hierarchical modeling of systems, it is convenient to allow local declarations of variables. For this purpose, the *variable scope operator* process term $[\![_{\mathrm{V}} \sigma_\perp, C, L \mid p \,]\!]$ is introduced, where $\sigma_\perp$ denotes a valuation of local variables, where values may be undefined ($\perp$), C denotes a set of local (non-jumping) continuous variables, and L denotes a set of local algebraic variables. The set of local discrete variables is $\mathrm{dom}(\sigma_\perp) \setminus C$. We assume $C \subseteq \mathrm{dom}(\sigma_\perp)$, $\mathrm{dom}(\sigma_\perp) \cap L = \emptyset$, and $C \cap L = \emptyset$. It is allowed that the local variables have been declared on a more global level already. Any occurrence of a variable from $\mathrm{dom}(\sigma_\perp) \cup \dot{C} \cup L$ in process term p refers to the local variable and not to any more global declaration of the same variable name.

For similar purposes, local channels can be declared by means of a *channel scope* process term $[\![_{\mathrm{H}} H \mid p \,]\!]$, and local recursive definitions by means of a *recursion scope* process term $[\![_{\mathrm{R}} R \mid p \,]\!]$. The channel scope process term $[\![_{\mathrm{H}} H \mid p \,]\!]$ is used to declare the channels from the set $H \subseteq \mathcal{H}$ to be local. Communication actions via those local channels are abstracted from (replaced by internal action τ), and the separate send and receive actions via local channels are blocked. The recursion scope process term $[\![_{\mathrm{R}} R \mid p \,]\!]$ is used to declare local recursion definitions $R \subseteq \mathcal{R}$ (see Section 3.1 for the definition of $\mathcal{R}$).

2.5 Syntactic extensions

For many of the χ processes, process terms and operators introduced before, there is additional, more user-friendly syntax available, the so-called syntactic extensions. In this section, all of these syntactic extensions are expressed in terms of the syntax introduced in the previous sections.

2.5.1 Processes

Notation

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s_1, \ldots, s_k \\
,\ \mathsf{cont}\ x_1, \ldots, x_n \\
,\ \mathsf{alg}\ z_1, \ldots, z_m \\
,\ \mathsf{chan}\ h_1, \ldots, h_l \\
,\ i \\
,\ X_1 \mapsto p_1, \ldots, X_r \mapsto p_r \\
\mid\ p \\
\rangle,
\end{array}
$$

where $s_1, \ldots, s_k$ denote the discrete variables, $x_1, \ldots, x_n$ denote the non-jumping continuous variables, $z_1, \ldots, z_m$ denote the algebraic variables, $h_1, \ldots, h_l$ denote the urgent channels, i denotes an initialization predicate that restricts the allowed values of the variables initially, $X_1 \mapsto p_1, \ldots, X_r \mapsto p_r$ denote the recursion definitions, and p is a process term, is an abbreviation for the set of χ processes defined by:

$$
\begin{array}{l}
\langle\ \partial_{A_{\mathrm{ia}}}(\upsilon_{\{h_1,\ldots,h_l\}}((i \wedge \mathsf{time} = 0) \curvearrowright p)) \\
,\ \sigma_{sxt} \\
,\ (\ \{x_1, \ldots, x_n\} \\
\quad ,\ \emptyset \\
\quad ,\ \{z_1, \ldots, z_m\} \\
\quad ,\ \{h_1, \ldots, h_l\} \\
\quad ,\ \{X_1 \mapsto p_1, \ldots, X_r \mapsto p_r\} \\
\quad) \\
\rangle,
\end{array}
$$

namely for each valuation σ_{sxt}, with $\mathrm{dom}(\sigma_{sxt}) = \{s_1, \ldots, s_k, x_1, \ldots, x_n, \mathsf{time}\}$, a separate χ process. In the χ process, A_{ia} represents the internal send and receive actions via channels $h_1, \ldots, h_l$.

In the notation defined above, it is required that the discrete, continuous, and algebraic variables are all different. Besides the declared variables, the existence of the predefined reserved global variable time which denotes the current time, the value of which is initially zero, is assumed. This variable cannot be declared. It can only be used in expressions in process term p, or in $p_1, \ldots, p_r$.

As a shorthand, the keyword preceding variables of a certain type is omitted when there are no variables of that type, and the keyword chan is omitted when there are no channel declarations. Also the initialization predicate i and the recursive definitions $X_1 \mapsto p_1, \ldots, X_r \mapsto p_r$ may be omitted, indicating a predicate that always holds and an empty list of recursive definitions, respectively.

2.5.2 Process terms

For many of the core process terms introduced before, there is additional, more user-friendly syntax available. The set of process terms P_{ext} is defined by the following grammar for the process terms $p_{\mathrm{ext}} \in P_{\mathrm{ext}}$ and $p \in P$:

$$
\begin{array}{rcl}
p_{\mathrm{ext}} & ::= & \mathrm{skip} \mid \mathbf{x}_n := \mathbf{e}_n \mid h\,!\,\mathbf{e}_n \mid h\,?\,\mathbf{x}_n \\
& \mid & \Delta_d(p) \mid \Delta d \mid *p \mid b \xrightarrow{*} p \\
& \mid & (\mathsf{jump}\ \mathbf{y}_m\ \text{`}|\text{'}p) \\
& \mid & [\![\ \mathsf{disc}\ \mathbf{s}_k, \mathsf{cont}\ \mathbf{x}_n, \mathsf{alg}\ \mathbf{z}_l, \mathsf{chan}\ \mathbf{h}_m, i, L_R\ \text{`}|\text{'}p\ ]\!] \\
& \mid & l_{\mathrm{p}}(\mathbf{x}_k,\ \mathbf{h}_m,\ \mathbf{e}_n)
\end{array}
$$

The operators of p and p_{ext} are listed in descending order of their binding strength as follows: $\{*,\ \xrightarrow{*}, \curvearrowright,\ \rightarrow\},\ ;\ ,\ \{\|, [\!]\}$.

Skip Process term skip is an abbreviation for an action predicate that can perform an internal action (τ), such that only the jumping variables can change.

$$\boxed{\mathrm{skip} \triangleq \emptyset : \mathrm{true} \gg \tau}$$

Multi-assignment Multi-assignment $\mathbf{x}_n := \mathbf{e}_n$ for $n \geq 1$ is an abbreviation for an internal action that changes variables $x_1, \ldots, x_n$ to the values of expressions $e_1, \ldots, e_n$, respectively. For $n = 1$, this gives an assignment $x := e$.

$$\boxed{\mathbf{x}_n := \mathbf{e}_n \triangleq \{\mathbf{x}_n\} : x_1 = e_1^- \wedge \cdots \wedge x_n = e_n^- \gg \tau}$$

Here e^- denotes the result of replacing all variables x_i in e by their '$-$' superscripted version x_i^-. For example, process term $x := 2x + yz$ is defined as $\{x\} : x = 2x^- + y^- z^- \gg \tau$, and process term $x, y := x + y, x - y$ is defined as $\{x, y\} : (x = x^- + y^-) \wedge (y = x^- - y^-) \gg \tau$.

Delayable send and receive Process terms $h\,!\,\mathbf{e}_n$, and $h\,?\,\mathbf{x}_n$ are the respective delayable counterparts of $h\,!!\,\mathbf{e}_n$ and $h\,??\,\mathbf{x}_n$. They are defined by means of the any delay operator $[p]$, which adds arbitrary delay behavior to p.

$$\boxed{h\,!\,\mathbf{e}_n \triangleq [h\,!!\,\mathbf{e}_n] \qquad h\,?\,\mathbf{x}_n \triangleq [h\,??\,\mathbf{x}_n]}$$

Delay operators By means of the delay operator $\Delta_d(p)$, a process term is forced to delay for the amount of time units specified by the value of numerical expression d, and then proceeds as p. The abbreviation Δd denotes a process term that first delays for d time units, and then terminates by means of an internal action τ. The fact that process term Δd terminates by means of an action ensures that time-outs enforce a choice in alternative composition. The value of expression d is evaluated at the first delay or action by Δd.

$$\boxed{\begin{array}{lcl} \Delta_d(p) & \triangleq & [\![_{\mathrm{V}}\ \{t \mapsto \bot\}, \emptyset, \emptyset \mid t = \mathsf{time} + d \curvearrowright \mathsf{time} \geq t \rightarrow p\ ]\!] \\ \Delta d & \triangleq & \Delta_d(\mathrm{skip}) \end{array}}$$

In the definition of $\Delta_d(p)$, t denotes a fresh variable, not occurring free in p. Delays are only defined for non-negative values of d. Therefore, we assume that the value of d in the extended valuation is non-negative.

Repetition operators Process term $*p$ represents the infinite repetition of process term p. Guarded repetition $b \overset{*}{\rightarrow} p$ can be interpreted as 'while b do p'.

$$\begin{array}{lcl} *p & \triangleq & [\![_{\mathrm{R}}\, \{X \mapsto p;\ X\} \mid X\,]\!] \\ b \overset{*}{\rightarrow} p & \triangleq & [\![_{\mathrm{R}}\, \{X \mapsto b \rightarrow \mathrm{skip};\ p;\ X \,[\!]\, \neg b \rightarrow \mathrm{skip}\} \mid X\,]\!] \end{array}$$

In the definition of $*p$ and $b \overset{*}{\rightarrow} p$, recursion variable X denotes a fresh recursion variable not occurring free in p.

Jump enabling operator Jump enabling operator $(\mathsf{jump}\ \mathbf{y}_m \mid p)$, where $\mathbf{y}_m$ denotes a comma separated list of variables, is used to redefine the variables $\mathbf{y}_m$ as jumping variables.

$$(\mathsf{jump}\ \mathbf{y}_m \mid p) \quad \triangleq \quad \iota_{\{\mathbf{y}_m\}}(p)$$

Modeling scope operator The modeling scope operator process term

$$[\![\ \mathsf{disc}\ \mathbf{s}_k, \mathsf{cont}\ \mathbf{x}_n, \mathsf{alg}\ \mathbf{z}_l, \mathsf{chan}\ \mathbf{h}_m, i, L_R\ \text{`}|\text{'}p\ ]\!]$$

is used to declare a scope consisting of local discrete variables $s_1, \ldots, s_k$, local (non-jumping) continuous variables $x_1, \ldots, x_n$, local algebraic variables $z_1, \ldots, z_l$, local channels $h_1, \ldots, h_m$, initialization predicate i, and local recursion definition list L_R. The variables all have to be different.

$$\begin{array}{l} [\![\ \mathsf{disc}\ \mathbf{s}_k \\ \quad ,\ \mathsf{cont}\ \mathbf{x}_n \\ \quad ,\ \mathsf{alg}\ \mathbf{z}_l \\ \quad ,\ \mathsf{chan}\ \mathbf{h}_m \\ \quad ,\ i \\ \quad ,\ L_R \\ \quad \mid\ p \\ ]\!] \end{array} \quad \triangleq \quad \begin{array}{l} [\![_{\mathrm{V}}\, \sigma_{sx} \\ \quad ,\ \{x_1, \ldots, x_n\} \\ \quad ,\ \{z_1, \ldots, z_l\} \\ \quad \mid\ [\![_{\mathrm{H}}\, \{h_1, \ldots, h_m\} \\ \qquad \mid\ \upsilon_{\{h_1,\ldots,h_m\}}([\![_{\mathrm{R}}\, \{L_R\} \mid i \curvearrowright p\ ]\!]) \\ \qquad]\!] \\ ]\!] \end{array}$$

Here L_R denotes the recursion definitions $X_1 \mapsto p_1, \ldots, X_r \mapsto p_r$, σ_{sx} denotes a valuation with $\mathrm{dom}(\sigma_{sx}) = \{s_1, \ldots, s_k,\ x_1, \ldots, x_n\}$, and σ_{sx} is undefined for all elements from its domain: $\forall_{v \in \mathrm{dom}(\sigma_{sx})}\, \sigma_{sx}(v) = \bot$.

In a similar way as defined for χ processes, the keyword preceding variables of a certain type is omitted when there are no variables of that type, and the keyword chan is omitted when there are no local channel declarations. Also the initialization predicate i and the recursion definitions may be omitted, indicating a predicate that always holds and an empty list of recursion definitions, respectively.

Process instantiation Process instantiation process term $l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$, where l_{p} denotes a process label, enables (re)-use of a process definition. A process definition is specified once, but it can be instantiated many times, possibly with different parameters: external variables $\mathbf{x}_k$, external channels $\mathbf{h}_m$, and expressions $\mathbf{e}_n$.

Chi specifications in which process instantiations $l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$ are used have the following structure:

$$
\begin{array}{l}
pd_1 \\
\vdots \\
pd_j \\
\\
\langle\ \mathsf{disc}\ \ldots,\ \mathsf{cont}\ \ldots,\ \mathsf{alg}\ \ldots,\ \mathsf{chan}\ \ldots,\ i,\ L_R \mid q\ \rangle,
\end{array}
$$

where for each process instantiation $l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$ occurring in process term q, a matching process definition pd_j of the form

$$l_{\mathrm{p}}(\mathsf{ext}\ \mathbf{x}'_k, \mathsf{chan}\ \mathbf{h}'_m, \mathsf{val}\ \mathbf{v}_n) = p$$

must be present among the j process definitions $pd_1 \ldots pd_j$. Here l_{p} denotes a process label, $\mathbf{x}_k$ denotes the 'actual external' variables $x_1, \ldots, x_k$, $\mathbf{h}_m$ denotes the 'actual external' channels $h_1, \ldots, h_m$, $\mathbf{e}_n$ denotes the expressions $e_1, \ldots, e_n$, $\mathbf{x}'_k$ denotes the 'formal external' variables $x'_1, \ldots, x'_k$, $\mathbf{h}'_m$ denotes the 'formal external' channels $h'_1, \ldots, h'_m$, and $\mathbf{v}_n$ denotes the 'value parameters' $v_1, \ldots, v_n$.

The only free variables and free channels that are allowed in process term p are the formal external variables $\mathbf{x}'_k$, the formal external channels $\mathbf{h}'_m$, and the value parameters $\mathbf{v}_n$. We assume that the formal external variables $\mathbf{x}'_k$ and the value parameters $\mathbf{v}_n$ are different.

Formally, the syntactic translation of process instantiation

$$l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$$

with corresponding process definition

$$l_{\mathrm{p}}(\mathsf{ext}\ \mathbf{x}'_k, \mathsf{chan}\ \mathbf{h}'_m, \mathsf{val}\ \mathbf{v}_n) = p$$

is given by

$$
\begin{array}{l}
[\![_{\mathrm{V}}\ \{v_1 \mapsto \bot, \ldots, v_n \mapsto \bot\},\ \emptyset,\ \emptyset \\
\mid\ \mathbf{v}_n = \mathbf{w}_n \curvearrowright p \\
]\!]\ [\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n / \mathbf{x}'_k, \mathbf{h}'_m, \mathbf{w}_n].
\end{array}
$$

Notation $q[\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n / \mathbf{x}'_k, \mathbf{h}'_m, \mathbf{w}_n]$ denotes the process term obtained from $q \in P$ by substitution of the (free) variables $\mathbf{x}'_k$ by $\mathbf{x}_k$, of the (free) channels $\mathbf{h}'_m$ by $\mathbf{h}_m$, and of the (free) variables $\mathbf{w}_n$ by expressions $\mathbf{e}_n$.

The variables $\mathbf{w}_n$ are assumed to be fresh with respect to $\mathbf{x}'_k$ and $\mathbf{v}_n$. The substitution is defined in such a way that no variables from $\mathbf{x}_k$ or $\mathbf{e}_n$, and no channels from $\mathbf{h}_m$ become

bound. If substitution would cause new bindings, the local variable or local channel that a variable or channel from $\mathbf{x}_k$, $\mathbf{e}_n$, or $\mathbf{h}_m$ would become bound to, is renamed into a fresh variable or fresh channel before the substitution takes place.

The translation declares the value parameters $\mathbf{v}_n$ as local discrete variables with initial values $\mathbf{e}_n$. By convention, however, process term p normally does not change the values of these variables.

2.6 Data types

The χ formalism is statically strongly typed. Besides the classification of variables as defined before, all variables have a type. The type of a variable defines the allowed values of the variable and the allowed operations on the variable. The atomic types are nat (natural numbers, including zero), int (integers), real (real-valued numbers), bool (booleans), string (strings), and enum (enumerations). Type constructors operate on existing types to create structured types. The χ formalism defines type constructors to create sets, lists, array tuples, record tuples, dictionaries, functions, and distributions (for stochastic models). Channels also have a type that indicates the type of data that is communicated via the channel. Pure synchronization channels, that do not communicate data, are of the predefined type void. The χ type system is strictly enforced in the χ tools. However, since the type system is not formalized, it is omitted from the specifications in this paper.

CHAPTER

THREE

Semantics of the Chi formalism

This section presents the structured operational semantics (SOS [Plo81]) of χ. It associates a hybrid transition system [CRH02] with a χ process.

3.1 General description of the SOS

The main purpose of SOS is to define the behavior of χ processes at a certain chosen level of abstraction. The meaning of a χ process depends on the values of the variables and on the environment. A set $\mathcal{V}$ of variables, and a set $\mathcal{H}$ of channel labels are assumed. The values of the variables at a specific moment in time are captured by means of a valuation, i.e., a partial function from the variables to the set of values Λ (containing at least the booleans $\mathbb{B}$ and the reals $\mathbb{R}$). The set of all valuations is denoted Σ: $\Sigma = \mathcal{V} \mapsto \Lambda$, where notation $\mathcal{V} \mapsto \Lambda$ denotes the set of all partial functions from $\mathcal{V}$ to Λ. We assume $\sigma \in \Sigma$ for all χ processes $\langle p, \sigma, E \rangle$. Extended valuations also include the values of dotted continuous variables and the algebraic variables. The set of all extended valuations is denoted $\dot{\Sigma}$: $\dot{\Sigma} = (\mathcal{V} \cup \dot{\mathcal{V}}) \mapsto \Lambda$, where $\dot{\mathcal{V}}$ denotes the set of all dotted variables. The set $T = \mathbb{R}_{\geq 0}$ is used to represent points in time. The set of environments $\mathcal{E}$ is defined as $\mathcal{E} = \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\mathcal{H}) \times \mathcal{R}$, where $\mathcal{P}(\mathcal{V})$ denotes the powerset of variables, $\mathcal{P}(\mathcal{H})$ denotes the powerset of channels, and $\mathcal{R} = \mathcal{X} \mapsto p$ denotes the set of all partial functions of recursion variables $\mathcal{X}$ to process terms p.

The SOS is chosen to represent the following:

1. Discrete behavior by means of action transitions:

   (a) $\_ \xrightarrow{\cdot} \_ \subseteq (P \times \Sigma \times \mathcal{E}) \times (\dot{\Sigma} \times \mathcal{A} \times \dot{\Sigma}) \times (P \times \Sigma \times \mathcal{E})$, where $\mathcal{A}$ denotes the set of actions, and is defined as $\mathcal{A} = A_{\text{label}} \cup A_{\text{com}}$. The set of action labels A_{label} includes at least the pre-defined internal action τ. The set of communication actions A_{com} is defined as $A_{\text{com}} = \{\text{isa}(h, cs), \text{ira}(h, cs, W), \text{ca}(h, cs) \mid h \in \mathcal{H}, cs \in \Lambda^*, W \subseteq \mathcal{V}\}$, where isa, ira, and ca denote action labels for the internal send action, the internal receive action, and the communication action respectively, $h \in \mathcal{H}$ denotes a channel, $cs \in \Lambda^*$ denotes a list $[c_1, \ldots, c_n]$ of values, and W denotes a set of variables. The intuition of an action transition $\langle p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle p', \sigma', E' \rangle$ is that the process $\langle p, \sigma, E \rangle$ executes the discrete action $a \in \mathcal{A}$ with

extended valuations ξ and ξ' and thereby transforms into the process $\langle p', \sigma', E' \rangle$, where σ' and E' denote the accompanying valuation and environment of the process term p', respectively, after the discrete action a is executed.

(b) $\_ \xrightarrow{\cdot} \langle \checkmark, \_, \_ \rangle \subseteq (P \times \Sigma \times \mathcal{E}) \times (\dot{\Sigma} \times \mathcal{A} \times \dot{\Sigma}) \times (\Sigma \times \mathcal{E})$. The intuition of a (termination) transition $\langle p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E' \rangle$ is that the process $\langle p, \sigma, E \rangle$ executes the discrete action a with extended valuations ξ and ξ' and thereby transforms into the terminated process $\langle \checkmark, \sigma', E' \rangle$.

2. Continuous behavior by means of time transitions: $\_ \xmapsto{\cdot} \_ \subseteq (P \times \Sigma \times \mathcal{E}) \times (T \times (T \mapsto \dot{\Sigma})) \times (P \times \Sigma \times \mathcal{E})$. The intuition of a time transition $\langle p, \sigma, E \rangle \xmapsto{t, \rho} \langle p', \sigma', E' \rangle$ is that during the time transition, the extended valuation at each time-point $s \in [0, t]$ is given by $\rho(s)$. At the end-point t, the resulting process is $\langle p', \sigma', E' \rangle$.

3. Consistency by means of a predicate: $\_ \overset{\cdot}{\rightsquigarrow} \subseteq (P \times \Sigma \times \mathcal{E}) \times \dot{\Sigma}$. The intuition of the consistency predicate $\langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$ is that the process term p is consistent with the extended valuation ξ in environment E.

The following properties of the semantics can be found in Section 3.5:

- For all transitions, the domain of the valuation σ equals the domain of valuation σ', and environment E equals environment E', i.e. the environment is never changed in a transition.

- For all action transitions $\langle p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E' \rangle$ and $\langle p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle p', \sigma', E' \rangle$: $\mathrm{dom}(\sigma) \subseteq \mathrm{dom}(\xi)$, $\mathrm{dom}(\xi) = \mathrm{dom}(\xi')$, extended valuation ξ restricted to $\mathrm{dom}(\sigma)$ equals valuation σ, and extended ξ' restricted to $\mathrm{dom}(\sigma')$ equals valuation σ'.

- For all time transitions $\langle p, \sigma, E \rangle \xmapsto{t, \rho} \langle p, \sigma', E' \rangle$: $\mathrm{dom}(\rho) = [0, t]$, and for all variables $x \in \mathrm{dom}(\sigma)$, the value in the resulting valuation $\sigma'(x)$ equals the value of the variable in the end-point of the trajectory $\rho(t)(x)$.

- For all consistency predicates $\langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$: extended valuation ξ restricted to $\mathrm{dom}(\sigma)$ equals valuation σ.

The relations and predicates mentioned above are defined through so-called deduction rules. A deduction rule is of the form $\frac{H}{r}$, where H is a number of hypotheses separated by commas and r is the result of the rule. The result of a deduction rule can be derived if all of its hypotheses are derived. In case the set of hypotheses is empty, the deduction rule is called an axiom.

In order to increase the readability of the χ deduction rules, some additional abbreviations are used. Notation $E \Vdash \langle p, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle q, \sigma' \rangle$, where $q \in P \cup \{\checkmark\}$ is an abbreviation for $\langle p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle q, \sigma', E \rangle$, notation $E \Vdash \langle p, \sigma \rangle \xmapsto{t, \rho} \langle q, \sigma' \rangle$ is an abbreviation for $\langle p, \sigma, E \rangle \xmapsto{t, \rho} \langle q, \sigma', E \rangle$, and notation $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ is an abbreviation for $\langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$.

Notation $E \Vdash f_1, \ldots, f_n$, where f_i represents one of the previously defined transition relations (of the forms $\langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$ or $\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle q, \sigma'\rangle$ or $\langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$) is an abbreviation for $E \Vdash f_1, \ldots, E \Vdash f_n$.

Notation

$$\frac{E' \Vdash \langle p_1, \sigma_1\rangle \xrightarrow{\xi_1,a_1,\xi'_1} \left\langle \begin{matrix} q_{1_1} \\ \vdots \\ q_{1_n} \end{matrix} , \sigma'_1 \right\rangle, \ldots, \langle p_m, \sigma_m\rangle \xrightarrow{\xi_m,a_m,\xi'_m} \left\langle \begin{matrix} q_{m_1} \\ \vdots \\ q_{m_n} \end{matrix} , \sigma'_m \right\rangle,\ C}{E \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,b,\xi'} \left\langle \begin{matrix} s_1 \\ \vdots \\ s_n \end{matrix} , \sigma' \right\rangle}$$

where $q_{j_i}, s_i \in P \cup \{\checkmark\}$, $p_i, r \in P$, and C denotes an optional hypothesis that must be satisfied in the deduction rule, is an abbreviation for the following rules (one for each i):

$$\frac{E' \Vdash \langle p_1, \sigma_1\rangle \xrightarrow{\xi_1,a_1,\xi'_1} \langle q_{1i}, \sigma'_1\rangle, \ldots, \langle p_m, \sigma_m\rangle \xrightarrow{\xi_m,a_m,\xi'_m} \langle q_{mi}, \sigma'_m\rangle,\ C}{E \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,b,\xi'} \langle s_i, \sigma'\rangle}$$

The notation $\frac{H}{R}$, where R is a number of results separated by commas, is an abbreviation for a set of deduction rules of the form $\frac{H}{r}$; one for each $r \in R$, and notation ${}_E\frac{H}{r}$ is an abbreviation for $\frac{E \Vdash H}{E \Vdash r}$.

Furthermore, notation $\langle p, \sigma, E\rangle \overset{\mathrm{ca}(h,*)}{\nrightarrow}$ denotes $(\nexists_{\xi,cs,\xi',p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p', \sigma', E'\rangle) \wedge (\nexists_{\xi,cs,\xi',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle \checkmark, \sigma', E'\rangle)$, and notation $\langle p, \sigma, E\rangle \xrightarrow{\alpha} \langle p', \sigma', E'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$ for some ξ, a, and ξ'.

3.2 Notations and mathematical definitions

Notations $f \in M \to G$ and $g \in M \mapsto G$ define function f, $\mathrm{dom}(f) = M$, and partial (or f induced) function g, $\mathrm{dom}(g) \subseteq M$, both with range G.

3.2.1 Operators on functions

Based on [LSV03], the following definitions of operators $\restriction$, $\cup$, and $\downarrow$ applied on functions are used. If f is a function, $\mathrm{dom}(f)$ and $\mathrm{range}(f)$ denote the domain and range of f, respectively. If S is a set, $f \restriction S$ denotes the restriction of f to S, that is, the function g with $\mathrm{dom}(g) = \mathrm{dom}(f) \cap S$, such that $g(c) = f(c)$ for each $c \in \mathrm{dom}(g)$.

If f and g are functions with $\mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset$, then $f \cup g$ denotes the unique function h with $\mathrm{dom}(h) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ satisfying the condition: for each $c \in \mathrm{dom}(h)$, if $c \in \mathrm{dom}(f)$ then $h(c) = f(c)$, and $h(c) = g(c)$ otherwise.

If f is a function whose range is a set of functions and S is a set, then $f \downarrow S$ denotes the function g with $\mathrm{dom}(g) = \mathrm{dom}(f)$ such that $g(c) = f(c) \restriction S$ for each $c \in \mathrm{dom}(g)$. If f is a

function whose range is a set of functions, all of which have a particular element d in their domain, then $f \downarrow d$ denotes the function g with $\mathrm{dom}(g) = \mathrm{dom}(f)$ such that $g(c) = f(c)(d)$ for each $c \in \mathrm{dom}(g)$.

3.2.2 Notations

Let $x \in \mathcal{V}$ be a variable, $S, C, L \subseteq \mathcal{V}$ be sets of variables, $\sigma \in \Sigma$ be a valuation, e be an expression over variables and constants, and $t \in T$ be a time-point, then the following notations are defined:

- $\sigma(x)$ denotes the value of variable x in valuation σ. We use the similar notation $\sigma(e)$ to denote the value of expression e evaluated in valuation σ.
- $\dot{S}$ denotes the set of dotted variables $\{\dot{x} \mid x \in S\}$.
- $\xi^{\dot{C}L} \in (\dot{C} \cup L) \to \Lambda$ denotes an arbitrary valuation with domain $\dot{C} \cup L$.
- ξ_σ is an abbreviation for $\xi \restriction \mathrm{dom}(\sigma)$.
- Function $\Xi \in (\Sigma \times \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\mathcal{V})) \to \mathcal{P}(\dot{\Sigma})$ returns a set of extended valuations, given a valuation, a set of continuous variables, a set of jumping variables, and a set of algebraic variables. Formally, function Ξ is defined as:

  $$\Xi(\sigma, C, J, L) = \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C} \cup L,\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\, \xi(x) = \sigma(x)\}.$$

  The values of the variables in ξ are defined as follows: the values of the variables in $\mathrm{dom}(\sigma) \setminus J$ are given by σ. The jumping variables J, the dotted variables $\dot{C}$ and the algebraic variables L have arbitrary values.
- $\Omega_{\sigma E t}$, where environment E denotes the tuple (C, J, L, H, R), is an abbreviation for $\Omega_{FG}(\sigma, C, L, \mathrm{true}, t)$. Here, Ω_{FG} is the solution function as defined in Section 3.3.2.
- ρ_σ is an abbreviation for $\rho \downarrow \mathrm{dom}(\sigma)$.

3.3 Deduction rules for atomic process terms

3.3.1 Action predicate

Action predicate process term $W : r \gg l_\mathrm{a}$ denotes instantaneous changes to the variables from set W, by means of an action labeled $l_\mathrm{a} \in A_\mathrm{label}$, such that predicate r over variables from the domains of the extended valuations ξ^- and ξ' is satisfied, see Rule 1, where $\xi, \xi' \in (\mathrm{dom}(\sigma) \cup \dot{C} \cup L) \to \Lambda$, and ξ^- is defined below.

The values of the variables from $\mathrm{dom}(\sigma)$ in ξ are given by σ. The dotted variables $\dot{C}$ and the algebraic variables L in ξ can in principle take any value ($\xi = \sigma \cup \xi^{\dot{C}L}$) as long as the action predicate r is satisfied ($\xi^- \cup \xi' \models r$). Variables occurring with a '$-$' superscript

in r are evaluated in ξ^-, which denotes the extended valuation with the values of variables before the discrete change. Extended valuation ξ^- is defined as $\mathrm{dom}(\xi^-) = \{x^- \mid x \in \mathrm{dom}(\xi)\}$, and $\xi^-(x^-) = \xi(x)$. For extended valuation ξ', the values of the discrete and the non-jumping variables $(\mathrm{dom}(\sigma) \setminus (J \cup W))$ are given by σ. The jumping variables J, the variables from set W, the dotted variables $\dot{C}$ and the algebraic variables L are allowed to change such that the action predicate is satisfied. Since there are no time transition rules defined for action predicates, this means that action predicates cannot perform any time transitions.

Rule 2 states that action predicates are always consistent with any extended valuation $\sigma \cup \xi^{\dot{C}L}$ with respect to σ in any environment E.

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J \cup W, L),\ \xi^- \cup \xi' \models r}{(C, J, L, H, R) \Vdash \langle W : r \gg l_{\mathrm{a}}, \sigma\rangle \xrightarrow{\xi,\ l_{\mathrm{a}},\ \xi'} \langle \checkmark, \xi'_\sigma\rangle}\ 1$$

$$\frac{}{(C, J, L, H, R) \Vdash \langle W : r \gg l_{\mathrm{a}}, \sigma\rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 2$$

3.3.2 Delay predicate

Delay predicate u is a predicate over variables and dotted continuous variables.

$$\frac{\rho \in \Omega_{FG}(\sigma, C, L, u, t)}{(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \xmapsto{t,\rho} \langle u, \rho_\sigma(t)\rangle}\ 3 \qquad \frac{\sigma \cup \xi^{\dot{C}L} \models u}{(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 4$$

Function $\Omega_{FG} \in \Sigma \times \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\mathcal{V}) \times U \times T \to \mathcal{P}(T \mapsto \dot{\Sigma})$, where U denotes the set of all predicates over $\mathcal{V}$ and $\dot{\mathcal{V}}$, returns a set of trajectories from time to an extended valuation for the variables and dotted variables, given a valuation representing the current values of the discrete and continuous variables, the set of continuous variables, the set of algebraic variables, a delay predicate and a time point that denotes the duration of the trajectory. Formally, the function Ω_{FG} is defined as:

$$\begin{array}{ll}
\Omega_{FG}(\sigma, C, L, u, t) = & \\
\{\ \rho & \\
\mid\ \rho \in [0, t] \to ((\mathrm{dom}(\sigma) \cup \dot{C} \cup L) \to \Lambda) & \\
,\ t \geq 0 & \\
,\ \forall s \in [0, t] : & \rho(s) \models u \\
,\ \forall x \in \mathrm{dom}(\sigma) \setminus (\{\mathsf{time}\} \cup C) : & \rho \downarrow x \text{ is a constant function.} \\
,\ \forall x \in \mathrm{dom}(\sigma) : & (\rho \downarrow x)(0) = \sigma(x) \\
,\ \forall x \in L : & \rho \downarrow x \in F \\
,\ \forall x \in C : & \rho \downarrow \dot{x} \text{ is an integrable function in the} \\
 & \text{Lesbesgue sense.} \\
,\ \forall s \in [0, t], x \in C : & (\rho \downarrow x)(s) = (\rho \downarrow x)(0) + \int_0^s (\rho \downarrow \dot{x})(s')ds' \\
,\ \forall x \in C : & (\rho \downarrow x, \rho \downarrow \dot{x}) \in G \\
,\ \forall s \in [0, t] : & \rho(s)(\mathsf{time}) = \sigma(\mathsf{time}) + s \\
\} &
\end{array}$$

The trajectory ρ is a function from the time interval $[0, t]$, where $t \geq 0$, to a valuation, where the domain of each valuation consists of all variables and dotted continuous variables. The trajectory ρ satisfies the predicate u for all time points of its domain ($\forall_{s \in [0,t]}\, \rho(s) \models u$). The trajectory of each discrete variable $x \in \mathrm{dom}(\sigma) \setminus (\{\mathsf{time}\} \cup C)$ is restricted to a constant function. The initial value (starting-point) of the trajectory of each discrete and continuous variable equals the value of that variable in σ ($\forall_{x \in \mathrm{dom}(\sigma)}\, (\rho \downarrow x)(0) = \sigma(x)$).

The trajectories of the algebraic variables ($\rho \downarrow x$ for $x \in L$) are required to be functions of type F. This set of functions is a parameter of the solution concept of χ. The definition of the trajectory as $\rho \in [0, t] \to ((\mathrm{dom}(\sigma) \cup \dot{C} \cup L) \to \Lambda)$ ensures that $\forall_{x \in L}\, (\rho \downarrow x) \in [0, t] \to \Lambda$. Having the set F as a parameter of the solution concept allows us to restrict F to, for instance, the set of piecewise constant functions, if this would be required for certain properties to hold.

The trajectories of the dotted variables are required to be integrable. This ensures that the integral $\int_0^s (\rho \downarrow \dot{x})(s')ds'$ is defined. The relation between the trajectory of a continuous variable x and the trajectory of its 'derivative' $\dot{x}$ is given by the Caratheodory solution concept [Fil88]: $(\rho \downarrow x)(s) = (\rho \downarrow x)(0) + \int_0^s (\rho \downarrow \dot{x})(s')ds'$. Note that this integral relation can hold only for those continuous variables for which $\rho \downarrow x$ is an absolutely continuous function. Thus the solution function Ω_{FG} restricts the trajectory $\rho \downarrow x$ of every continuous variable x to an absolutely continuous function, but it does allow a non-smooth trajectory for a continuous variable in the case that the trajectory of its 'derivative' $\rho \downarrow \dot{x}$ is non-smooth or even discontinuous, as in, for example, $\langle\ \mathsf{cont}\ y,\ y = 0 \mid \dot{y} = \mathrm{step}(\mathsf{time} - 1)\ \rangle$, where $\mathrm{step}(x)$ equals 0 for $x \leq 0$ and 1 for $x > 0$.

The disadvantage of the Caratheodory solution concept is that it introduces spurious solutions in a higher index system such as $\langle\ \mathsf{cont}\ y,\ \mathsf{alg}\ z \mid y = \mathsf{time}, z = \dot{y}\ \rangle$. Here, one could argue that the trajectory for z should be the constant function 1. The Caratheodory solution concept, however, allows trajectories for z that are 1, except for discontinuity points, where any other value is allowed. Such spurious discontinuities in $\rho \downarrow \dot{x}$, in the case that the trajectory of a differential variable x is smooth, and thus $\rho \downarrow x$ is differentiable, on some interval I, can be prevented in two ways.

First, by changing the model to $\langle\ \mathsf{cont}\ y, z \mid y = \mathsf{time}, z = \dot{y}\ \rangle$. Defining z as a continuous variable requires its trajectory to be (absolutely) continuous.

Second, by restricting the solution concept. This can be done by restricting set G in the requirement $\forall_{x \in C}\, (\rho \downarrow x, \rho \downarrow \dot{x}) \in G$, where G is a parameter of the χ solution concept. Defining $G = \{(f, f') \mid \forall_{I \subseteq \mathrm{dom}(f)}\, f$ is differentiable on $I \Rightarrow f' \restriction I$ is the derivative function of $f \restriction I\}$, where I denotes some interval, requires the solution function $\rho \downarrow \dot{x}$ for the dotted variable $\dot{x}$ to be indeed the derivative function of the solution function $\rho \downarrow x$ for the differential variable x, for all intervals where $\rho \downarrow x$ is differentiable. This prevents spurious discontinuities from occurring in higher index systems as discussed above. The disadvantage of this set G is that for instance the delay predicate $(\mathsf{time} = 1 \Rightarrow \dot{x} = 1) \wedge (\mathsf{time} \neq 1 \Rightarrow \dot{x} = 0)$ has no solution for x ($\rho \downarrow x$) on the interval $[0, t]$, for $t > 1$, starting from a valuation in which $\mathsf{time} = 0$. A constant function of time for x with domain $[0, t]$ for $t > 0$, which is a solution for $G = \{(f, f') \mid \mathrm{true}\}$, is not a solution for the restricted version of G defined above, because the derivative function (here $\rho \downarrow \dot{x}$) of a constant function (here $\rho \downarrow x$) is

always zero, and therefore the valuation at time point 1 ($\rho(1)$) does not satisfy the delay predicate.

The properties derived in Section 3.5.2 are valid for all parameters F and G. For the translation of a hybrid automaton to χ as defined in Section 5.2, differentiable functions are assumed for the trajectories of the continuous variables: $G = \{(f, f') | f$ is differentiable, and f' is the derivative function of $f\}$. In this way, the semantics of the χ translation corresponds to the semantics of the hybrid automaton. For the examples in Section 4, differentiability would be too strong a restriction. Therefore, piecewise continuous functions for the trajectories of the algebraic and dotted variables are assumed: $F = \{f \mid f$ is a piecewise continuous function$\}$, $G = \{(f, f') \mid f'$ is a piecewise continuous function$\}$. There is no fundamental reason for this choice. Another possibility would have been not to define additional restrictions: $F = \{f \mid \text{true}\}$, $G = \{(f, f') \mid \text{true}\}$. For a model with just one solution such as: $\langle\ \mathsf{cont}\ x,\ \mathsf{alg}\ y \mid \dot{x} = y,\ y = \text{step}(\mathsf{time} - 1)\ \rangle$, the solution is the same for both cases of F and G. For a model that allows infinitely many solutions, such as $\langle\ \mathsf{cont}\ x,\ \mathsf{alg}\ y \mid \text{true}\ \rangle$, there would obviously be a difference.

In some deduction rules describing delay behavior, abbreviation $\Omega_{\sigma Et}$, which denotes $\Omega_{FG}(\sigma, C, L, \text{true}, t)$, is used as a hypothesis. The true predicate does not restrict t and the trajectory ρ other than by means of the default restrictions. Among others, the discrete variables remain constant, and the trajectory of each continuous variable is an absolutely continuous function that starts with the value of the continuous variable in σ.

3.3.3 Send and receive

Send and receive process terms $h\,!!\,\mathbf{e}_n$ and $h\,??\,\mathbf{x}_n$ denote undelayable sending of expression $\mathbf{e}_n$ via channel h, and undelayable receiving of information via channel h into variable(s) $\mathbf{x}_n$, respectively.

The values of expressions $e_1, \ldots, e_n$ which are sent via channel h are evaluated in extended valuation ξ, see Rule 5, where $\mathbf{e}_n$ denotes $e_1, \ldots, e_n$, $[\xi(\mathbf{e}_n)]$ denotes the list of values $[\xi(e_1), \ldots, \xi(e_n)]$ for $n \geq 1$, and $\xi(e)$ denotes the value of expression e for extended valuation ξ. The case that n equals 0, represents the case where nothing is sent via the channel, and $\mathbf{e}_0$ and $[\xi(\mathbf{e}_0)]$ denote an empty expression and an empty list, respectively. For $n \geq 1$, the receive process term $h\,??\,x_1, \ldots, x_n$ can receive the list of values $[c_1, \ldots, c_n]$, see Rule 6, where $\mathbf{x}_n$ denotes $x_1, \ldots, x_n$, $\{\mathbf{x}_n\}$ denotes the set $\{x_1, \ldots, x_n\}$, $[\mathbf{c}_n]$ denotes the list of values $[c_1, \ldots, c_n]$, and $\xi'(\mathbf{x}_n) = \mathbf{c}_n$ is an abbreviation for $\xi'(x_1) = c_1, \ldots, \xi'(x_n) = c_n$. For $n = 0$, nothing is received, so that $\mathbf{x}_0$ and $\mathbf{c}_0$ are empty, and $\xi'(\mathbf{x}_0) = \mathbf{c}_0$ always holds.

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J, L)}{(C, J, L, H, R) \Vdash \langle h\,!!\,\mathbf{e}_n, \sigma\rangle \xrightarrow{\xi,\ \text{isa}(h, [\xi(\mathbf{e}_n)]),\ \xi'} \langle \checkmark, \xi'_\sigma\rangle}\ 5$$

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L),\ \xi'(\mathbf{x}_n) = \mathbf{c}_n}{(C, J, L, H, R) \Vdash \langle h\,??\,\mathbf{x}_n, \sigma\rangle \xrightarrow{\xi,\ \text{ira}(h, [\mathbf{c}_n], \{\mathbf{x}_n\}),\ \xi'} \langle \checkmark, \xi'_\sigma\rangle}\ 6$$

$$\frac{}{(C, J, L, H, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} 7 \qquad \frac{}{(C, J, L, H, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} 8$$

3.3.4 Deadlock and inconsistent process term

Process term δ cannot perform any action transitions, nor time transitions. It is, however, consistent for arbitrary extended valuations $\sigma \cup \xi^{\dot{C}L}$.

$$\frac{}{(C, J, L, H, R) \Vdash \langle \delta, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} 9$$

There are no rules for the inconsistent process term $\bot$. Therefore, it cannot do actions transition, nor time transitions, and it is inconsistent for all valuations and environments. Process term $\bot$ originates from the process algebra with propositional signals $\mathrm{ACP}_{\mathrm{ps}}$ ([BB97]).

3.4 Deduction rules for operators

3.4.1 Any delay operator

The any delay operator $[p]$ allows arbitrary time transitions, that need to satisfy only the general solution function ($\Omega_{\sigma Et}$) requirements, regardless of the time transitions allowed by p (see Rule 11). A requirement can be the trajectory of each continuous variable is an absolutely continuous function that starts with the value of the continuous variable in σ. This means the values of continuous variables can change according to such a trajectory. The any delay operator does not affect the action behavior of p (see Rule 10). Process term $[p]$ is consistent with any extended valuation $\sigma \cup \xi^{\dot{C}L}$ with respect to σ in any environment E (see Rule 12).

$$E \;\; \frac{\langle p, \sigma \rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle}{\langle [p], \sigma \rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle} 10 \qquad E \;\; \frac{\rho \in \Omega_{\sigma Et}}{\langle [p], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [p], \rho_\sigma(t) \rangle} 11$$

$$\frac{}{(C, J, L, H, R) \Vdash \langle [p], \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} 12$$

3.4.2 Signal emission operator

The signal emission operator $u \curvearrowright p$ ensures that p starts its behavior from an extended valuation ξ in which initialization predicate u is satisfied. This operator was inspired by the signal emission operator from the process algebra with propositional signals $\mathrm{ACP}_{\mathrm{ps}}$ [BB97], which was also used in [BM05].

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix},\sigma'\rangle,\ \xi \models u}{\langle u \curvearrowright p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix},\sigma'\rangle}\ 13 \qquad E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\ \rho(0) \models u}{\langle\ u \curvearrowright p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}\ 14$$

$$E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}\ ,\ \xi \models u}{\langle\ u \curvearrowright p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 15$$

3.4.3 Sequential composition operator

The sequential composition of process terms p and q behaves as process term p until p terminates, and then continues to behave as process term q. When p terminates, its right-hand extended valuation ξ' must be consistent with q (see Rule 16).

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark,\sigma'\rangle,\ \langle q,\sigma'\rangle \overset{\xi'}{\rightsquigarrow}}{\langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q,\sigma'\rangle}\ 16 \qquad E\ \frac{\langle p,\sigma\rangle \xrightarrow{\alpha} \langle p',\sigma'\rangle}{\langle p;\ q,\sigma\rangle \xrightarrow{\alpha} \langle p';\ q,\sigma'\rangle}\ 17$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}{\langle p;\ q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p';\ q,\sigma'\rangle}\ 18 \qquad E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p;\ q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 19$$

3.4.4 Guard operator

The guarded process term $b \to p$ can do whatever actions p can do under the condition that the guard evaluates to true using extended valuation ξ. Evaluating the guard in ξ ensures that when guard operators are nested with signal emission operators, actions can be executed only if all predicates of the signal emission operators and all guards hold, independently of the order. Furthermore, the values of the dotted variables and algebraic variables are defined in ξ, whereas they are not defined in σ.

The guarded process term can delay according to p under the condition that for all intermediate valuations the guard evaluates to true ($\forall_{s\in[0,t]}\, \rho(s) \models b$, see Rule 21).

The guarded process term can perform arbitrary delays under the condition that for the intermediate valuations, possibly excluding the first and last valuation, the guard does not hold ($\forall_{s\in(0,t)}\, \rho(s) \models \neg b$). This ensures that, for example, the process $\langle$ disc $x,\ x = 1 \mid$ time $\geq x \to$ skip $\rangle$ behaves as expected: it can first do a time transition of 1, such that the value of the current time time becomes 1, and thereafter it can do a τ action to the terminated process. If the condition in Rule 22 would be $\forall_{s\in[0,t]}\, \rho(s) \models \neg b$, then a time transition of 1 would be impossible. This is because the value of the guard should then also be false for the last time point of the time transition, so that the point where the value of time equals 1 could not be reached. The condition $\rho(0) \models b \Rightarrow \langle p,\sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p',\sigma'\rangle$ in Rule 22, which states that p must be able to delay for a duration of 0 if the guard is initially true, ensures that undelayable actions in p have priority over delay behavior of a

guard that is initially true and continues as false. The condition $\rho(t) \models b \Rightarrow \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ in Rule 22 requires consistency if the guard holds in the end-point of the trajectory. This ensures that it is impossible to delay to an inconsistent state.

Finally, $b \to p$ is consistent with extended valuations for which b holds and with which p is consistent (Rule 23), and with extended valuations for which b does not hold (Rule 24).

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \xi \models b}{\langle b \to p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle}\ 20 \qquad E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\ \forall_{s\in[0,t]}\,\rho(s) \models b}{\langle b \to p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p',\sigma'\rangle}\ 21$$

$$E\ \frac{\begin{array}{c}\rho \in \Omega_{\sigma E t},\ \forall_{s\in(0,t)}\,\rho(s) \models \neg b,\\ \exists_{s\in[0,t]}\,\rho(s) \models \neg b,\\ \rho(0) \models b \Rightarrow \langle p,\sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p',\sigma'\rangle,\\ \rho(t) \models b \Rightarrow \langle p,\rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}\end{array}}{\langle b \to p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p,\rho_\sigma(t)\rangle}\ 22$$

$$E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow},\ \xi \models b}{\langle b \to p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 23 \qquad \frac{\sigma \cup \xi^{\dot{C}L} \models \neg b}{(C,J,L,H,R) \Vdash \langle b \to p,\sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}}\ 24$$

3.4.5 Alternative composition operator

Applying the alternative composition operator to process terms p and q models a non-deterministic choice between p and q for action transitions. Process term p can perform action transitions only if the initial extended valuation is consistent with q, as specified in Rule 25. Consider for example the following process term: $y = 1 \,[\!]\, x := y$. This corresponds to a hybrid automaton with one location with flow predicate true, invariant $y = 1$, and an urgent outgoing edge with jump condition $x := y$. The invariant $y = 1$ ensures that the value of y equals 1 when the outgoing edge is taken.

The passage of time cannot result in making a choice between p and q, since the time transitions of the process terms p and q have to synchronize to obtain the time transition (with the same time step t and trajectory ρ) of their alternative composition as defined by Rule 26.

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \langle q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p \,[\!]\, q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \langle q \,[\!]\, p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle}\ 25$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\ \langle q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle q',\sigma'\rangle}{\langle p \,[\!]\, q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p' \,[\!]\, q',\sigma'\rangle}\ 26 \qquad E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow},\ \langle q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p \,[\!]\, q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 27$$

3.4.6 Parallel composition operator

The parallel composition of process terms p and q has as its behavior with respect to action transitions the interleaving of the behaviors of p and q (see Rule 29). Process term p can only perform action transitions from an extended valuation ξ which is consistent with q. Furthermore, the resulting extended valuation ξ' must be consistent with q (see Rule 29).

The parallel composition allows the synchronization of matching send and receive actions. A send action $\mathrm{isa}(h, cs)$ and a receive action $\mathrm{ira}(h', cs', W)$ match iff $h = h'$ and $cs = cs'$; i.e. the channels used for sending and receiving are the same, and also the values sent and the values received are identical. Furthermore, the resulting extended valuations ξ' of both the send action and the receive action have to be the same. In order to be able to receive values in variables of the same scope as the send process term, the variables of which the value changes due to the receive action are passed on to the send process term. This is achieved by means of set W on the receive action, and the addition of this set W to the set of jumping variables in the environment where the send action takes place (see Rule 28). The result of the synchronization is a communication action that is represented by $\mathrm{ca}(h, cs)$ as defined by Rule 28.

The time transitions of the process terms that are put in parallel have to synchronize in the same way as for alternative composition, see Rules 26 and 30.

$$\frac{(C, J \cup W, L, H, R) \Vdash \langle p, \sigma \rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \left\langle \begin{matrix} \checkmark \\ p' \\ \checkmark \\ p' \end{matrix}, \sigma' \right\rangle, \quad (C, J, L, H, R) \Vdash \langle q, \sigma \rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \left\langle \begin{matrix} \checkmark \\ \checkmark \\ q' \\ q' \end{matrix}, \sigma' \right\rangle}{(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \left\langle \begin{matrix} \checkmark \\ p' \\ q' \\ p' \parallel q' \end{matrix}, \sigma' \right\rangle, \quad \langle q \parallel p, \sigma \rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \left\langle \begin{matrix} \checkmark \\ p' \\ q' \\ q' \parallel p' \end{matrix}, \sigma' \right\rangle} 28$$

$$E \ \frac{\langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}, \ \langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} \checkmark \\ p' \end{matrix}, \sigma' \rangle, \ \langle q, \sigma' \rangle \overset{\xi'}{\rightsquigarrow}}{\langle p \parallel q, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} q \\ p' \parallel q \end{matrix}, \sigma' \rangle, \ \langle q \parallel p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} q \\ q \parallel p' \end{matrix}, \sigma' \rangle} 29$$

$$E \ \frac{\langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle, \ \langle q, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p' \parallel q', \sigma' \rangle} 30$$

$$E \ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow} \ , \ \langle q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p \parallel q,\sigma\rangle \overset{\xi}{\rightsquigarrow}} \ 31$$

3.4.7 Action encapsulation operator

The behavior of the action encapsulation applied to a process term, $\partial_A(p)$, is the same as the behavior of its argument with the restriction that actions from the set A ($A \subseteq \mathcal{A} \setminus \{\tau\}$) cannot be executed (see Rule 32). Action encapsulation has no effect on time transitions and consistency, as defined by Rules 33 and 34.

$$E \ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma'\rangle, \ a \notin A}{\langle \partial_A(p),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{\partial_A(p')},\sigma'\rangle} \ 32$$

$$E \ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}{\langle \partial_A(p),\sigma\rangle \overset{t,\rho}{\longmapsto} \langle \partial_A(p'),\sigma'\rangle} \ 33 \qquad E \ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle \partial_A(p),\sigma\rangle \overset{\xi}{\rightsquigarrow}} \ 34$$

3.4.8 Urgent communication operator

The urgent communication operator $\upsilon_H(p)$ gives communication actions via channels from set $H \subseteq \mathcal{H}$ a higher priority than time transitions. Action behavior and consistency are not affected by the urgent communication operator, see Rules 35 and 36. Time transitions are allowed only if at each intermediate state while delaying no communication actions via channels from H are possible.

$$E \ \frac{\langle p,\sigma\rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'},\sigma'\rangle}{\langle \upsilon_H(p),\sigma\rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{\upsilon_H(p')},\sigma'\rangle} \ 35 \qquad E \ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle \upsilon_H(p),\sigma\rangle \overset{\xi}{\rightsquigarrow}} \ 36$$

$$E \ \frac{\begin{array}{l}\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle, \ \langle p,\sigma\rangle \overset{\mathrm{ca}(h,*)}{\not\rightarrow} \ , \ \forall_{s\in[0,t)} \ (\langle p,\sigma\rangle \overset{s,\rho\restriction[0,s]}{\longmapsto} \langle p_s,\sigma_s\rangle, \\ \qquad\qquad\qquad\qquad\qquad\qquad \langle p_s,\sigma_s\rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle p',\sigma'\rangle, \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall_{h\in\mathcal{H}} \langle p_s,\sigma_s,E\rangle \overset{\mathrm{ca}(h,*)}{\not\rightarrow})\end{array}}{\langle \upsilon_H(p),\sigma\rangle \overset{t,\rho}{\longmapsto} \langle \upsilon_H(p'),\sigma'\rangle} \ 37$$

where ρ_{-s} denotes the trajectory ρ shifted left by s time-units and starting at 0: $\mathrm{dom}(\rho_{-s}) = [0, t-s]$, assuming $\mathrm{dom}(\rho) = [0,t]$, and $\forall_{t'\in\mathrm{dom}(\rho_{-s})} \, \rho_{-s}(t') = \rho(t'+s)$.

3.4.9 Recursion variable

A recursion variable process term X behaves as the process term given by $R(X)$. Here $R(X)$ is the process term that is defined for recursion variable X in function R. This is equivalent to syntactically replacing recursion variable X by its defining process term $R(X)$. Function R can be defined in the environment of the χ process directly, or by means of the recursion scope operator, see Section 3.4.13.

$$(C, J, L, H, R)\ \frac{\langle R(X), \sigma\rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\surd\\ p'\end{smallmatrix}, \sigma'\rangle}{\langle X, \sigma\rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\surd\\ p'\end{smallmatrix}, \sigma'\rangle}\ 38$$

$$(C, J, L, H, R)\ \frac{\langle R(X), \sigma\rangle \xmapsto{t,\rho} \langle p', \sigma'\rangle}{\langle X, \sigma\rangle \xmapsto{t,\rho} \langle p', \sigma'\rangle}\ 39$$

$$(C, J, L, H, R)\ \frac{\langle R(X), \sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle X, \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 40$$

3.4.10 Jump enabling operator

The jump enabling operator applied to a process term p with set J^+ ($\iota_{J^+}(p)$) behaves the same as its argument in an environment where the variables from set J^+ are jumping variables.

$$\frac{(C, J \cup J^+, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\surd\\ p'\end{smallmatrix}, \sigma'\rangle}{(C, J, L, H, R) \Vdash \langle \iota_{J^+}(p), \sigma\rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\surd\\ \iota_{J^+}(p')\end{smallmatrix}, \sigma'\rangle}\ 41$$

$$\frac{(C, J \cup J^+, L, H, R) \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle p', \sigma'\rangle}{(C, J, L, H, R) \Vdash \langle \iota_{J^+}(p), \sigma\rangle \xmapsto{t,\rho} \langle \iota_{J^+}(p'), \sigma'\rangle}\ 42$$

$$\frac{(C, J \cup J^+, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}}{(C, J, L, H, R) \Vdash \langle \iota_{J^+}(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 43$$

3.4.11 Variable scope operator

By means of the variable scope operator, local variables are introduced in a χ process. A variable scope operator process term

$$[\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p\,]\!],$$

that is used in an environment (C, J, L, H, R), with valuation σ, and where $\sigma_{\mathrm{dx}_\perp}$ denotes a local valuation that may have undefined values and that has domain $\{\mathbf{d}, \mathbf{x}\}$, $\mathbf{d}$ denotes the local discrete variables $d_1, \ldots, d_k$, $\mathbf{x}$ denotes the local (non-jumping) continuous variables $x_1, \ldots, x_n$, and $\mathbf{g}$ denotes the local algebraic variables $g_1, \ldots, g_m$, behaves as p after taking the union of the respective categories (discrete, continuous and algebraic) of local and global variables and taking the union of the local and global valuation. To ensure that all local variables are fresh with respect to the global variables, the local variables are first renamed. Thus $\mathbf{d}', \mathbf{x}', \mathbf{g}'$, in the rules below, denote fresh variables $d'_1, \ldots, d'_k$, $x'_1, \ldots, x'_n$, $g'_1, \ldots, g'_m$ with respect to $\mathrm{dom}(\sigma) \cup L \cup \{\mathbf{d}\} \cup \{\mathbf{x}\} \cup \{\mathbf{g}\}$. Notation $p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}]$ denotes the process term that is obtained by substitution of the (free) variables $\mathbf{d}, \mathbf{x}, \mathbf{g}$ in p by the fresh variables $\mathbf{d}', \mathbf{x}', \mathbf{g}'$, respectively, choosing the fresh variables $\mathbf{d}', \mathbf{x}', \mathbf{g}'$ in such a way that they remain free in p. After execution of an action or a delay transition, the local variables of the variable scope operator are renamed back to their original names. Note that the variables used in the recursion definitions R are not renamed to ensure that the bindings of these variables remain unchanged. In this way, the variables occurring in recursion definitions are bound statically, as is illustrated by the following example χ process:

$$\begin{array}{l}\langle [\![_{\mathrm{V}}\, \{n \mapsto 2\},\ \emptyset,\ \emptyset \mid X;\ z := n\ ]\!] \\ ,\ \{n \mapsto 0,\ y \mapsto 0,\ z \mapsto 0\} \\ ,\ (\emptyset, \emptyset, \emptyset, \emptyset, \{X \mapsto n := 1;\ y := n\}) \\ \rangle.\end{array}$$

The process defines the variables n, y, z that are initialized to 0, a recursion definition $X \mapsto n := 1;\ y := n$, and a variable scope operator that redefines n as a local variable that is initialized to 2. When the process term $X;\ z := n$ terminates, the value of y equals 1, and the value of z equals 2. The recursion variable X is executed in the scope of its definition.

The variable scope operator is the only operator that affects the set of continuous variables C and the set of algebraic variables L from the environment. In this way, it is ensured that the discrete, continuous, or algebraic variables in any χ process $\langle p, \sigma, E\rangle$ remain discrete, continuous, or algebraic, respectively. Continuous variables, on the other hand, can change from non-jumping continuous variables to jumping continuous variables, using the jump enabling operator (see Section 3.4.10).

The local variables are invisible outside of the scope operator. This is done by means of data abstraction. For action transitions, data abstraction takes place by restricting the extended valuations, and the valuation of the resulting process, to the global variables, and by keeping only the global variables in the set W of the internal receive actions. For time transitions, data abstraction takes place by restricting the trajectory to the global variables. In this way, all changes to local variables are removed.

Action transition abstraction function $\kappa \in \Sigma \times \mathcal{P}(\dot{\mathcal{V}}) \times \mathcal{P}(\mathcal{V}) \times \dot{\Sigma} \times \mathcal{A} \times \dot{\Sigma} \to \dot{\Sigma} \times \mathcal{A} \times \dot{\Sigma}$ is defined as follows. For arbitrary receive actions $\mathrm{ira}(h, cs, W)$:

$$\kappa_{\sigma\dot{C}L}(\xi, \mathrm{ira}(h, cs, W), \xi') = \xi_{\sigma\dot{C}L},\ \mathrm{ira}(h, cs, W \cap (\mathrm{dom}(\sigma) \cup L)),\ \xi'_{\sigma\dot{C}L},$$

and for all other actions:

$$\kappa_{\sigma\dot{C}L}(\xi, a, \xi') = \xi_{\sigma\dot{C}L}, a, \xi'_{\sigma\dot{C}L},$$

where extended valuations $\xi_{\sigma\dot{C}L}$ and $\xi'_{\sigma\dot{C}L}$ denote $\xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$, respectively. Furthermore, in the rules below, the following abbreviations are used: valuation σ'_σ denotes $\sigma' \restriction \mathrm{dom}(\sigma)$, and trajectory $\rho_{\sigma\dot{C}L}$ denotes $\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$.

Valuation $\sigma_{\mathrm{dx}_\perp} \in \{\mathbf{d}, \mathbf{x}\} \mapsto (\Lambda \cup \{\perp\})$ and valuation $\sigma_{\mathrm{d'x'}} \in \{\mathbf{d}', \mathbf{x}'\} \mapsto \Lambda$ define the same values for all (renamed) variables for which $\sigma_{\mathrm{dx}_\perp}$ is defined. For the undefined variables in $\sigma_{\mathrm{dx}_\perp}$, $\sigma_{\mathrm{d'x'}}$ has an arbitrary value: $\forall_{v \in \mathrm{dom}(\sigma_{\mathrm{dx}_\perp})} \sigma_{\mathrm{dx}_\perp}(v) \neq \perp \Rightarrow \sigma_{\mathrm{d'x'}}(v[\mathbf{d}', \mathbf{x}'/\mathbf{d}, \mathbf{x}]) = \sigma_{\mathrm{dx}_\perp}(v)$, where $v[\mathbf{d}', \mathbf{x}'/\mathbf{d}, \mathbf{x}]$ denotes the renamed version of variable v.

$$\frac{(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}} \rangle \xrightarrow{\xi, a, \xi'} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!], \sigma \rangle \xrightarrow{\kappa_{\sigma\dot{C}L}(\xi, a, \xi')} \left\langle \overset{\checkmark}{[\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid p'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}'] \,]\!]}, \sigma'_\sigma \right\rangle} \; 44$$

$$\frac{(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}} \rangle \xmapsto{t, \rho} \langle p', \sigma' \rangle}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!], \sigma \rangle \xmapsto{t, \rho_{\sigma\dot{C}L}} \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid p'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}'] \,]\!], \sigma'_\sigma \rangle} \; 45$$

$$\frac{(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}} \rangle \overset{\xi}{\rightsquigarrow}}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!], \sigma \rangle \overset{\xi_{\sigma\dot{C}L}}{\rightsquigarrow}} \; 46$$

3.4.12 Channel scope operator

By means of the channel scope operator, local channels can be introduced in a χ process. By means of action abstraction, communication actions on local channels are made invisible outside of the scope operator.

Action abstraction takes place by substituting communication actions $\mathrm{ca}(h, cs)$ using a local channel by internal τ actions (see Rule 47). The internal send and receive actions ($\mathrm{isa}(h, cs)$ and $\mathrm{ira}(h, cs, W)$) on a local channel h are blocked, because Rule 47 only specifies behavior for communication actions $\mathrm{ca}(h, cs)$. Therefore, these internal send and receive actions are not visible outside of the scope operator. Function $\mathrm{ch} \in \mathcal{A} \to \mathcal{H} \cup \{\perp\}$ extracts the channel label from an action. It is defined as $\mathrm{ch}(\mathrm{ca}(h, cs)) = h$, $\mathrm{ch}(\mathrm{isa}(h, cs)) = h$, $\mathrm{ch}(\mathrm{ira}(h, cs, W)) = h$, and $\mathrm{ch}(l_\mathrm{a}) = \perp$, where $l_\mathrm{a} \in A_{\mathrm{label}}$. Note that no renaming is applied to action a in Rule 48, because this action cannot refer to local channels.

The local channels $\mathbf{h}$ occurring in p are renamed to fresh channels $\mathbf{h}'$ in a similar way as for the local variables in the variable scope operator. Also here, in the channel scope operator, renaming does not take place in the recursion definitions R to ensure that the bindings of channels in R remain unchanged.

$$\frac{(C, J, L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle p[\mathbf{h}'/\mathbf{h}], \sigma\rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ h \in \{\mathbf{h}'\}}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid p\ ]\!], \sigma\rangle \xrightarrow{\xi, \tau, \xi'} \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid \overset{\checkmark}{p'}[\mathbf{h}/\mathbf{h}']\ ]\!], \sigma'\rangle}\ 47$$

$$\frac{(C, J, L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle p[\mathbf{h}'/\mathbf{h}], \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \mathrm{ch}(a) \notin \{\mathbf{h}'\}}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid p\ ]\!], \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid \overset{\checkmark}{p'}[\mathbf{h}/\mathbf{h}']\ ]\!], \sigma'\rangle}\ 48$$

$$\frac{(C, J, L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle p[\mathbf{h}'/\mathbf{h}], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid p\ ]\!], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid p'[\mathbf{h}/\mathbf{h}']\ ]\!], \sigma'\rangle}\ 49$$

$$\frac{(C, J, L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle p[\mathbf{h}'/\mathbf{h}], \sigma\rangle \overset{\xi}{\rightsquigarrow}}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\ \{\mathbf{h}\} \mid p\ ]\!], \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 50$$

3.4.13 Recursion scope operator

By means of the recursion scope operator, local recursion definitions are introduced in a χ process. The application of the recursion scope operator to a process term p with a 'global' valuation σ and a 'global' environment (C, J, L, H, R) behaves as p after the addition of local recursion definitions to the global recursion definitions. In the rules below, $\mathbf{X} \mapsto \mathbf{q}$ denotes the recursion definitions $X_1 \mapsto q_1, \ldots, X_r \mapsto q_r$. To prevent redefinition of recursion definitions already existing in the environment, the local recursion variables $\mathbf{X}$ are renamed to fresh variables $\mathbf{X}'$ with respect to the variables from the domain of R.

$$\frac{(C, J, L, H, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma\rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma'\rangle}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\ ]\!], \sigma\rangle \xrightarrow{\alpha} \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid \overset{\checkmark}{p'}[\mathbf{X}/\mathbf{X}']\ ]\!], \sigma'\rangle}\ 51$$

$$\frac{(C, J, L, H, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\ ]\!], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}']\ ]\!], \sigma'\rangle}\ 52$$

$$\frac{(C, J, L, H, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma\rangle \overset{\xi}{\rightsquigarrow}}{(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\ ]\!], \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 53$$

Consider, for example, the process term $[\![_{\mathrm{R}}\ X \mapsto Y, Y \mapsto x := 0 \mid [\![_{\mathrm{R}}\ Y \mapsto x := 1 \mid X\ ]\!]]\!]$. Local recursion variable Y with definition $Y \mapsto x := 1$ conflicts with the recursion variable definition $Y \mapsto x := 0$ from the outer scope. The renaming of the local variable in the rules of the recursion scope operator ensures that the process term behaves as $[\![_{\mathrm{R}}\ X \mapsto Y, Y \mapsto x := 0 \mid [\![_{\mathrm{R}}\ Z \mapsto x := 1 \mid X\ ]\!]]\!]$. Thus, the value of variable x becomes 0.

3.5 Validation of the semantics

First we consider the well-definedness of the semantics in Section 3.5.1. Then, in Section 3.5.2, some properties of the χ semantics are given. In Section 3.5.3, a notion of equivalence is defined, called stateless bisimilarity [MRG05], which is similar to the well-known notion of bisimilarity [Par81, Mil80]. It is also shown that this relation is an equivalence and a congruence for all χ operators. Some useful properties of closed χ process terms are given in Section 3.5.4. Many of these properties express intuitions about the meaning of the χ operators such as the commutativity and associativity of the alternative composition and the parallel composition operator. Other properties are introduced for the purpose of simplifying χ models. Both the examples treated in the next chapter and the properties treated in this section add to the level of confidence one has with respect to the 'correctness' of the semantics.

3.5.1 Well-definedness of the semantics

Well-definedness of the term deduction system means that the system defines a unique transition system for each closed process term. In the term deduction system of χ, negative premises are used in Rule 37 of the urgent communication operator. As a consequence it is not obvious at first sight whether the term deduction system defines a unique transition system for each closed process term. Well-definedness of the term deduction system can be obtained by providing a *stratification* [BV95, MRG05]. A stratification is a metric on formulae that, for each deduction rule of the transition system specification, does not increase from conclusion to all positive premises and strictly decreases from conclusion to negative premises.

We define the mapping that associates with every positive termination transition, action transition and positive consistency predicate the value 0 and with every positive time transition the value 1. Then, it is not hard to see that the χ deduction rules of the transition system specification are stratifiable.

3.5.2 Properties of the semantics

In this section, some useful properties about the semantics of χ are introduced. The proofs of these properties are given from in Appendix A.1 to Appendix A.6. The properties are applied in the remainder of the chapter, especially in the proofs of the properties defined in Section 3.5.4.

With the current set of deduction rules for the semantics of χ, the left-hand (ξ) and right-hand (ξ') extended valuation restricted to the domain of σ are always the same as the initial (σ) and resulting (σ') valuation of an action transition, respectively. A similar reasoning applies to the first and last valuation of a trajectory on a time transition and the initial and resulting valuation, respectively. Also note that the environment is never changed in a transition, and that the extended valuation in the consistency predicate restricted to the model variables is the same as the initial valuation.

The following lemma captures these facts.

Lemma 3.5.1 *Let p and p' be closed process terms, σ, σ' be valuations, ξ, ξ' be extended valuations, E and E' be environments, a be an action, ρ be a trajectory, and $t \in T$. Then*

$$\begin{aligned}
\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E'\rangle \quad &\Rightarrow \quad \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma' \\
&\qquad \wedge E = E', \\
\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle \quad &\Rightarrow \quad \mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma' \\
&\qquad \wedge E = E', \\
\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \quad &\Rightarrow \quad \xi_\sigma = \sigma.
\end{aligned}$$

The χ processes that can perform action or time transitions are consistent (the consistency predicate holds).

Lemma 3.5.2 *Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, ξ and ξ' be extended valuations and a be an action. Then*

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \quad \Rightarrow \quad \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow},$$

where $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ is an abbreviation for $\exists_{p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E'\rangle$.

Lemma 3.5.3 *Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, $t \in T$, and ρ be a trajectory. Then,*

$$\langle p, \sigma, E\rangle \xmapsto{t,\rho} \quad \Rightarrow \quad \langle p, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow},$$

where $\langle p, \sigma, E\rangle \xmapsto{t,\rho}$ is an abbreviation for $\exists_{p',\sigma',E'} \langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle$.

Lemma 3.5.4 *Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, ξ and ξ' be extended valuations and a be an action. Then*

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle \quad \Rightarrow \quad \langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}.$$

Lemma 3.5.5 *Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, $t \in T$, and ρ be a trajectory. Then,*

$$\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle \quad \Rightarrow \quad \langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}.$$

The following lemma shows that any variation in the set of jumping variables in the environment of a consistent χ process has no effect on the consistency predicate.

Lemma 3.5.6 *Let p be a closed process term, σ be a valuation, C, J, W, L be sets of various classes of χ variables such that J and $W \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, H be a set of channels, R be a recursion definition, and ξ be an extended valuation. Then*

$$\langle p, \sigma, (C, J, L, H, R)\rangle \overset{\xi}{\rightsquigarrow} \quad \Leftrightarrow \quad \langle p, \sigma, (C, J \cup W, L, H, R)\rangle \overset{\xi}{\rightsquigarrow}.$$

3.5.3 Stateless bisimilarity

Two closed χ process terms are considered equivalent if they have the same behavior (in the bisimulation sense) in case both are considered, from the current state, the valuation of model variables and the same environment. We also assume that the valuation (of the current state) contains at least the free occurrences of variables in the two closed χ process terms being equivalent.

Definition 3.5.1 (Stateless bisimilarity) *A stateless bisimulation relation on closed process terms is a relation $R \subseteq P \times P$ such that for all $(p, q) \in R$, the following holds:*

1. $\forall_{\sigma,E,\xi,a,\xi',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$
   $\Leftrightarrow \langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$,
2. $\forall_{\sigma,E,\xi,a,\xi',p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$
   $\Rightarrow \exists_{q'} \langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle \wedge (p', q') \in R$,
3. $\forall_{\sigma,E,\xi,a,\xi',q',\sigma',E'} \langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle$
   $\Rightarrow \exists_{p'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle \wedge (p', q') \in R$,
4. $\forall_{\sigma,E,t,\rho,p',\sigma',E'} \langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle$
   $\Rightarrow \exists_{q'} \langle q, \sigma, E\rangle \xmapsto{t,\rho} \langle q', \sigma', E'\rangle \wedge (p', q') \in R$,
5. $\forall_{\sigma,E,t,\rho,q',\sigma',E'} \langle q, \sigma, E\rangle \xmapsto{t,\rho} \langle q', \sigma', E'\rangle$
   $\Rightarrow \exists_{p'} \langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle \wedge (p', q') \in R$,
6. $\forall_{\sigma,E,\xi} \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \Leftrightarrow \langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

Two closed process terms p and q are stateless bisimilar, denoted by $p \underline{\leftrightarrow} q$, if there exists a stateless bisimulation relation R such that $(p, q) \in R$.

As a consequence of Lemma 3.5.1, the definition of stateless bisimilarity can be simplified considerably. Yet, with in mind future extensions of the χ formalism, it might well be the case that these properties of the semantics are lost. Since we would prefer not to redo all the coming proofs (in such a future), this presentation was chosen.

Stateless bisimilarity is proved to be a congruence with respect to all χ operators. As a consequence, algebraic reasoning is facilitated, since it is allowed to replace equals by equals in any context.

Theorem 3.5.1 (Congruence) *Stateless bisimilarity is a congruence with respect to all χ operators.*

PROOF. See Appendix A.7.

3.5.4 Properties of the Chi operators

In this section, some properties of the operators of χ that hold with respect to stateless bisimilarity are discussed. Most of these correspond well with our intuitions, and hence this can be considered as an additional validation of the semantics. It is not our intention to provide a complete list of such properties (complete in the sense that every equivalence between closed process terms is derivable from those properties). The proofs of the properties from this section are given from Appendix B.1 to Appendix B.8.

Proposition 3.5.1 (Any delay operator) *The following properties hold for all closed process terms $p \in P$ and predicate u:*

$[p]$	$\leftrightarrow$	$[[p]]$	$[u]$	$\leftrightarrow$	true

Multiple applications of the any delay operator are equivalent to a single application. The application of the any delay operator to a delay predicate u is equivalent to a predicate true.

Proposition 3.5.2 (Signal emission operator) *The following properties hold for all closed process terms $p \in P$ and predicates u, u':*

true $\curvearrowright p$	$\leftrightarrow$	p	$u \curvearrowright u$	$\leftrightarrow$	u
false $\curvearrowright p$	$\leftrightarrow$	$\bot$	$u \curvearrowright (u' \curvearrowright p)$	$\leftrightarrow$	$(u \wedge u') \curvearrowright p$

If a true predicate is emitted, the process term is simply executed. If falsity holds initially, the process term is inconsistent. There is no effect if a predicate u is emitted to itself. A concatenation of signal emissions leads to a signal emission with conjunction of predicates.

Proposition 3.5.3 (Alternative composition) *The following properties hold for all closed process terms p, q, $r \in P$:*

$p \parallel$ true	$\leftrightarrow$	p	$(p \parallel q) \parallel r$	$\leftrightarrow$	$p \parallel (q \parallel r)$
$p \parallel p$	$\leftrightarrow$	p	$[p \parallel q]$	$\leftrightarrow$	$[p] \parallel [q]$
$p \parallel q$	$\leftrightarrow$	$q \parallel p$			

Delay predicate true is a zero element for alternative composition.The alternative composition is idempotent, commutative and associative. The property $p \parallel \delta \leftrightarrow p$ does not hold. Consider, for example $p =$ true. Then $p \parallel \delta$ cannot perform any time transitions, while p can perform arbitrary time transitions. Property $p \parallel \delta \leftrightarrow \delta$ does not hold either. Consider, for example $p =$ skip. Then $p \parallel \delta$ can perform a τ transition, while δ cannot. The any delay operator distributes over the alternative composition.

Proposition 3.5.4 (Guard operator) *The following properties hold for closed process terms $p, q \in P$ and guard b:*

true $\rightarrow p$	$\leftrightarrow$	p	$b \rightarrow \bot$	$\leftrightarrow$	$\neg b$
false $\rightarrow p$	$\leftrightarrow$	true	$b \rightarrow (p \parallel q)$	$\leftrightarrow$	$b \rightarrow p \parallel b \rightarrow q$

If a process term is guarded by a true predicate, the process term is simply executed. In case a process term is guarded by a false predicate, process term false $\rightarrow p$ can perform any time transition, hence equals a true predicate. An inconsistent process term that is guarded by any guard is equivalent to the negation of the guard. By rewriting this property as $u \leftrightarrows \neg u \rightarrow \bot$, where the delay predicate u and guard b share the same syntax, it becomes clear that the delay predicate is not a primitive. Finally, the guard distributes over the alternative composition operator.

Proposition 3.5.5 (Sequential composition) *The following properties hold for all closed process terms $p, q, r \in P$, guard b, and predicate u:*

$\delta;\ p$	$\leftrightarrows$	δ	$b \rightarrow (p;\ q)$	$\leftrightarrows$	$(b \rightarrow p);\ q$
$(p;\ q);\ r$	$\leftrightarrows$	$p;\ (q;\ r)$	$u;\ p$	$\leftrightarrows$	u
$(p \between q);\ r$	$\leftrightarrows$	$p;\ r \between q;\ r$	$[p];\ q$	$\leftrightarrows$	$[p;\ q]$

A deadlock process term followed by some other process terms is equivalent to the deadlock process term itself since the deadlock process term does not terminate successfully, i.e., deadlock is a left-zero element for sequential composition. Sequential composition is associative. Also, a delay predicate u followed by some other process terms is equivalent to the delay predicate u itself since the delay predicate u does not terminate successfully. Alternative composition distributes over sequential composition from the left, but not from the right. A guard distributes to the left argument of a sequential composition. The any delay operator distributes to the right argument of a sequential composition.

Proposition 3.5.6 (Parallel composition) *The following properties hold for all closed process terms $p, q, r \in P$ and predicates u, u':*

$p \parallel q$	$\leftrightarrows$	$q \parallel p$	$u \parallel u'$	$\leftrightarrows$	$u \wedge u'$
$(p \parallel q) \parallel r$	$\leftrightarrows$	$p \parallel (q \parallel r)$			

Parallel composition is commutative and associative. The parallel composition of two delay predicates is the same as the conjunction of the delay predicates.

Proposition 3.5.7 (Action encapsulation operator) *The following properties hold for all closed process terms $p, q \in P$, guard b, predicate u and sets of actions A, A':*

$\partial_A(\delta)$	$\leftrightarrows$	δ	$\partial_A(p;\ q)$	$\leftrightarrows$	$\partial_A(p);\ \partial_A(q)$
$\partial_\emptyset(p)$	$\leftrightarrows$	p	$\partial_A(u)$	$\leftrightarrows$	u
$\partial_A(\partial_{A'}(p))$	$\leftrightarrows$	$\partial_{A \cup A'}(p)$	$\partial_A([p])$	$\leftrightarrows$	$[\partial_A(p)]$
$\partial_A(p \between q)$	$\leftrightarrows$	$\partial_A(p) \between \partial_A(q)$	$\partial_A(b \rightarrow p)$	$\leftrightarrows$	$b \rightarrow \partial_A(p)$

Process term δ is a zero element for the action encapsulation operator. If there are no actions to be encapsulated, the application of the action encapsulation operator to a process term p has no effect. Encapsulation of actions distributes over the alternative composition operator and the sequential composition operator. Action encapsulation has

no effect on delay predicates. Multiple applications of the action encapsulation operator are equivalent to a single application where all the actions to be encapsulated are combined using union of sets of actions. The order of the application of the any delay operator and the action encapsulation operator to a process term is irrelevant. A guard distributes over action encapsulation.

Proposition 3.5.8 (Inconsistent process) *The following properties hold for all closed process terms $p \in P$, predicate u and set of action A:*

$u \curvearrowright \bot$	$\underline{\leftrightarrow}$	$\bot$	$\bot; p$	$\underline{\leftrightarrow}$	$\bot$
$p \mathbin{[\!]} \bot$	$\underline{\leftrightarrow}$	$\bot$	skip; $\bot$	$\underline{\leftrightarrow}$	δ
$p \parallel \bot$	$\underline{\leftrightarrow}$	$\bot$	$\bot$	$\underline{\leftrightarrow}$	false
$\partial_A(\bot)$	$\underline{\leftrightarrow}$	$\bot$			

The inconsistent process term is a zero element for the signal emission operator, alternative composition, parallel composition and the action encapsulation operator. It is also a left-zero element for sequential composition. Going on as $\bot$ after performing an action transition, for example skip, is impossible. Since $\bot$ and false predicate cannot perform any transition, both process terms are equivalent.

CHAPTER

FOUR

Examples of hybrid Chi models

In this chapter, the χ formalism is illustrated by several examples taken from various application domains.

4.1 Tank controller

Figure 4.1 shows a liquid storage tank with a volume controller VC. The incoming flow Q_i can be controlled by means of a valve n. The outgoing flow is given by equation $Q_\mathrm{o} = \sqrt{V}$. The volume controller maintains the volume V of the liquid in the tank between 2 and 10.

The χ model of the volume controller and the storage tank is as follows:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ n, \mathsf{cont}\ V, \mathsf{alg}\ Q_\mathrm{i}, Q_\mathrm{o} \\
,\ n = 0, V = 10 \\
|\ \dot{V} = Q_\mathrm{i} - Q_\mathrm{o} \\
,\ Q_\mathrm{i} = n \cdot 5 \\
,\ Q_\mathrm{o} = \sqrt{V} \\
\|\ *(\ V \leq 2 \rightarrow n := 1;\ V \geq 10 \rightarrow n := 0\ ) \\
\rangle
\end{array}
$$

Initially, the volume in the tank equals 10, and the valve is closed ($n = 0$). The outgoing flow Q_o is given by equation $Q_\mathrm{o} = \sqrt{V}$. When the volume equals 2, the valve is opened ($V \leq 2 \rightarrow n := 1$). When the volume in the tank equals 10, the valve is closed again ($V \geq 10 \rightarrow n := 0$).

Figure 4.1 shows (a part of) the hybrid transition system of the tank controller. The circles represent the states, arrows $\rightarrow$ represent action transitions that are labelled as defined in Chapter 3, and arrows $\longmapsto$ represent time transitions. The labels (flows) of the time transitions are represented graphically. Constant t_f denotes the value $-5\ln(15) - 5\ln(5 - \sqrt{10}) + 5\ln(5 + \sqrt{10}) + 5\ln(23) + 5\ln(5 - \sqrt{2}) - 5\ln(5 + \sqrt{2})$.

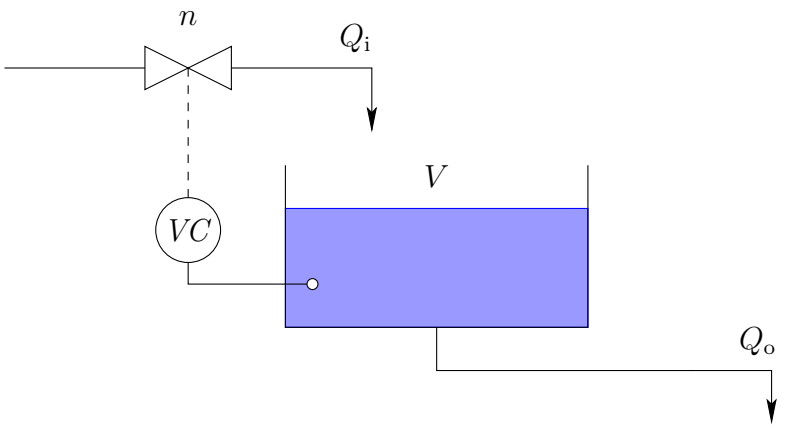

Figure 4.1: Tank controller.

4.2 Diode

An ideal diode can either block or conduct the current. When it blocks, the diode voltage $v \leq 0$, and the current $i = 0$. When it conducts, the diode voltage $v = 0$, and the current $i \geq 0$. Figure 4.3 shows the characteristics of an ideal diode.

$$
\begin{array}{l}
[\![\ \mathrm{block} \quad \mapsto i = 0,\ v \leq 0\ ]\!]\ [\mathrm{skip}];\ \mathrm{conduct} \\
,\ \mathrm{conduct} \mapsto v = 0,\ i \geq 0\ ]\!]\ [\mathrm{skip}];\ \mathrm{block} \\
|\ \mathrm{skip};\ \mathrm{block}\ ]\!]\ \mathrm{skip};\ \mathrm{conduct} \\
]\!]
\end{array}
$$

The modes of the diode are each specified by means of a mode definition for the modes block and conduct. Initially, the diode can be in either one of the two modes, depending on its environment; or more specific, depending on, among others, voltage or current sources in its environment. The alternative composition operator is used in combination with the skip internal action to select one of the two modes. The skip process term is needed because alternative composition makes a choice only by means of an action. The any delay operator in [skip] is needed because otherwise the assignment and the alternative composition would not be able to delay. If $v = 0$ and $i = 0$, the modes block and conduct can both delay. In this case, a non-deterministic choice between the two modes is made. The diode can also be specified in χ as follows:

Using χ, the process term modeling the diode is a single delay predicate:

$$(v \leq 0 \wedge i = 0) \vee (i \geq 0 \wedge v = 0)$$

The difference between the two specifications is that in the first specification, a delay transition must take place within a mode. In case of a mode switch, the delay transition is 'divided' into two separate delay transitions: one delay transition according to the first mode, until the time point where the mode switch occurs, and another delay transition

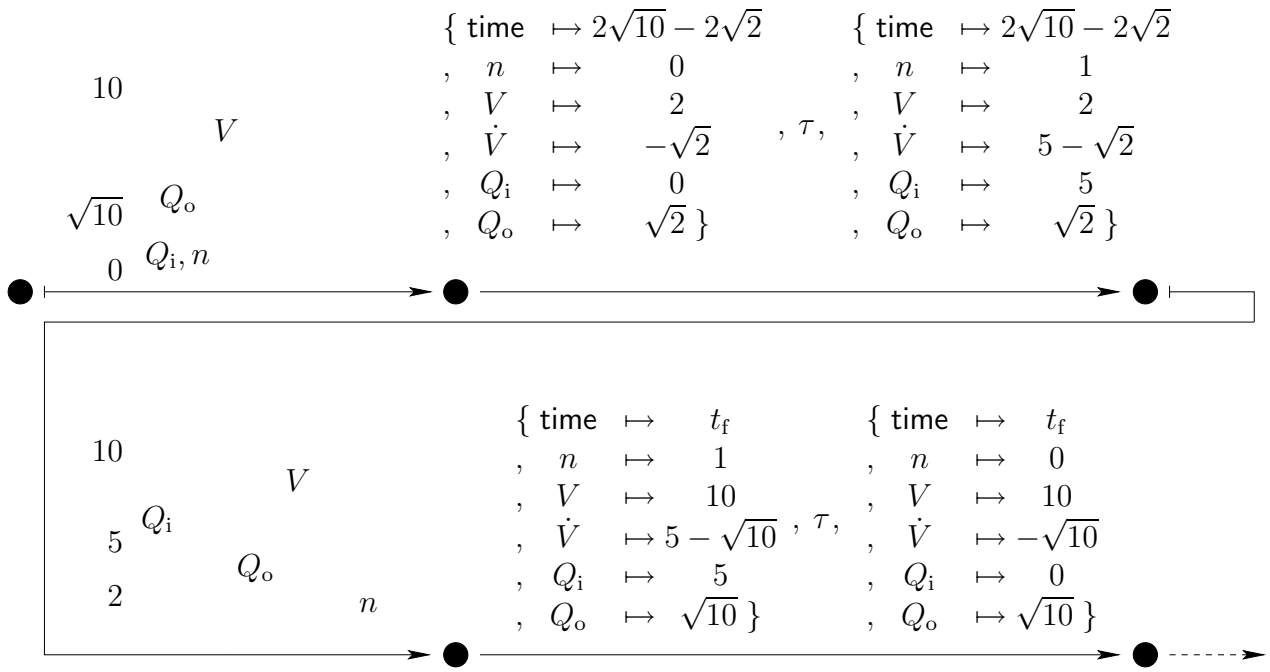

Figure 4.2: Hybrid transition system of the tanklevel controller.

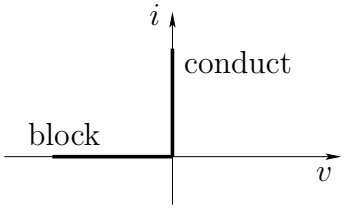

Figure 4.3: Characteristics of an ideal diode.

according to the second mode. This limits the length of the delay transitions. The second specification does not limit the length of delay transitions. Here, a 'mode switch' can occur within a delay transition. The difference between the two specifications is that in the first specification, mode switches can take place only by means of an (internal) action. Therefore, the delay transitions in each mode are interleaved with actions for mode switching. In the second specification, no actions are needed for 'mode switching': 'mode switches' can occur within a single delay transition.

4.3 Half wave rectifier circuit

Figure 4.4 shows a half wave rectifier circuit. It consists of a diode D, two resistors with resistance R_0 and R_1, respectively, a capacitor with capacity C_0, and a voltage source with voltage v_0. The model is a parallel composition of a diode D, two resistors R_0 and R_1, modeled by two process instantiation of process definition R and a capacitor C. In the χ model, symbols f, π, C_0, R_0 and R_1 denote constants.

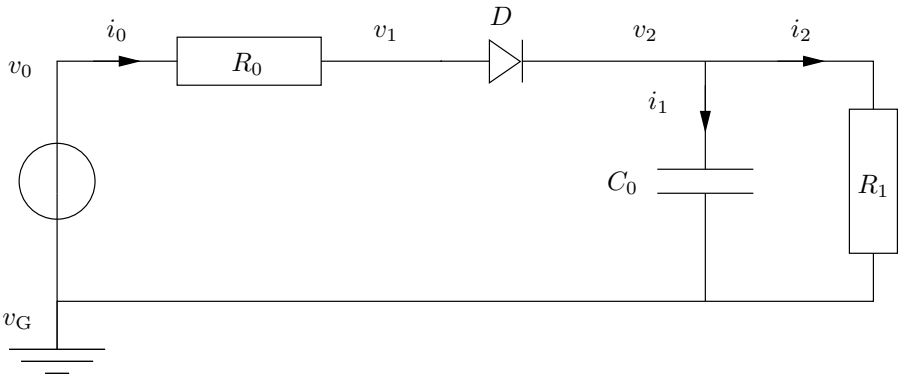

Figure 4.4: Half wave rectifier circuit.

The χ model is as follows:

$\langle$ cont $v_G, v_0, v_1, v_2, i_0, i_1, i_2$
, $v_2 = 0$

$$
\begin{array}{l}
\mid\ v_0 = \sin(2\pi f \mathsf{time}) \\
\|\ R(i_0, v_0, v_1, R_0) \\
\|\ D(i_0, v_1, v_2) \\
\|\ C(i_1, v_2, v_\mathrm{G}, C_0) \\
\|\ R(i_2, v_2, v_\mathrm{G}, R_1) \\
\|\ v_\mathrm{G} = 0 \\
\|\ i_0 = i_1 + i_2 \\
\rangle
\end{array}
$$

The process definitions of a diode D, a resistor R, and a capacitor C follow below.

$$
\begin{array}{l}
D(\mathsf{ext}\ i, v_\mathrm{in}, v_\mathrm{out}) = \\
[\![\ \mathrm{block} \quad \mapsto i = 0,\ v_\mathrm{out} \geq v_\mathrm{in}\ ]\!]\ [\mathrm{skip}];\ \mathrm{conduct} \\
,\ \mathrm{conduct} \mapsto v_\mathrm{in} = v_\mathrm{out},\ i \geq 0\ ]\!]\ [\mathrm{skip}];\ \mathrm{block} \\
\mid\ \mathrm{skip};\ \mathrm{block}\ ]\!]\ \mathrm{skip};\ \mathrm{conduct} \\
]\!]
\end{array}
$$

$$R(\mathsf{ext}\ i, v_\mathrm{in}, v_\mathrm{out}, \mathsf{val}\ R) = [\![\ v_\mathrm{in} - v_\mathrm{out} = iR\ ]\!]$$

Note that the same identifier R is used to denote both the value parameter and the process name in the resistor model. Using a different name for the value parameter, e.g. R_0:

$$R(\mathsf{ext}\ i, v_\mathrm{in}, v_\mathrm{out}, \mathsf{val}\ R_0) = [\![\ v_\mathrm{in} - v_\mathrm{out} = iR_0\ ]\!]$$

does not change the meaning of the model. The capacitor model is:

$$C(\mathsf{ext}\ i, v_\mathrm{in}, v_\mathrm{out}, \mathsf{val}\ C) = [\![\ \mathsf{cont}\ v \mid v = v_\mathrm{in} - v_\mathrm{out},\ C\dot{v} = i\ ]\!]$$

After replacing the process instantiations by their process bodies as defined in Section 2.5.2, the following χ process is obtained:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ v_\mathrm{G}, v_0, v_1, v_2, i_0, i_1, i_2 \\
,\ v_2 = 0 \\
\mid\ v_0 = \sin(2\pi f \mathsf{time}) \\
\|\ [\![\ \mathsf{disc}\ R,\ R = R_0 \\
\quad \mid\ v_0 - v_1 = i_0 R \\
\quad]\!] \\
\|\ [\![\ \mathrm{block} \quad \mapsto i_0 = 0,\ v_2 \geq v_1\ ]\!]\ [\mathrm{skip}];\ \mathrm{conduct} \\
\quad ,\ \mathrm{conduct} \mapsto v_1 = v_2,\ i_0 \geq 0\ ]\!]\ [\mathrm{skip}];\ \mathrm{block} \\
\quad \mid\ \mathrm{skip};\ \mathrm{block}\ ]\!]\ \mathrm{skip};\ \mathrm{conduct} \\
\quad]\!] \\
\|\ [\![\ \mathsf{disc}\ C,\ \mathsf{cont}\ v,\ C = C_0 \\
\quad \mid\ v = v_2 - v_\mathrm{G},\ C\dot{v} = i_1 \\
\quad]\!]
\end{array}
$$

$$
\begin{array}{l}
\| \; [\![\; \mathsf{disc}\ R,\ R = R_1 \\
\quad \mid\ v_2 - v_{\mathrm{G}} = i_2 R \\
\quad]\!] \\
\| \; v_{\mathrm{G}} = 0 \\
\| \; i_0 = i_1 + i_2 \\
\rangle
\end{array}
$$

4.4 A game of billiards

Figure 4.5 shows a billiard table of dimensions l and h with a ball, as defined in [ACH+95]. Initially, the position and velocity of the ball are given by (x_0, y_0) and $(v_{\mathrm{x}0}, v_{\mathrm{y}0})$, respectively. If the ball reaches a vertical side it rebounds, i.e. the sign of the horizontal velocity component v_{x} changes. The same occurs with the vertical velocity component v_{y} when the ball reaches a horizontal side. The combination of the signs of velocity components gives four different directions of movements.

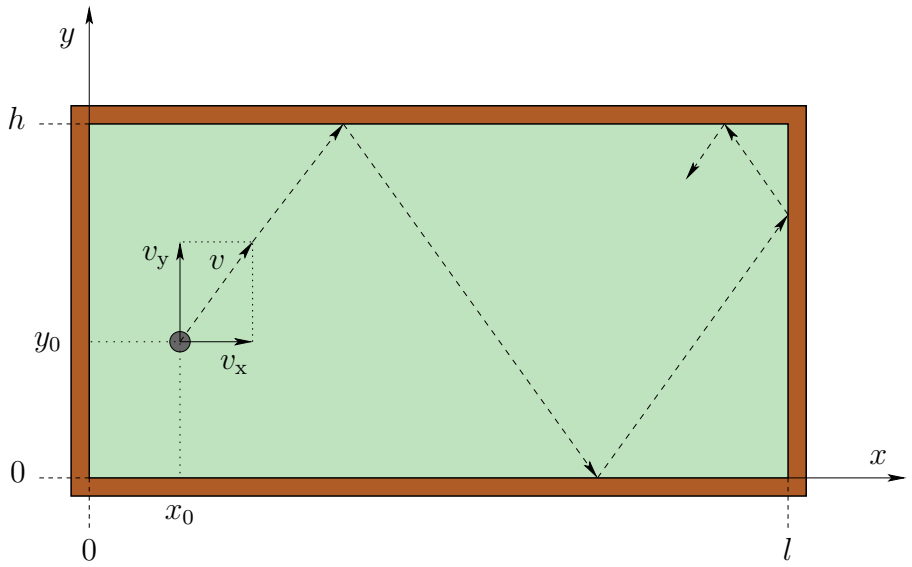

Figure 4.5: Billiard table

The movement of the ball is modeled by the following χ model, where l, h, x_0, y_0, $v_{\mathrm{x}0}$, and $v_{\mathrm{y}0}$ denote constants. Each possible combination of directions is represented by a recursion definition. The recursion variables RU, LU, LD, and RD correspond with the directions right-up, left-up, left-down, and right-down, respectively.

The relation between the position (x, y) and the velocity (v_x, v_y) is given by $\dot{x} = v_x$, and $\dot{y} = v_y$. When a collision occurs, the velocity changes instantaneously.

$$
\langle \; \mathsf{disc}\ v_{\mathrm{x}}, v_{\mathrm{y}},\ \mathsf{cont}\ x, y
$$

$$
\begin{array}{l}
\langle \\
\quad , \; v_{\mathrm{x}} = v_{\mathrm{x}0}, v_{\mathrm{y}} = v_{\mathrm{y}0}, x = x_0, y = y_0 \\
\quad , \; RU \mapsto x \leq l, y \leq h \;\; [\!] \; x = l \;\; \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}} \; ; \; LU \\
\qquad\qquad\qquad\qquad\quad [\!] \; y = h \;\; \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}} \; ; \; RD \\
\quad , \; LU \mapsto x \geq 0, y \leq h \;\; [\!] \; x = 0 \;\; \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}} \; ; \; RU \\
\qquad\qquad\qquad\qquad\quad [\!] \; y = h \;\; \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}} \; ; \; LD \\
\quad , \; LD \mapsto x \geq 0, y \geq 0 \;\; [\!] \; x = 0 \;\; \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}} \; ; \; RD \\
\qquad\qquad\qquad\qquad\quad [\!] \; y = 0 \;\; \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}} \; ; \; LU \\
\quad , \; RD \mapsto x \leq l, y \geq 0 \;\; [\!] \; x = l \;\; \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}} \; ; \; LD \\
\qquad\qquad\qquad\qquad\quad [\!] \; y = 0 \;\; \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}} \; ; \; RU \\
\quad | \; \dot{x} = v_{\mathrm{x}}, \dot{y} = v_{\mathrm{y}} \parallel RU \\
\rangle
\end{array}
$$

When the ball hits a corner, the sign of the horizontal and the sign of the vertical velocity component are changed sequentially in a arbitrary order, requiring two mode changes.

In the following model, the four recursion definitions are merged into one definition with ODE $\dot{x} = v_{\mathrm{x}}, \dot{y} = v_{\mathrm{y}}$. The any delay operator $[\ldots]$ around the assignments, e.g. $[v_{\mathrm{x}} := -v_{\mathrm{x}}]$, is required. Without the any delay operator, the assignment and thus the alternative composition cannot delay.

$$
\begin{array}{l}
\langle \; \mathsf{disc}\; v_{\mathrm{x}}, v_{\mathrm{y}}, \; \mathsf{cont}\; x, y \\
\quad , \; v_{\mathrm{x}} = v_{\mathrm{x}0}, v_{\mathrm{y}} = v_{\mathrm{y}0}, x = x_0, y = y_0 \\
\quad | \; * \; (\; \dot{x} = v_{\mathrm{x}}, \dot{y} = v_{\mathrm{y}}, 0 \leq x \leq l, 0 \leq y \leq h \\
\qquad [\!] \; (x = 0 \vee x = l) \rightarrow [v_{\mathrm{x}} := -v_{\mathrm{x}}] \\
\qquad [\!] \; (y = 0 \vee y = h) \rightarrow [v_{\mathrm{y}} := -v_{\mathrm{y}}] \\
\qquad) \\
\rangle
\end{array}
$$

4.5 Constrained pendulum

Figure 4.6 shows a constrained pendulum that is also defined in [vdSS00, BH04]. The equations of motion of this pendulum are given by Equation 4.1. The angle between the pendulum and the vertical is denoted by θ, ω denotes the angular velocity of the pendulum, and l denotes the distance between the rotation point and the mass.

$$
\begin{aligned}
\dot{\theta} &= \omega \\
ml\dot{\omega} &= -mg\sin(\theta) - dl\omega
\end{aligned}
\tag{4.1}
$$

The mass and maximum length of the pendulum are represented by m and L, respectively. The damping coefficient and the acceleration due to gravity are denoted by d and g. The angle of the constraint is denoted by θ_{p}. In order to keep the example as small and clear as possible, it is assumed that $\theta_{\mathrm{p}} \geq 0$ and $|\theta| \leq \pi/2$. Also, it is assumed that the pendulum always remains in a straight line from the rotation point to the end point. The χ model is:

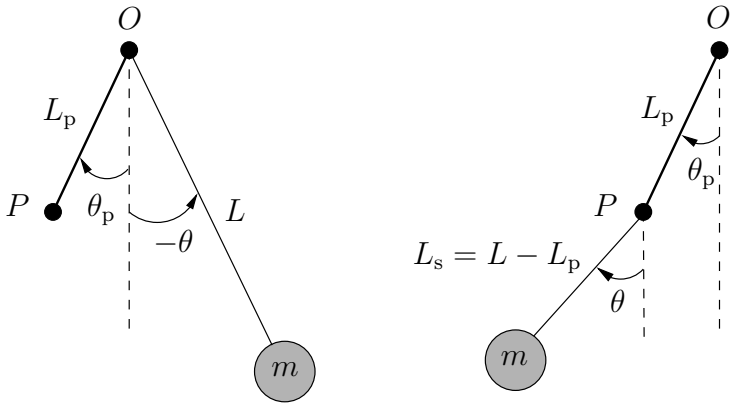

Figure 4.6: Constrained Pendulum.

$\langle$ cont θ, ω, alg l
, $\theta = \theta_0, \omega = \omega_0$
, long $\mapsto l = L,\ \theta \leq \theta_\mathrm{p}$ [] $[\omega := \frac{L}{L_\mathrm{s}}\omega]$; short
, short $\mapsto l = L_\mathrm{s},\ \theta \geq \theta_\mathrm{p}$ [] $[\omega := \frac{L_\mathrm{s}}{L}\omega]$; long
| (skip; long [] skip; short)
|| $\dot{\theta} = \omega$
, $ml\dot{\omega} = -mg\sin(\theta) - dl\omega$
$\rangle$,

where θ_0 and ω_0 denote constants representing the initial values of θ and ω, respectively. When $\theta \leq \theta_\mathrm{p}$ or $\theta \geq \theta_\mathrm{p}$, the pendulum can delay in mode long or short, respectively. In mode long, the assignment $\omega := \frac{L}{L_\mathrm{s}}\omega$ can be executed only if the new state after the assignment to ω is consistent with the constraints $l = L_\mathrm{s}$, $\theta \geq \theta_\mathrm{p}$ of mode short, because a process cannot enter an inconsistent state. Therefore, mode switches are possible only for $\theta = \theta_\mathrm{p}$. The any delay operator applied on the assignment in $[\omega := \frac{L}{L_\mathrm{s}}\omega]$ is needed, because otherwise the assignment and the alternative composition would not be able to delay. Note that the model allows infinite switching between modes long and short, without progress of time, when $\theta = \theta_\mathrm{p}$. This switching behavior can, in principle, be avoided by guarding the delayable assignments $[\omega := \frac{L}{L_\mathrm{s}}\omega]$ and $[\omega := \frac{L_\mathrm{s}}{L}\omega]$ with (non-trivial) conditions that prevent mode switching when no delay behavior is possible in the new mode.

4.6 Dry friction phenomenon

Figure 4.7 shows a driving force F_d applied to a body on a flat surface with frictional force F_f. When the body is moving with positive velocity v, the frictional force is given by $F_\mathrm{f} = \mu F_\mathrm{N}$, where $F_\mathrm{N} = mg$. When the velocity of the body equals zero and $-\mu_0 F_\mathrm{N} \leq F_\mathrm{d} \leq \mu_0 F_\mathrm{N}$, where $\mu_0 > \mu$, the frictional force neutralizes the applied driving force.

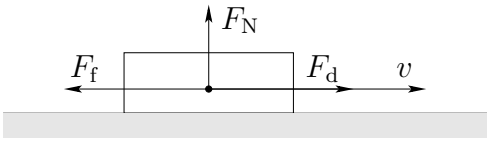

Figure 4.7: Dry Friction.

In the χ specification of the dry friction phenomenon, modes neg, stop, and pos are specified by means of mode definitions. Mode stop can be maintained for as long as the delay predicate $v = 0, -\mu_0 F_N \leq F_d \leq \mu_0 F_N$ can delay. One of the two skip actions can be executed when the corresponding guard $F_d \leq -\mu_0 F_N$ or $F_d \geq \mu_0 F_N$ becomes true, and the state is consistent with the constraints $v \leq 0$ or $v \geq 0$ of the respective mode neg or pos that is activated after the skip action.

The mode pos (or neg) is maintained until the guard $F_d < \mu_0 F_N$ (or $F_d > -\mu_0 F_N$) becomes true. The skip action after the guard cannot delay. Therefore, when the guard becomes true, the skip action must be executed. Subsequently, mode stop becomes active again.

Initially either mode neg, stop or pos is chosen by means of the internal skip action (skip; neg [] skip; stop [] skip; pos), based on the initial values of v and F_d. Identifier f denotes some function $\in \mathbb{R} \to \mathbb{R}$; m, F_N, μ_0, μ ($\mu_0 > \mu$) denote constants.

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x, v,\ \mathsf{alg}\ F_d \\
,\ x = 0, v = 0 \\
,\ \text{stop} \mapsto v = 0,\ -\mu_0 F_N \leq F_d \leq \mu_0 F_N \ [\!]\ [F_d \leq -\mu_0 F_N \to \text{skip}];\ \text{neg} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad [\!]\ [F_d \geq \mu_0 F_N \quad \to \text{skip}];\ \text{pos} \\
,\ \text{pos} \mapsto m\dot{v} = F_d - \mu F_N,\ v \geq 0 \ [\!]\ F_d < \mu_0 F_N \to \text{skip};\ \text{stop} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad [\!]\ [F_d \leq -\mu_0 F_N \to \text{skip}];\ \text{neg} \\
,\ \text{neg} \mapsto m\dot{v} = F_d + \mu F_N,\ v \leq 0 \ [\!]\ F_d > -\mu_0 F_N \to \text{skip};\ \text{stop} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad [\!]\ [F_d \geq \mu_0 F_N \to \text{skip}];\ \text{pos} \\
|\ F_d = f(\mathsf{time}), \dot{x} = v \\
\|\ \text{skip};\ \text{neg}\ [\!]\ \text{skip};\ \text{stop}\ [\!]\ \text{skip};\ \text{pos} \\
\rangle
\end{array}
$$

4.7 Railroad gate control

Consider a train on a circular track, a gate and a controller. When the train approaches the gate, the controller must lower the gate. The controller has a reaction delay u of at most 5 time units. After the train has passed the gate the controller must raise the gate. The purpose of the model is to determine whether or not the gate is always fully lowered when the train is at a certain distance from the gate. Figures 4.8, 4.9, and 4.10 show automaton models of the train, gate and controller respectively. Together, they form the rail gate control system as defined in [Hen00b]. Note that in a vertex, the predicate at the top denotes the flow predicate, and the predicate at the bottom denotes the invariant

predicate of the vertex. Furthermore, as usual, event labels of edges which do not have to synchronize with other edges, and init predicates false are omitted in the figures.

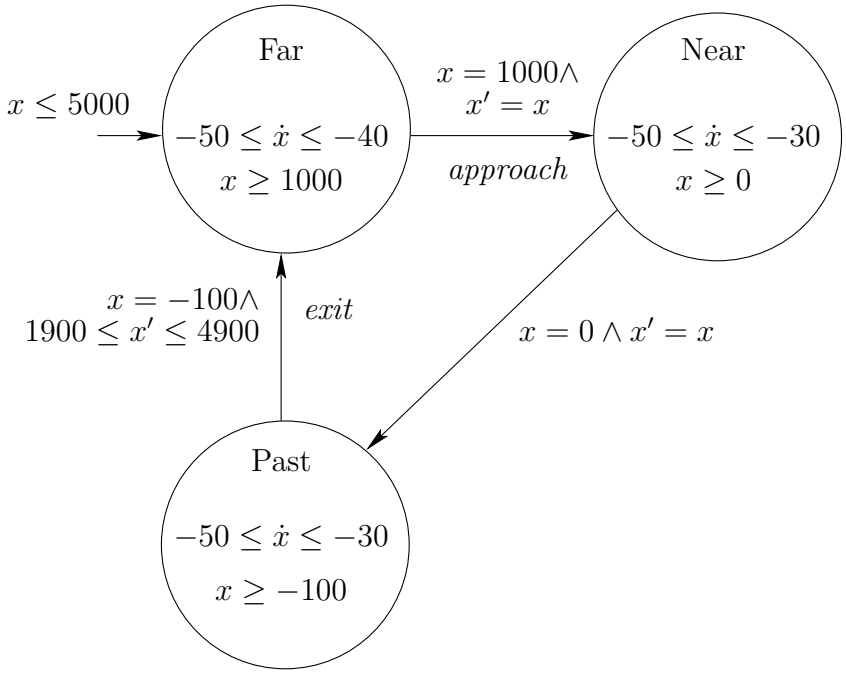

Figure 4.8: *Train* Automaton.

The χ model takes into account that there is only one train on the circular track, as defined in [Hen00b], which implies that the transitions of the self loops of the controller automaton can never occur. Figure 4.11 shows the iconic χ model of the railroad gate controller. The dashed lines with arrow heads represent synchronization channels (*approach*, *exit*, *raise*, *lower*), no data is communicated.

The following process is a formal specification of the informal iconic model from Figure 4.11. Variable x and y are initialized to a value ≤ 5000 and 90, respectively. The maximum reaction delay (5 time units) of the controller is specified in its process instantiation C.

$\langle$ cont x, y
, chan *approach*, *exit*, *raise*, *lower*
, $1000 \leq x \leq 5000, y = 90$
| *Train*(*x*, *approach*, *exit*)
|| *Gate*(*y*, *raise*, *lower*)
|| *C*(*approach*, *exit*, *raise*, *lower*, 5)
$\rangle$

The train is modeled by the following process definition:

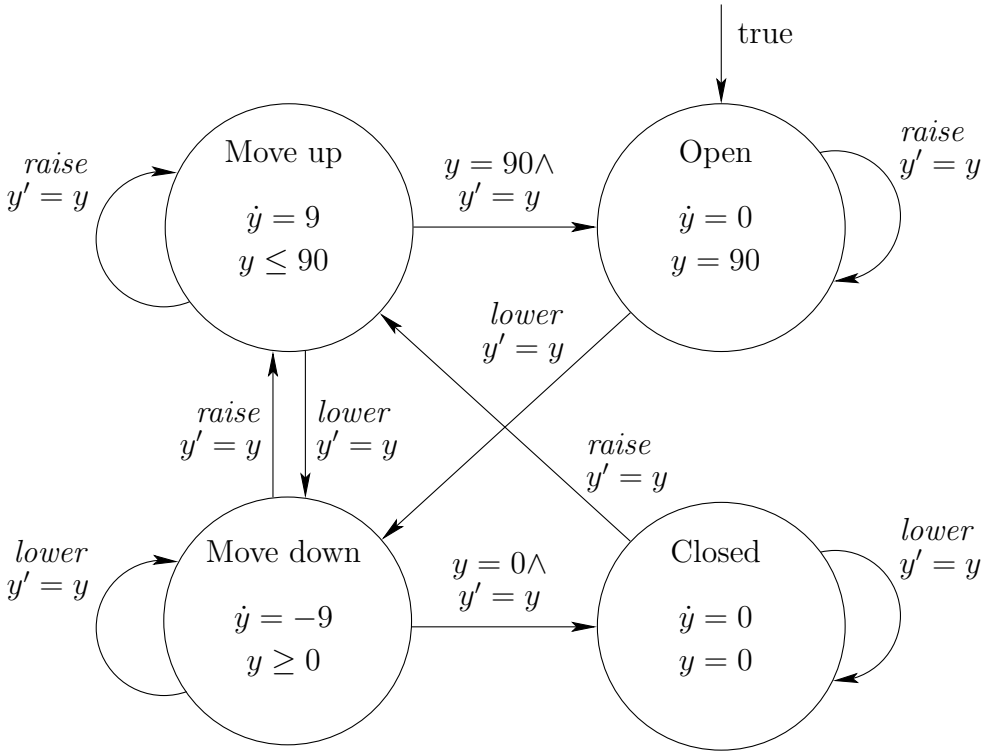

Figure 4.9: *Gate* Automaton.

$$
\begin{array}{l}
\mathit{Train}(\mathsf{ext}\ x,\ \mathsf{chan}\ \mathit{approach}, \mathit{exit}) = \\
[\![\ *(\ (\dot{x} \in [-40, -50]\ [\!]\ x = 1000 \rightarrow \mathit{approach}\,!!) \\
\quad ;\ (\dot{x} \in [-30, -50]\ [\!]\ x = -100 \rightarrow \mathit{exit}\,!!;\ x : x \in [1900, 4900]) \\
\quad) \\
]\!]
\end{array}
$$

The process definition consists of an infinite loop $*(\cdots)$. The velocity $\dot{x}$ of the train can be any function of time, the value of which remains between -50 and -40. The process waits until the train has reached position $x = 1000$ and then synchronizes with the controller (*approach* !!). The train is now approaching the gate. If the train has reached the exit position, such that $x = -100$, the process synchronizes with the controller, the position x of the train is reset to a value between 1900 and 4900, and the loop is re-executed.

The gate is modeled by the following process definition:

$$
\begin{array}{l}
\mathit{Gate}(\mathsf{ext}\ y,\ \mathsf{chan}\ \mathit{raise}, \mathit{lower}) = \\
[\![\ \mathsf{disc}\ n,\ n = 0 \\
|\ \dot{y} = n
\end{array}
$$

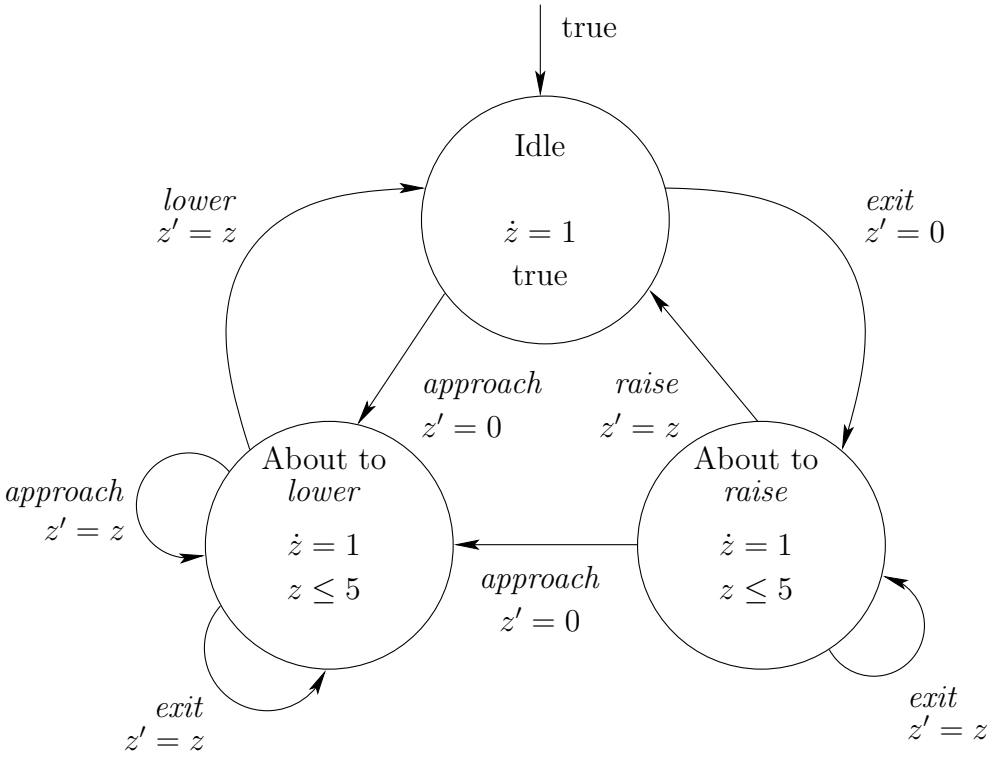

Figure 4.10: *Control* Automaton.

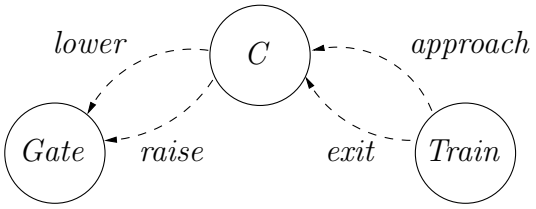

Figure 4.11: Iconic χ model of the railroad gate controller.

$$
\begin{array}{l}
[\!] * (\; n < 0 \wedge y \leq 0 \rightarrow n := 0 \\
\quad [\!] \; n > 0 \wedge y \geq 90 \rightarrow n := 0 \\
\quad [\!] \; \mathit{raise}\,?;\; n := 9 \\
\quad [\!] \; \mathit{lower}\,?;\; n := -9 \\
\quad) \\
[\!]
\end{array}
$$

It consists of a parallel composition of an equation ($\dot{y} = n$), where n denotes a local discrete variable, and an infinite loop. This infinite loop is an alternative composition of four process terms. The first process term waits until the gate is lowered ($y = 0$) and then stops the gate from lowering ($n := 0$). The second process term waits until the gate is raised ($y = 90$).

The third and fourth process term wait for synchronization with the controller in order to raise or lower the gate (*raise*? and *lower*?, respectively). The four process terms delay in parallel until y becomes equal to 0 or 90, or one of the synchronizations (*raise*? or *lower*?) succeeds.
The controller is modeled by the following process definition:

$$
\begin{array}{l}
C(\mathsf{chan}\ \mathit{approach}, \mathit{exit}, \mathit{raise}, \mathit{lower},\ \mathsf{val}\ u) = \\
[\![\ \mathsf{disc}\ \mathit{atr},\ \mathit{atr} = \mathrm{false} \\
\mid *(\ \mathit{approach}?;\ \mathit{atr} := \mathrm{false};\ (\Delta u\ [\!]\ [\mathrm{skip}]);\ \Delta\, t;\ \mathit{lower}\,!! \\
\quad [\!]\ \mathit{exit}?;\ \mathit{atr} := \mathrm{true} \\
\quad [\!]\ \mathit{atr} \rightarrow (\Delta u\ [\!]\ [\mathrm{skip}]);\ \mathit{atr} := \mathrm{false};\ \mathit{raise}\,!! \\
\quad) \\
]\!]
\end{array}
$$

The main part is an infinite loop of three alternatives. The process waits for one of the following events to occur: an approaching train (*approach*?), a leaving train (*exit*?), or if *atr* is true, the end of the reaction delay ($\Delta u\ [\!]\ [\mathrm{skip}]$) that precedes raising of the gate. Process term $\Delta u\ [\!]\ [\mathrm{skip}]$, where u denotes the maximum reaction delay in the controller, models a non-deterministic delay between 0 and u.

Boolean variable *atr* is true if and only if the hybrid automaton that models the controller is in control mode (vertex) 'About to raise'. Note that variable z in the hybrid automaton is used to model a clock. In χ, clocks need not be modeled explicitly. Delay process terms Δu are used for that purpose.

4.8 Glider take-off

Figure 4.12 shows a glider that is towed off the ground by a tow plane. The position, velocity and acceleration of the tow plane are given by x_1, v_1, a, respectively. The position and velocity of the glider are given by x_2 and v_2.

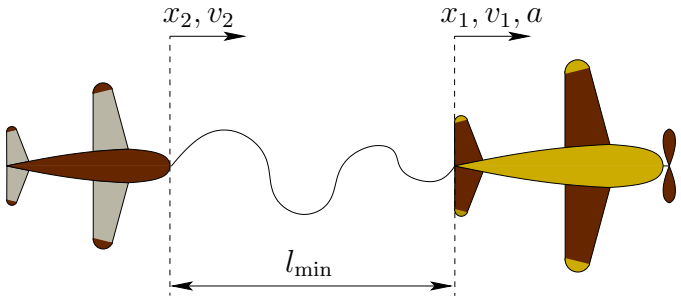

Figure 4.12: Glider take-off.

Initially, the tow plane and glider are standing still at a distance of $l_{\min}$. After one unit of time, the tow plane very slowly accelerates ($a := 0.02$) until the tow cable is at its

maximum length of $l_{\max}$. At that moment, the velocity of the glider jumps discontinuously to the velocity of the tow plane. We assume the mass of the glider to be considerably smaller than the mass of the tow plane. The tow plane then accelerates ($a := 0.5$) until its velocity is at $v_{\max}$. After another t units of time, the glider releases the tow cable, and continues on its own. Its velocity is then assumed to be determined by the air resistance, which is proportional to the squared velocity (kv_2^2), and the propelling forces F, which we assume constant in the model below:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s, a, \mathsf{cont}\ x_1, x_2, v_1, v_2 \\
,\ s = \textsc{stop}, a = 0, x_1 = l_{\min}, x_2 = 0, v_1 = 0 \\
|\ \dot{x}_1 = v_1,\ \dot{x}_2 = v_2,\ \dot{v}_1 = m_1 a \\
\|s = \textsc{stop}\ \rightarrow v_2 = 0 \\
\|s = \textsc{tow}\ \rightarrow v_2 = v_1 \\
\|s = \textsc{fly}\ \rightarrow \dot{v_2} = F - kv_2^2 \\
\|\ \Delta 1;\ a := 0.02 \\
;\ x_1 - x_2 \geq l_{\max} \rightarrow (\mathsf{jump}\ v_2 \mid s := \textsc{tow}) \\
;\ a := 0.5;\ v_1 \geq v_{\max} \rightarrow a := 0 \\
;\ \Delta t;\ s := \textsc{fly} \\
\rangle
\end{array}
$$

In the model, m_1 is a constant denoting the mass of the towing plane, k is some constant, and enumeration variable s denotes the state of the glider. When the distance between the tow plane and the glider becomes equal to the maximum length of the cable, the glider abruptly starts moving. This is modeled by ($\mathsf{jump}\ v_2 \mid s := \textsc{tow}$). The jump enabling operator ($\mathsf{jump}\ v_2 \mid \ldots$) enables jumps for continuous variable v_2 when assignment $s :=$ TOW) is executed. This is necessary, because v_2 is declared as a (non-jumping) continuous variable. The only assignment where v_2 must be able to jump is the assignment $s :=$ TOW, because then v_2 must discontinuously change to the value of v_1 in order to satisfy equation $v_2 = v_1$ that must hold for $s =$ TOW. In this example, the relation $v_2 = v_1$ in mode TOW is so straightforward, that the jumping behavior of variable v_2 when mode TOW becomes active can also be modeled explicitly by means of a multi-assignment ($s, v_2 :=$ TOW$, v_1$) instead of ($\mathsf{jump}\ v_2 \mid s :=$ TOW). The model with the jump enabling operator is more general, because it can also be used in cases where the algebraic constraints are so complex that it becomes difficult, or impossible, to explicitly calculate the new value of the jumping variable after the discontinuity.

4.9 Bottle filling system

The bottle filling system from Figure 7.1 consists of a liquid storage tank, and two identical bottle filling lines.

The bottles are filled with liquid from the storage tank. A control system keeps the volume V_{T} in the storage tank between 2 and 10, and the pH level (acidity) of the liquid in the storage tank between 7 and 7.1. The liquid in the storage tank slowly becomes less

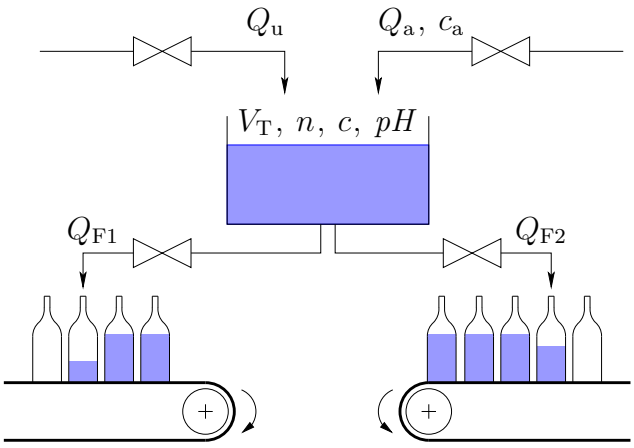

Figure 4.13: The bottle filling system.

acidic (pH level increases). To correct this, a strong acid is dribbled into the storage tank when the acidity of the liquid becomes too low ($pH \geq 7.1$).

Figure 4.14 shows the iconic model of the bottle filling system. The lines ending in a small circle represent shared variables (V_{T}, Q_{F_1}, Q_{F_1}).

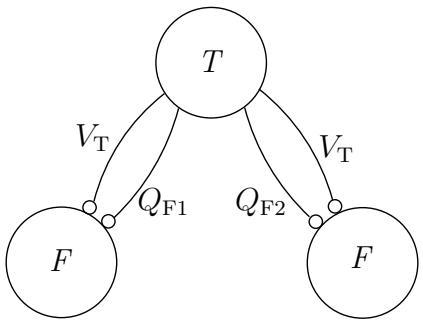

Figure 4.14: Iconic representation of the bottle filling system model.

The acid and liquid supply processes are not modeled, since we consider the acid and liquid always to be available, and we are not interested in the amount of acid or liquid that is used. The χ specification of the bottle filling system is as follows:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ V_{\mathrm{T}},\ \mathsf{alg}\ Q_{\mathrm{F1}}, Q_{\mathrm{F2}} \\
,\ V_{\mathrm{T}} = 2 \\
|\ T(V_{\mathrm{T}}, Q_{\mathrm{F1}}, Q_{\mathrm{F2}}) \\
\|\ F(V_{\mathrm{T}}, Q_{\mathrm{F1}}) \\
\|\ F(V_{\mathrm{T}}, Q_{\mathrm{F2}}) \\
\rangle
\end{array}
$$

The storage tank and the two bottle filling lines are connected by means of the variables Q_{F1}, and Q_{F2}, respectively. Since a bottle may start filling only if the storage tank contains at least a volume of 0.7, the volume V_{T} of the storage tank is available in both bottle filling processes.

The molar quantity and molar concentration of the acid in the storage tank are denoted by n and c, respectively, where $n = cV$. The incoming flows of liquid and acid of the liquid storage tank T are denoted by Q_{u} and Q_{a}, respectively. The outgoing flows to the two bottle filling processes are denoted by Q_{F1} and Q_{F2}, respectively.

It is assumed that the liquids are incompressible, and that the volumes of the fluids remain the same when they are mixed. In such a case, the volume V of the mixed liquid equals the sum of its components which leads to the following equation

$$\dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F1}} - Q_{\mathrm{F2}}.$$

Next, the mass balance (actually mol balance) for the dissolved substance is derived. Acid comes into the tank by means of the flows Q_{u} and Q_{a}. Acid leaves the tank in outgoing flows Q_{F1} and Q_{F2}. Because the concentrations are in [mol/m^3], they can be directly multiplied with the flows (in [m^3/s]), which leads to

$$\dot{n} = c_{\mathrm{u}}Q_{\mathrm{u}} + c_{\mathrm{a}}Q_{\mathrm{a}} - cQ_{\mathrm{F1}} - cQ_{\mathrm{F2}},$$

where c_{u} and c_{a} denote the concentrations of acid in the flows Q_{u} and Q_{a}. The gradual reduction of the acidity of the liquid is modeled by means of a constant K_{loss}, which leads to

$$\dot{n} = c_{\mathrm{u}}Q_{\mathrm{u}} + c_{\mathrm{a}}Q_{\mathrm{a}} - cQ_{\mathrm{F1}} - cQ_{\mathrm{F2}} - K_{\mathrm{loss}}V.$$

It is assumed that the acid is completely decomposed. Taking into account that the units of c are in [mol/m^3] instead of [mol/l], the pH is given by

$$pH = -\log c/1000.$$

The χ specification of the liquid storage tank follows below, where symbols Q_{seta}, Q_{setu}, c_{a}, c_{u}, and K_{loss} denote constants:

$$
\begin{array}{l}
T(\mathsf{ext}\ V, Q_{\mathrm{F1}}, Q_{\mathrm{F2}}) \\
[\![\ \mathsf{disc}\ \alpha, \beta,\ \mathsf{cont}\ n,\ \mathsf{alg}\ pH, c, Q_{\mathrm{a}}, Q_{\mathrm{u}} \\
,\ \alpha = 0, \beta = 0, pH = 7 \\
|\quad \dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F1}} - Q_{\mathrm{F2}} \\
,\quad \dot{n} = c_{\mathrm{u}}Q_{\mathrm{u}} + c_{\mathrm{a}}Q_{\mathrm{a}} - cQ_{\mathrm{F1}} - cQ_{\mathrm{F2}} - K_{\mathrm{loss}}V \\
,\quad n = cV \\
,\ pH = -\log c/1000 \\
,\ Q_{\mathrm{a}} = \alpha Q_{\mathrm{seta}} \\
,\ Q_{\mathrm{u}} = \beta Q_{\mathrm{setu}} \\
[\!|\ *(\ pH \geq 7.1 \rightarrow \alpha := 1;\ pH \leq 7 \rightarrow \alpha := 0\ ) \\
[\!|\ *(\ V \leq 2 \rightarrow \beta := 1;\ V \geq 10 \rightarrow \beta := 0\ ) \\
]\!]
\end{array}
$$

The model of the liquid storage tank T illustrates that a differential variable, such as variable n, is not necessarily initialized. In this case, instead, the algebraic variable pH is initialized ($pH = 7$). The continuous variables of the bottle filling system with tank T, can be declared in different ways.

In most cases, the differential variables, in this case V and n, are declared as (non-jumping) continuous variables. The other variables, not occurring with a dot (derivative) are then declared as algebraic variables. This ensures that the differential variables can be assigned new values, causing discontinuities. The algebraic variables will then simultaneously jump to their new values satisfying the equations. This declaration scheme is used in process T. Note that variable V is an external variable that is declared as a (non-jumping) continuous variable in the preceding χ process that defines the complete bottle filling system. Note that even though pH is an algebraic variable, which is not normally assigned new values, pH can be initialized, in this case to a value of 7, in the initialization predicate.

In process T, the only discontinuities in continuous variables occur in the flows Q_{F1}, Q_{F2}, Q_{a}, and Q_{u}, that are switched on and off discontinuously in process T, and in process F that follows below. Therefore, the algebraic variables apart from these flows could just as well have been declared as (non-jumping) continuous variables as in $\mathsf{cont}\ n, pH, c$.

The behavior of the model is explained as follows. Initially, the pH of the liquid in the storage tank equals 7. It is assumed that the pH level of the incoming liquid is 7 or more, since the acidity controller can only make the acidity of the storage tank increase, causing the pH to decrease. If the pH value exceeds the maximum value ($pH \geq 7.1$), the acid valve is opened ($\alpha := 1$) so that acid is dribbled into the tank. Dribbling of the acid continues until the pH value comes back at 7, and the valve is closed ($\alpha := 0$). In a similar way, the controller tries to keep the level of the storage tank between 2 and 10.

The model of a bottle filling line follows below, where symbols Q_{setF}, and t_{tr} denote constants.

$$
\begin{array}{l}
F(\mathsf{ext}\ V_{\mathrm{T}}, Q_{\mathrm{F}}) = \\
[\![\ \mathsf{disc}\ \alpha,\ \mathsf{cont}\ V \\
,\ \alpha = 0, V = 0 \\
|\ \ \dot{V} = Q_{\mathrm{F}} \\
,\ Q_{\mathrm{F}} = \alpha Q_{\mathrm{setF}} \\
\|\ *(\ V_{\mathrm{T}} \geq 0.7 \rightarrow \alpha := 1 \\
\quad ;\ \alpha = 1\ \stackrel{*}{\rightarrow}\ (\ V \geq 1 \rightarrow \alpha := 0 \\
\qquad\qquad\qquad [\!]\ V_{\mathrm{T}} \leq 0.5 \rightarrow \alpha := 0;\ V_{\mathrm{T}} \geq 0.7 \rightarrow \alpha := 1 \\
\qquad\qquad\qquad)\\
\quad ;\ \Delta t_{\mathrm{tr}};\ V := 0 \\
\quad) \\
]\!]
\end{array}
$$

The valve switching the flow Q_{F} is modeled by means of the discrete variable α. When the volume in the storage tank is at least 0.7, the bottle filling process can be started ($\alpha :=$

1). Filling stops when the volume in the storage tank drops below 0.5 ($V_{\mathrm{T}} \leq 0.5 \rightarrow \alpha := 0$). Filling resumes when the volume in the storage tank is at least 0.7. Filling also stops when the bottle is full ($V \geq 1 \rightarrow \alpha := 0$). The time needed to place a new bottle under the filling nozzle is given by t_{tr}. After that, the bottle volume is reset to 0, which models the arrival of a new bottle, and the filling process is repeated.

4.10 Conveyor system

Figure 4.15 shows a conveyor system that is used for transportation and buffering of boxes. It consists of a line of conveyor belts driven by motors. Each conveyor belt is equipped with a sensor (represented in the figure by a small rectangle), that detects the presence of a box.

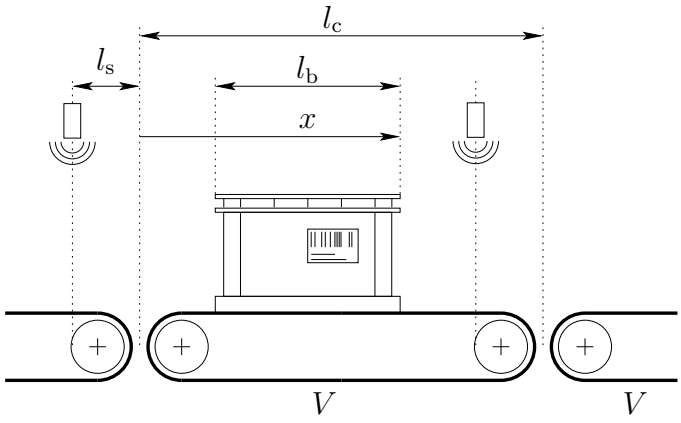

Figure 4.15: The conveyor system.

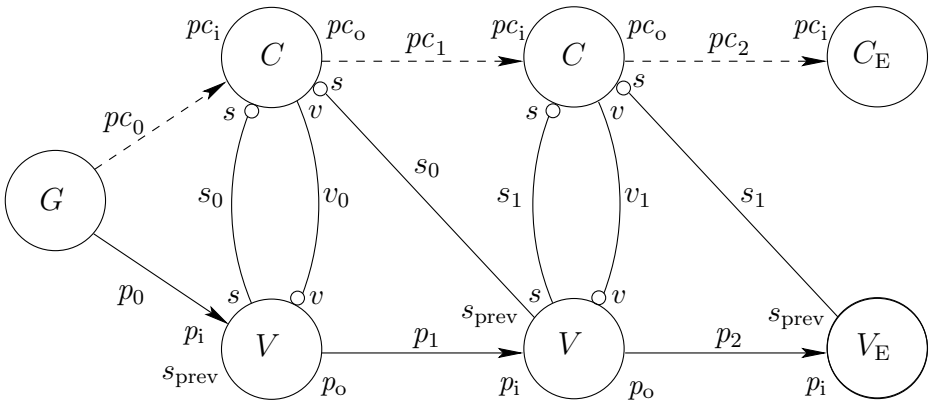

Figure 4.16: Iconic model of the conveyor system.

Figure 4.16 shows the iconic model of a generator G, two conveyor belts V and the associated control processes C. Processes V_{E} and C_{E} are added to obtain a closed system; they do not model actual behavior. The model is a simplified version of the model treated in [vBGR97]. The dashed lines with arrow heads represent directed synchronization channels (pc_0, pc_1, pc_2), the solid lines with arrow heads represent directed communication channels (p_0, p_1, p_2), and the lines ending in a small circle represent shared variables (s_0, s_1, v_0, v_1). The χ specification of the iconic model from Figure 4.16 is as follows:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s_{\mathrm{G}}, s_0, s_1, l_{\mathrm{s}}, v_0, v_1 \\
,\ \mathsf{chan}\ pc_0, pc_1, pc_2, p_0, p_1, p_2 \\
,\ s_0 = \mathrm{false}, s_1 = \mathrm{false}, l_{\mathrm{s}} = 2, v_0 = 0, v_1 = 0 \\
|\ G(pc_0, p_0) \\
\|\ C(v_0, s_0, pc_0, pc_1) \\
\|\ C(v_1, s_1, pc_1, pc_2) \\
\|\ C_{\mathrm{E}}(pc_2) \\
\|\ V(s_{\mathrm{G}}, s_0, v_0, p_0, p_1, 10, 15, l_{\mathrm{s}}) \\
\|\ V(s_0, s_1, v_1, p_1, p_2, 10, 15, l_{\mathrm{s}}) \\
\|\ V_{\mathrm{E}}(s_1, p_2, 10, l_{\mathrm{s}}) \\
\rangle\ ,
\end{array}
$$

where l_{s} represents the distance l_{s} as shown in Figure 4.15. Channels p_0, p_1, p_2 are used to communicate box numbers. Channels pc_0, pc_1, pc_2 are directed synchronization channels. Variables s_0, s_1, v_0, v_1 are shared variables, where s_0 and s_1 represent the sensors that indicate the presence of a box, and v_0 and v_1 are actuators that determine the velocity of the respective conveyors. The process definitions are:

$$
\begin{array}{l}
G(\mathsf{chan}\ pc_{\mathrm{o}}, p_{\mathrm{o}}) = \\
[\![\ \mathsf{disc}\ box,\ box = 1 \\
|\ *(\ pc_{\mathrm{o}}!;\ p_{\mathrm{o}}\,!\,box;\ box := box + 1\ ) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
C(\mathsf{ext}\ v, s,\ \mathsf{chan}\ pc_{\mathrm{i}}, pc_{\mathrm{o}}) = \\
[\![\ *(\ v := 1;\ \neg s \rightarrow pc_{\mathrm{i}}\,?;\ s \rightarrow v := 0;\ pc_{\mathrm{o}}!\ ) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
V(\mathsf{ext}\ s_{\mathrm{prev}}, s, v,\ \mathsf{chan}\ p_{\mathrm{i}}, p_{\mathrm{o}},\ \mathsf{val}\ l_{\mathrm{b}}, l_{\mathrm{c}}, l_{\mathrm{s}}) = \\
[\![\ \mathsf{disc}\ box,\ \mathsf{cont}\ x \\
,\ box = 0, x = -1 \\
|\ box = 0 \rightarrow \dot{x} = 0 \parallel box \neq 0 \rightarrow \dot{x} = v \\
\|\ *(\ p_{\mathrm{i}}\,?\,box;\ x := 0 \\
\quad ;\ x \geq l_{\mathrm{b}} - l_{\mathrm{s}} \rightarrow s_{\mathrm{prev}} := \mathrm{false} \\
\quad ;\ x \geq l_{\mathrm{c}} - l_{\mathrm{s}} \rightarrow s := \mathrm{true} \\
\quad ;\ x \geq l_{\mathrm{c}} \rightarrow p_{\mathrm{o}}\,!\,box;\ x, box := -1, 0 \\
\quad) \\
]\!]
\end{array}
$$

$$C_{\mathrm{E}}(\mathsf{chan}\ pc_{\mathrm{i}}) = [\![\ *(\ pc_{\mathrm{i}}\,?\ )\ ]\!]$$

$$\begin{array}{l} V_{\mathrm{E}}(\mathsf{ext}\ s_{\mathrm{prev}},\ \mathsf{chan}\ p_{\mathrm{i}},\ \mathsf{val}\ l_{\mathrm{b}}, l_{\mathrm{s}}) = \\ [\![\ \mathsf{disc}\ box \\ \mid\ *(\ p_{\mathrm{i}}\,?\,box;\ \Delta(l_{\mathrm{b}} - l_{\mathrm{s}});\ s_{\mathrm{prev}} := \mathrm{false}\ ) \\ ]\!] \end{array}$$

After replacing the process instantiations by their process bodies as defined in Section 2.5.2, the following χ process is obtained:

$$\begin{array}{l} \langle\ \mathsf{disc}\ s_{\mathrm{G}}, s_0, s_1, l_{\mathrm{s}}, v_0, v_1 \\ ,\ \mathsf{chan}\ pc_0, pc_1, pc_2, p_0, p_1, p_2 \\ ,\ s_0 = \mathrm{false}, s_1 = \mathrm{false}, l_{\mathrm{s}} = 2, v_0 = 0, v_1 = 0 \\ \mid\ [\![\ \mathsf{disc}\ box,\ box = 1 \\ \quad \mid\ *(\ pc_0\,!;\ p_0\,!\,box;\ box := box + 1\ ) \\ \quad]\!] \\ \|\ [\![\ *(\ v_0 := 1;\ \neg s_0 \rightarrow pc_0\,?;\ s_0 \rightarrow v_0 := 0;\ pc_1\,!\ )]\!] \\ \|\ [\![\ *(\ v_1 := 1;\ \neg s_1 \rightarrow pc_1\,?;\ s_1 \rightarrow v_1 := 0;\ pc_2\,!\ )]\!] \\ \|\ [\![\ *(\ pc_2\,?\ )\ ]\!] \\ \|\|[\![\ \mathsf{disc}\ box, l_{\mathrm{b}}, l_{\mathrm{c}}, \_l_{\mathrm{s}},\ \mathsf{cont}\ x \\ \quad ,\ box = 0, x = -1, l_{\mathrm{b}} = 10, l_{\mathrm{c}} = 15, \_l_{\mathrm{s}} = l_{\mathrm{s}} \\ \quad \mid\ box = 0 \rightarrow \dot{x} = 0\ \|\ box \neq 0 \rightarrow \dot{x} = v_0 \\ \quad\ \|\ *(\ p_0\,?\,box;\ x := 0 \\ \qquad\ ;\ x \geq l_{\mathrm{b}} - \_l_{\mathrm{s}} \rightarrow s_{\mathrm{G}} := \mathrm{false} \\ \qquad\ ;\ x \geq l_{\mathrm{c}} - \_l_{\mathrm{s}} \rightarrow s_0 := \mathrm{true} \\ \qquad\ ;\ x \geq l_{\mathrm{c}} \rightarrow p_1\,!\,box;\ x, box := -1, 0 \\ \qquad\ ) \\ \quad]\!] \\ \|\ [\![\ \mathsf{disc}\ box \mid\ *(\ pc_2\,?\,box\ )\ ]\!] \\ \|\|[\![\ \mathsf{disc}\ box, l_{\mathrm{b}}, l_{\mathrm{c}}, \_l_{\mathrm{s}},\ \mathsf{cont}\ x \\ \quad ,\ box = 0, x = -1, l_{\mathrm{b}} = 10, l_{\mathrm{c}} = 15, \_l_{\mathrm{s}} = l_{\mathrm{s}} \\ \quad \mid\ box = 0 \rightarrow \dot{x} = 0\ \|\ box \neq 0 \rightarrow \dot{x} = v_1 \\ \quad\ \|\ *(\ p_1\,?\,box;\ x := 0 \\ \qquad\ ;\ x \geq l_{\mathrm{b}} - \_l_{\mathrm{s}} \rightarrow s_0 := \mathrm{false} \\ \qquad\ ;\ x \geq l_{\mathrm{c}} - \_l_{\mathrm{s}} \rightarrow s_1 := \mathrm{true} \\ \qquad\ ;\ x \geq l_{\mathrm{c}} \rightarrow p_2\,!\,box;\ x, box := -1, 0 \\ \qquad\ ) \\ \quad]\!] \\ \|\ [\![\ \mathsf{disc}\ l_{\mathrm{b}}, \_l_{\mathrm{s}}, box \\ \quad ,\ l_{\mathrm{b}} = 10, \_l_{\mathrm{s}} = l_{\mathrm{s}} \\ \quad \mid\ *(\ p_2\,?\,box;\ \Delta(l_{\mathrm{b}} - \_l_{\mathrm{s}});\ s_1 := \mathrm{false}\ ) \\ \quad]\!] \\ \rangle \end{array}$$

In the process body of the first instantiation of control process C, first the conveyor is switched on ($v_0 := 1$), where v_0 is the actuator that determines the velocity of the first conveyor belt. The process subsequently waits until the sensor is off ($\neg s_0$, where $\neg$ means logical not). Initially, the conveyor is empty and the sensor is off. The process then waits until it can synchronize with the preceding control process by executing $pc_0\,?$. This means that a box may enter the conveyor. Subsequently, the process waits until the box has reached the sensor position (s_0) so that the sensor is on (value of s_0 equals true). Then the conveyor is switched off ($s_0 \rightarrow v_0 := 0$). Subsequently, the control process tries to synchronize with the next control process ($pc_1\,!$). After execution of the synchronization ($pc_1\,!$ in the first control process, simultaneous with $pc_1\,?$ in the second control process), the repetition is re-executed, and the conveyor is switched on again.

In the conveyor process V, variable x models the position of the front of the box on the conveyor, and variable *box* stores the identification number of the box. Identification number 0 means that there is no box on the conveyor. In that case, the value of variable x equals and remains -1 ($box = 0 \rightarrow \dot{x} = 0$. When there is a box on the conveyor, it moves with velocity v ($box \neq 0 \rightarrow \dot{x} = v$). The physical representation of variables l_b, l_c, and l_s is shown in Figure 4.15. The value parameters l_b, l_c, l_s in the process definition of V are defined as local discrete variables ($l_\mathrm{b}, l_\mathrm{c}, \_l_\mathrm{s}$) in the translation of the process instantiation, in accordance with the translation rules defined in Section 2.5.2. Value parameter l_s is renamed to $\_l_\mathrm{s}$ by prefixing it with an underscore, because it conflicts with the global variable l_s that is used in the process instantiations (e.g. $V(s_\mathrm{G}, s_0, v_0, p_0, p_1, 10, 15, l_\mathrm{s})$). The values of the local discrete variables l_b, l_c, and $\_l_\mathrm{s}$ are defined by initialization predicate $l_\mathrm{b} = 10, l_\mathrm{c} = 15, \_l_\mathrm{s} = l_\mathrm{s}$.

In the infinite repetition of the process body of the first instantiation of the conveyor process, the process starts by waiting until it can receive a box, and initializes the position of the box to 0 ($p_0\,?\,box;\ x := 0$). When $x = l_\mathrm{b} - \_l_\mathrm{s}$, the back end of the box has just passed the sensor of the previous "conveyor", which in this case is generator process G. Subsequently, the sensor of the previous conveyor (s_prev in the process definition of V, s_G in the first conveyor) is switched off. After that, the conveyor process waits until the box has reached the sensor of the conveyor ($x \geq l_\mathrm{c} - \_l_\mathrm{s}$), and the sensor is switched on ($s_0 :=$ true). Finally, when the box has reached the end of the conveyor ($x = l_\mathrm{c}$), the box is sent to the next conveyor ($p_1\,!\,box$), and the loop is re-executed. Simultaneously with execution of $p_1\,!\,box$ by the first conveyor, the second conveyor executes $p_1\,?\,box$. The box has now crossed the boundary of the two conveyors, and its position is registered by the second conveyor process.

The last conveyor process V_E is always on, meaning that the belt moves with velocity 1. Therefore, the duration of a box on this conveyor is given by $\Delta(l_\mathrm{b} - \_l_\mathrm{s})$.

An alternative specification of conveyor process V follows below. In this model, when a box enters a conveyor ($p_\mathrm{i}\,?\,box$), a new continuous variable x with initial value 0 is created by means of the scope operator $[\![\ \mathsf{icont}\ x,\ x = 0 \mid\ \cdots\ ]\!]$. The position of the box is defined by equation $\dot{x} = v$ until $x \geq l_\mathrm{b} - l_\mathrm{s}$. Then, sensor s_prev is switched off ($s_\mathrm{prev} :=$ false), which causes the alternative composition ($\dot{x} = v \ [\!]\ x \geq l_\mathrm{b} - l_\mathrm{s} \rightarrow s_\mathrm{prev} :=$ false) to terminate. The difference between this conveyor model and the previous model, is that in the previous

model, absence of a box was modeled by means of predicate $\dot{x} = 0$, whereas in the model below, absence of the conveyor means that variable x, which models the position of the front of the conveyor, does not exist.

$$
\begin{array}{l}
V(\mathsf{ext}\ s_{\mathrm{prev}}, s, v,\ \mathsf{chan}\ p_{\mathrm{i}}, p_{\mathrm{o}},\ \mathsf{val}\ l_{\mathrm{b}}, l_{\mathrm{c}}, l_{\mathrm{s}}) = \\
[\![\ \mathsf{disc}\ box \\
|\ *(\ p_{\mathrm{i}}\,?\,box \\
\quad ;\ [\![\ \mathsf{cont}\ x,\ x = 0 \\
\quad\quad |\ (\dot{x} = v\ [\!]\ x \geq l_{\mathrm{b}} - l_{\mathrm{s}} \rightarrow s_{\mathrm{prev}} := \mathrm{false}) \\
\quad\quad ;\ (\dot{x} = v\ [\!]\ x \geq l_{\mathrm{c}} - l_{\mathrm{s}} \rightarrow s := \mathrm{true}) \\
\quad\quad ;\ (\dot{x} = v\ [\!]\ x \geq l_{\mathrm{c}} \rightarrow p_{\mathrm{o}}\,!\,box) \\
\quad\quad]\!] \\
\quad) \\
]\!]
\end{array}
$$

4.11 Discrete-event model of a manufacturing line

A manufacturing line consists of a generator G, distributer D, two manufacturing cells C, and an assembling machine M_{A}. Figure 4.17 shows the iconic model of the manufacturing line. Processes R and E are added to obtain a closed system; they do not model actual behavior.

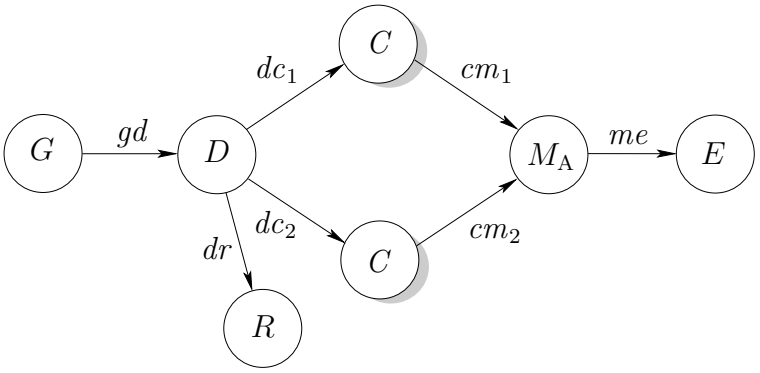

Figure 4.17: Iconic model of a manufacturing line.

The manufacturing line is modeled as follows, where t_{gen}, t_{out}, pt_{min1}, pt_{max1}, pt_{min2}, pt_{max2}, pt_{min3}, pt_{max3}, pt_{min4}, pt_{max4}, and pt denote constants.

$$
\begin{array}{l}
\langle\ G(gd, t_{\mathrm{gen}}) \\
\|\ D(gd, dc_1, dc_2, dr, t_{\mathrm{out}}) \\
\|\ R(dr) \\
\|\ C(dc_1, cm_1, pt_{\mathrm{min1}}, pt_{\mathrm{max1}}, N_1, pt_{\mathrm{min2}}, pt_{\mathrm{max2}})
\end{array}
$$

$$
\begin{array}{l}
\| \; C(dc_2, cm_2, pt_{\min 3}, pt_{\max 3}, N_2, pt_{\min 4}, pt_{\max 4}) \\
\| \; M_{\mathrm{A}}(cm_1, cm_2, me, pt) \\
\| \; E(me) \\
\rangle
\end{array}
$$

$$
G(\mathsf{chan}\ gd, \mathsf{val}\ t_{\mathrm{gen}}) = [\![\ \mathsf{disc}\ x,\ x = \mathrm{false} \mid *(\Delta t_{\mathrm{gen}};\ gd\,!\,x) \ ]\!]
$$

$$
\begin{array}{l}
D(\mathsf{chan}\ gd, dc_1, dc_2, dr, \mathsf{val}\ t_{\mathrm{out}}) = \\
[\![\ \mathsf{disc}\ x \\
\mid *(gd\,?\,x;\ (dc_1\,!\,x \ [\!] \ dc_2\,!\,x \ [\!] \ \Delta t_{\mathrm{out}};\ dr\,!\,x)) \\
]\!]
\end{array}
$$

$$
R(\mathsf{chan}\ dr) = [\![\ \mathsf{disc}\ x \mid *(dr\,?\,x) \ ]\!]
$$

$$
\begin{array}{l}
M_{\mathrm{A}}(\mathsf{chan}\ cm_1, cm_2, me, \ \mathsf{val}\ pt) = \\
[\![\ \mathsf{disc}\ x, y \\
\mid *((cm_1\,?\,x \ \| \ cm_2\,?\,y);\ \Delta pt;\ me\,!\,x) \\
]\!]
\end{array}
$$

$$
E(\mathsf{chan}\ me) = [\![\ \mathsf{disc}\ x \mid *(me\,?\,x) \ ]\!]
$$

Every t_{gen} time units (assuming $t_{\mathrm{gen}} \geq t_{\mathrm{out}}$, see process D), a product is generated by generator G. A product is modeled by a boolean variable x that is initially false. The boolean indicates whether the product has done the second round in a manufacturing cell C. A product enters the manufacturing line via channel gd. The distributor tries to send a product either via channel dc_1 or channel dc_2 . In case this is not possible within t_{out} time units, the product is rejected and sent to reject process R via channel dr ($dc_1\,!\,x \ [\!] \ dc_2\,!\,x \ [\!] \ \Delta t_{\mathrm{out}};\ dr\,!\,x$). Process R consumes the rejected products ($*(dr\,?\,x)$).

A manufacturing cell C, shown in Figure 4.18, consists of two machines (M_{rw}, M) and a N-place FIFO (first-in-first-out) buffer B.

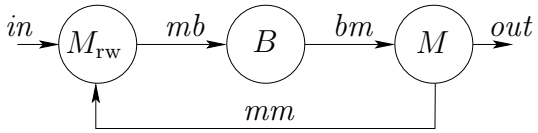

Figure 4.18: Iconic model of a manufacturing cell C.

The χ specification of the manufacturing cell is as follows:

$$
\begin{array}{l}
C(\mathsf{chan}\ in, out, \mathsf{val}\ pt_{\min 1}, pt_{\max 1}, N, pt_{\min 2}, pt_{\max 2}) \\
[\![\ \mathsf{chan}\ mb, bm, mm \\
\mid \; M_{\mathrm{rw}}(in, mb, mm, pt_{\min 1}, pt_{\max 1}) \\
\| \; B(mb, bm, N) \\
\| \; M(bm, out, mm, pt_{\min 2}, pt_{\max 2}) \\
]\!]
\end{array}
$$

Products enter the cell via channel in. The routing of a product in the manufacturing cell is as follows: M_{rw}, B, M, M_{rw}, B, M. Products leave the manufacturing cell via channel out. The process definitions of the buffer and the machines are given by

$$
\begin{array}{l}
B(\mathsf{chan}\ in, out, \mathsf{val}\ N) = \\
[\![\ \mathsf{disc}\ x \\
,\ xs = [\,] \\
\mid *(\ \mathrm{len}(xs) < N \rightarrow in\,?\,x;\ xs := xs +\!\!+ [x] \\
\quad [\!]\ \mathrm{len}(xs) > 0\ \rightarrow out\,!\,\mathrm{hd}(xs);\ xs :=\ \mathrm{tl}(xs) \\
\quad) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
M_{\mathrm{rw}}(\mathsf{chan}\ in, out, mm, \mathsf{val}\ pt_{\mathrm{min}}, pt_{\mathrm{max}}) = \\
[\![\ \mathsf{disc}\ x, pt \\
\mid *((in\,?\,x\ [\!]\ mm\,?\,x;\ x := \mathrm{true});\ pt : pt \in [pt_{\mathrm{min}}, pt_{\mathrm{max}}];\ \Delta pt;\ out\,!\,x) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
M(\mathsf{chan}\ in, out, mm, \mathsf{val}\ pt_{\mathrm{min}}, pt_{\mathrm{max}}) = \\
[\![\ \mathsf{disc}\ x, pt \\
\mid *(in\,?\,x;\ pt : pt \in [pt_{\mathrm{min}}, pt_{\mathrm{max}}];\ \Delta pt;\ (x \rightarrow out\,!\,x\ [\!]\ \neg x \rightarrow mm\,!\,x)) \\
]\!]
\end{array}
$$

The buffer can store up to N products, which are stored in a list xs ($\mathrm{len}(xs) < N \rightarrow in\,?\,x;\ xs := xs +\!\!+ [x]$), where $[x]$ denotes a list with one element x, and $+\!\!+$ denotes list concatenation. The empty list is denoted by [ ]. If the buffer is not empty, the first product in the buffer can be sent to the machine via channel out ($\mathrm{len}(xs) > 0 \rightarrow out\,!\,\mathrm{hd}(xs);\ xs :=\ \mathrm{tl}(xs)$), where $\mathrm{hd}(xs)$ denotes the first element (head) of list xs, and $\mathrm{tl}(xs)$ denotes the remainder (tail) of list xs without its first element. Machine M_{rw} receives products from channels in and mm. A product received from mm is assigned the value true, which indicates that this product is processed by machine M_{rw} for the second time ($in\,?\,x\ [\!]\ mm\,?\,x;\ x := \mathrm{true}$). The machine has a processing time between pt_{min} and pt_{max} time units ($pt : pt \in [pt_{\mathrm{min}}, pt_{\mathrm{max}}];\ \Delta pt$). Processed products are sent via channel out to the buffer B. Machine M receives products via channel in, and processes them for pt time units. Depending on the value of product variable x, the product is sent either via channel mm to machine M_{rw} (x equals false) or it leaves the manufacturing cell via channel out (x equals true) ($x \rightarrow out\,!\,x\ [\!]\ \neg x \rightarrow mm\,!\,x$).

After processing in one of the two manufacturing cells C, products are sent to machine M_{A}. Machine M_{A} waits to receive one product via channel cm_1, and one product via channel cm_2, in a non-deterministic order ($cm_1\,?\,x \parallel cm_2\,?\,y$). After processing these two products (Δpt), the combination of them leaves the manufacturing line via channel out ($out\,!\,x$). Process E consumes the processed products.

CHAPTER

FIVE

Translations between other formalisms and Chi

In this chapter, we investigate the connections between other formalisms and χ. One of the formalisms to describe hybrid systems are piecewise affine systems. General translation schemes from continuous-time piecewise affine systems and discrete-time piecewise affine systems to χ are defined, which show that these formalisms are closely related. Another formalism to describe hybrid systems is the theory of hybrid automata. Formal translations between the theory of hybrid automata and χ (in both directions) have been defined. The translation from hybrid automata to χ aims to show that the χ formalism is at least as expressive as the theory of hybrid automata. The translation from a reasonable subset (χ_{sub}) of χ to hybrid automata enables verification of χ_{sub} specifications using existing hybrid automata based verification tools. Furthermore, it is proved that any transition of a χ_{sub} specification can be mimicked by a transition in the corresponding hybrid automaton and vice versa, which indicates that the translation as defined in this chapter is correct.

5.1 Translations of piecewise affine systems to Chi

In this section, two general translation schemes are given. One scheme defines the translation of continuous-time piecewise affine systems to a χ specification. The other scheme defines a translation of discrete-time piecewise affine systems to χ.

5.1.1 Continuous-time PWA

Continuous-time piecewise affine systems are described by N systems of affine differential equations

$$\begin{array}{rcl} \dot{x}(t) & = & A_i x(t) + B_i u(t) + f_i \\ y(t) & = & C_i x(t) + D_i u(t) + g_i \end{array} \qquad \text{for} \quad \left[\begin{array}{c} x(t) \\ u(t) \end{array} \right] \in \Omega_i \ ,$$

where i ($i = 1, \ldots, N$) is the number of the mode. Each mode i is defined in a region Ω_i, which is a convex polyhedron, given by a finite number of linear inequalities, in the input/state space. Here, $u(t) \in \mathbb{R}^m$, $x(t) \in \mathbb{R}^n$, and $y(t) \in \mathbb{R}^l$ denote the input, state and output, respectively, at time t. Furthermore, f_i, and g_i denote constants. In each mode, the trajectories of the state variables x are continuous functions of time. The trajectories of the input/output variables in a mode may be discontinuous functions of time.

Continuous-time PWA systems using the Caratheodory solution concept, can be translated to χ as follows:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x,\ \mathsf{alg}\ y \\
,\ x = x_0 \\
|\quad (\Omega_1 \Rightarrow \dot{x} = A_1 x + B_1 u + f_1,\ y = C_1 x + D_1 u + g_1) \\
\quad \wedge \\
\quad \vdots \\
\quad \wedge\ (\Omega_N \Rightarrow \dot{x} = A_N x + B_N u + f_N,\ y = C_N x + D_N u + g_N) \\
\rangle
\end{array}
$$

The state variables x are modeled in χ by means of (non-jumping) continuous variables, with initial value x_0. The output variables y are modeled by means of algebraic variables. The behavior of u is not specified, as in the original PWA model. In the χ model, u could denote a function of time, or u could be defined as an algebraic variable, and additional equations specifying the behavior of u could be added. The behavior associated to a mode i is described by means of a delay predicate $(\Omega_i \Rightarrow \dot{x} = A_i x + B_i u + f_i,\ y = C_i x + D_i u + g_i)$.

5.1.2 Discrete-time PWA

Discrete-time PWA systems are described by

$$
\begin{array}{rcl}
x(k+1) & = & A_i x(k) + B_i u(k) + f_i \\
y(k) & = & C_i x(k) + D_i u(k) + g_i
\end{array}
\qquad \text{for} \quad
\left[\begin{array}{c} x(k) \\ u(k) \end{array} \right] \in \Omega_i,
$$

$i = 1, \ldots, N$. Here, $u(k) \in \mathbb{R}^m$, $x(k) \in \mathbb{R}^n$, and $y(k) \in \mathbb{R}^l$ denote the input, state and output, respectively, at the k'th time-point. Discrete-time PWA systems can be translated to χ as follows:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ x, y, k \\
,\ x = x_0, k = 0 \\
|\ *(\ (\ \Omega_1 \rightarrow \langle x, y \rangle := \langle A_1 x + B_1 u.k + f_1, C_1 x + D_1 u.k + g_1 \rangle \\
\qquad [\!] \\
\qquad \vdots \\
\qquad [\!]\ \Omega_N \rightarrow \langle x, y \rangle := \langle A_N x + B_N u.k + f_N, C_N x + D_N u.k + g_N \rangle \\
\qquad) \\
\quad ;\ k := k + 1 \\
\quad) \\
\rangle
\end{array}
$$

The state variables x are modeled in χ by means of discrete variables, and are initialized to x_0. The output variables y and variable k are also modeled by means of discrete variables. We assume u to denote an array of points, such that $u.i$ denotes the value of u at the i'th time-point. In the repetition $*(\quad)$, the state and output variables are assigned

new values according to one of the modes. Subsequently, k is increased by one. The behavior associated to a mode is described by means of a multiple assignment $\langle x, y\rangle := \langle A_i x + B_i u.k + f_i, C_i x + D_i u.k + g_i\rangle$. The alternative composition of the behavior of the modes allows the state and output variables to be assigned new values according to the mode for which the corresponding guard (Ω_i) holds.

Example: Integrator An integrator with upper saturation can be modeled as a discrete-time PWA model as follows:

$$\begin{aligned} x(k+1) &= \begin{cases} x(k)+u(k) & \text{if } x(k)+u(k) \le 1 \\ 1 & \text{if } x(k)+u(k) \ge 1 \end{cases} \\ y(k) &= x(k) \end{aligned}$$

This model can be translated to χ as follows:

$\langle$ disc x, y, k
, $x = x_0, k = 0$
| $*($ ($x + u.k \le 1 \rightarrow x, y := x + u.k, x$
[] $x + u.k \ge 1 \rightarrow x, y := 1, x$
)
; $k := k + 1$
)
$\rangle$

5.2 Translation of a hybrid automaton to Chi

In this section, the hybrid automaton model of [Hen00b] is related to the χ formalism.

5.2.1 Description hybrid automaton

A hybrid automaton [Hen00b] consists of the following components:

- A finite set of (real-valued) variables $X = \{x_1, \ldots, x_n\}$, the set $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ which denotes the first derivatives of these variables, and the set $X' = \{x'_1, \ldots, x'_n\}$ which denotes the primed variables that represent values at the conclusion of a discrete change.
- A finite directed multi-graph (V, E), where V denotes a set of vertices (control modes) and E denotes a set of edges (control switches).
- Three vertex labeling functions init, inv, and flow that assign to each control mode $v \in V$ a predicate for initial, invariant and flow conditions, respectively. The free variables of the initial and invariant predicates are from X. The free variables of the flow predicates are from $X \cup \dot{X}$.

- An edge labeling function jump, that assigns to each edge $e \in E$, a jump condition which is a predicate whose free variables are from $X \cup X'$.
- A finite set Σ of events, and an edge labeling function event : $E \to \Sigma$ that assigns to each edge an event.

In order to translate a hybrid automaton to χ, two additional functions are defined on a hybrid automaton: function edges $\in V \to \mathcal{P}(E)$ returns a set of outgoing edges for a location, and function target $\in E \to V$ returns the target vertex of an edge. Furthermore, the function $\mathcal{T}$ translates a jump predicate to the predicate r of a χ action predicate ($W : r \gg l_{\mathrm{a}}$) by renaming variables occurring without a prime in a jump predicate to variables with superscript '$^-$', and renaming variables occurring with a prime '$'$' to variables without the prime. E.g. $\mathcal{T}(x' = 2x + y \wedge x \geq 0 \wedge y' = y)$ becomes $x = 2x^- + y^- \wedge x^- \geq 0 \wedge y = y^-$. In the latter expression, x^- and y^- refer to the values of x and y, respectively, before the discrete jump, and x and y refer to the value of variables x and y after the discrete jump. The class of hybrid automata to be translated to χ is restricted to the hybrid automata without initial time non-determinism. In this section, we consider hybrid automata where the initial condition of all but one control modes equals false. The one control mode with the initial condition that may be not equal to false is called the *initial* control mode.

Furthermore, it should be possible to rewrite each flow predicate into one of the following forms: $\dot{\mathbf{x}} = f(\mathbf{x})$, $\dot{\mathbf{x}} \in f(\mathbf{x})$ or the predicate true. This means that we do not consider flow predicates such as false, or $\dot{x} = 0 \wedge \dot{x} = 1$.

5.2.2 Translation scheme

Consider a hybrid automaton model which belongs to the class of automata as defined in the previous section, with n variables ($X = \{x_1, \ldots, x_n\}$), k control modes ($V = \{v_1, \ldots, v_k\}$), and one initial control mode v_1. The translation to a corresponding χ specification is defined as follows:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x_1, \ldots, x_n \\
,\ \mathrm{init}(v_1) \\
,\ v_1 \mapsto \mathrm{flow}(v_1) \wedge \mathrm{inv}(v_1)\ [\!] \\
\qquad\quad ([\!]_{e:e \in \mathrm{edges}(v_1)}\ [\emptyset : \mathcal{T}(\mathrm{jump}(e)) \gg \mathrm{event}(e)];\ \mathrm{target}(e)) \\
\quad \vdots \\
,\ v_k \mapsto \mathrm{flow}(v_k) \wedge \mathrm{inv}(v_k)\ [\!] \\
\qquad\quad ([\!]_{e:e \in \mathrm{edges}(v_k)}\ [\emptyset : \mathcal{T}(\mathrm{jump}(e)) \gg \mathrm{event}(e)];\ \mathrm{target}(e)) \\
\\
|\ (\mathsf{jump}\ x_1, \ldots, x_n \mid v_1) \\
\rangle
\end{array}
$$

The variables $x_1, \ldots, x_n$ are declared as continuous variables. These variables are initialized by means of initialization predicate init(v_1). By means of the jump enabling operator

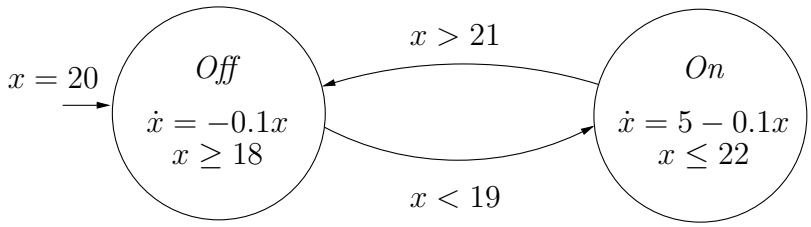

Figure 5.1: A hybrid automaton model of a thermostat

process term (jump $x_1, \ldots, x_n \mid \ldots$), all variables become jumping. I.e., in principle they may change arbitrarily during action transitions.

A vertex v_i of the hybrid automaton model is translated using a corresponding recursion variable v_i in the χ model. The process term associated with this recursion variable consists of the alternative composition of the process term describing the continuous behavior of the vertex, and the alternative compositions of all individual process terms of the outgoing edges of this vertex. Below, these process terms are explained in more detail.

The continuous behavior of a vertex v_i is translated to a delay predicate in χ, consisting of the conjunction of the flow predicate and the invariant of the vertex. For each outgoing edge, the jump predicate of that edge is translated to an action predicate labelled with the event label of the edge ($\emptyset : \mathcal{T}(\text{jump}(e)) \gg \text{event}(e)$). Since all variables are allowed to jump, the set W of the action predicate equals the empty set. The semantics of a hybrid automaton is such that when a guard of an edge is enabled, the transition via this edge can be taken, but it is not required to take this transition. Therefore, the action predicate associated with the edge is made delayable using the any delay operator [ ]. After the transition, the behavior is specified by the recursion variable associated with the target vertex (target(e)).

Note that for set $E = \{e_1, \ldots, e_k\}$, notation $[]_{e:e\in E}\, [\emptyset : \mathcal{T}(\text{jump}(e)) \gg \text{event}(e)]$; target($e$), denotes the process term $[\emptyset : \mathcal{T}(\text{jump}(e_1)) \gg \text{event}(e_1)]$; target($e_1$) $[]$ ... $[]$ $[\emptyset : \mathcal{T}(\text{jump}(e_k)) \gg \text{event}(e_k)]$; target($e_k$).

This straightforward translation of a hybrid automaton to a χ model shows that χ is expressive enough to model phenomena that are usually studied by means of a hybrid automaton. The translation scheme is illustrated by means of an example in the next section.

5.2.3 A thermostat

This example shows the translation of a hybrid automaton model of a thermostat to χ. The hybrid automaton is shown in Figure 5.1. Variable x represents the temperature. The control modes are *On* and *Off*. Initially, the temperature equals 20 degrees, and the heater is off (control mode *Off*). The temperature falls according to the flow condition $\dot{x} = -0.1x$. According to the jump condition $x < 19$, the heater may go on as soon as the temperature falls below 19 degrees. The invariant condition $x \geq 18$ ensures that at the latest the heater

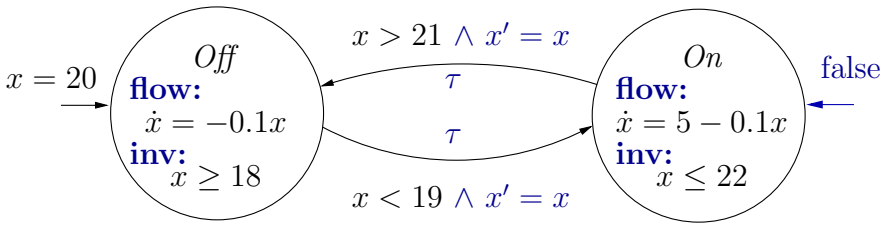

Figure 5.2: Complete hybrid automaton model of a thermostat

will go on when the temperature equals 18 degrees. In the control mode On, the heater is on, and the temperature rises according to the flow condition $\dot{x} = 5 - 0.1x$. When the temperature rises above 21 degrees, the heater may turn off. Due to the invariant condition $x \leq 22$, at the latest the heater will turn off when the temperature equals 22 degrees.

Figure 5.1 is taken from [Hen00b], where the usual informal notation is used: events on the edges are ignored, and the initial and jump conditions are incomplete. In particular, in Figure 5.1 both edges should have an event label, the initial condition of mode On equals false, and the jump conditions of the edges should have been $x < 19 \wedge x' = x$ and $x > 21 \wedge x' = x$, respectively. The complete, formal hybrid automata model of the thermostat is shown in Figure 5.2. Using the translation scheme, this model is translated to χ, which results in the following χ specification:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x \\
,\ x = 20 \\
,\ \mathit{Off}\ \mapsto\ \dot{x} = -0.1x \wedge x \geq 18\ [\!]\ [\emptyset : x < 19 \wedge x = x^{-} \gg \tau];\ \mathit{On} \\
,\ \mathit{On}\ \mapsto\ \dot{x} = 5 - 0.1x \wedge x \leq 22\ [\!]\ [\emptyset : x > 21 \wedge x = x^{-} \gg \tau];\ \mathit{Off} \\
|\ (\mathsf{jump}\ x \mid \mathit{Off}) \\
\rangle\ .
\end{array}
$$

Since the value of variable x does not change during action transitions, the model can be simplified to

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x \\
,\ x = 20 \\
,\ \mathit{Off}\ \mapsto\ \dot{x} = -0.1x \wedge x \geq 18\ [\!]\ [\emptyset : x < 19 \gg \tau];\ \mathit{On} \\
,\ \mathit{On}\ \mapsto\ \dot{x} = 5 - 0.1x \wedge x \leq 22\ [\!]\ [\emptyset : x > 21 \gg \tau];\ \mathit{Off} \\
|\ \mathit{Off} \\
\rangle\ .
\end{array}
$$

5.3 Translation of Chi to hybrid automata

In this section, the translation from χ to hybrid automata is defined.

Many different hybrid automata definitions exist. Some definitions require solutions for the continuous variables to be differential functions, e.g. [Hen00b, AHH96]. Other definitions allow the more general case of piecewise differential functions, e.g. [vdSS00]. Most hybrid automata definitions do not define urgent transitions, or they define urgent transitions in a restrictive way, as in [HHWT95]. In [NOSY92], urgent transitions are defined in a general way, using a predicate that defines the maximum sojourn time in a location, but instead of invariants and flow clauses, evolution functions are used. With respect to the meaning of jump clauses, that define the behavior of the variables in action transitions, differences also occur: where in [Hen00b] the variables can in principle perform arbitrary jumps unless restricted by the jump predicate, in [HHWT95], variables in principle remain unchanged unless changes are enforced by the jump predicate.

None of these hybrid automata definitions is expressive enough to be used as the target for the translation of hybrid χ. Therefore, the translation uses a target hybrid automata definition, which will be defined in Section 5.3.2, that uses features from different hybrid automata definitions. In particular, the definition of the jump predicate in combination with a set of changeable variables is based on [AHH96], the solution concept that allows piecewise differentiable functions is based on [vdSS00], and the definition of urgent transitions was inspired by [NOSY92].

The translation is defined only for a subset of χ. For instance, the guard operator is not translated as an operator, but the guarded atomic process terms are translated. Also the scope operators of χ are not translated.

This section is organized as follows: In Section 5.3.1, the subset of χ to be translated is defined. Section 5.3.2 presents the syntax and semantics of the hybrid automata definition. The translation from the subset of χ these hybrid automata is defined in Section 5.3.3. In Section 5.3.4, its is proved that any transition of a χ model can be mimicked by a transition in the corresponding hybrid automaton and vice versa. This indicates that the translation is correct. The translation is illustrated by means of an example of a bottle filling system in Section 5.3.5.

5.3.1 The χ_{sub} language

The subset χ_{sub} of the χ language that is translated consists of processes $\langle p, \sigma, (\text{dom}(\sigma) \setminus \{\mathsf{time}\}, J, \emptyset, H, \emptyset)\rangle$, where $p \in P_{\text{sub}}$ consists of the guarded atomic process terms: guarded action predicate $b \rightarrow W : r \gg l_{\text{a}}$, guarded send $b \rightarrow h\,!!\,\mathbf{e}_n$, guarded receive $b \rightarrow h\,??\,\mathbf{x}_n$, delay predicate u, consistent deadlock process term δ, and guarded inconsistent process term $b \rightarrow \bot$, the unary operators the any delay $[\,]$, repetition $*$, encapsulation $\partial_A(\,)$, urgent communication $\upsilon_H(\,)$ and jump enabling ι_{J^+}, and the binary operators sequential composition $;$, alternative composition $[\!]$, and parallel composition $\|$. Formally, P_{sub} is defined by:

$$\begin{aligned} P_{\text{sub}} ::= \;& u \mid \delta \mid b \rightarrow \bot \mid b \rightarrow W : r \gg l_{\text{a}} \mid b \rightarrow h\,!!\,\mathbf{e}_n \mid b \rightarrow h\,??\,\mathbf{x}_n \\ & \mid [P_{\text{sub}}] \mid *P_{\text{sub}} \mid \iota_{J^+}(P_{\text{sub}}) \mid P_{\text{sub}};\, P_{\text{sub}} \mid P_{\text{sub}} \,[\!]\, P_{\text{sub}} \mid P_{\text{sub}} \,\|\, P_{\text{sub}} \\ & \mid \partial_A(P_{\text{sub}}) \mid \upsilon_H(P_{\text{sub}}) \end{aligned}$$

In χ_{sub} processes, there are no discrete variables ($\text{dom}(\sigma) = C \cup \{\mathsf{time}\}$), no algebraic variables ($L = \emptyset$), and no recursion variables ($R = \emptyset$). This subset is, in addition to the restrictions in Chapter 2, further restricted such that dotted continuous variables are not allowed to occur in action predicates or guards. This restriction is in order to simplify some proofs. Furthermore, dotted continuous variables are not allowed to occur in the expressions of $h\,!!\,\mathbf{e}_n$ (for simplicity).

In χ, the guard operator can be applied to arbitrary process terms. Since it is not possible to translate the guard operator in a general way, the process terms to which the guard operator can be applied are restricted to the inconsistent process term, the action predicate, undelayable send and undelayable receive process terms.

In Chapter 2, the semantics of the repetition operator is defined in terms of the recursion scope operator. For simplicity, in Appendix C.1, the deduction rules for the repetition operator are given, such that when using these rules, the following property holds $*p \leftrightarrows [\![_{\text{R}}\,\{X \mapsto p;\, X\} \mid X\,]\!]$. The semantics of the solution function Ω of χ_{sub} equals the solution function Ω_{FG} of χ. In χ_{sub}, function Ω is defined as $\Omega_{FG} \in \Sigma \times \mathcal{P}(\mathcal{V}) \times \mathcal{P}(\dot{\mathcal{V}}) \times U \times T \to \mathcal{P}(T \mapsto \dot{\Sigma})$, where U denotes the set of all predicates over $\mathcal{V}$ and $\dot{\mathcal{V}}$, returns a set of trajectories from time to an extended valuation for the variables and dotted variables, given a valuation representing the current values of continuous variables, the set of continuous variables, a delay predicate and a time point that denotes the duration of the trajectory. Formally, function Ω is defined as:

$$
\begin{array}{l}
\Omega(\sigma, C, \emptyset, u, t) = \\
\{\, \rho \\
\mid\ \rho \in [0,t] \to ((\text{dom}(\sigma) \cup \dot{C}) \to \Lambda) \\
,\ t \geq 0 \\
,\ \forall_{s \in [0,t]} : \quad \rho(s) \models u \\
,\ \forall_{x \in \text{dom}(\sigma)} : \quad (\rho \downarrow x)(0) = \sigma(x) \\
,\ \forall_{x \in C} : \quad \rho \downarrow \dot{x} \text{ is an integrable function in the} \\
\qquad\qquad\quad \text{Lesbesgue sense.} \\
,\ \forall_{s \in [0,t], x \in C} : \quad (\rho \downarrow x)(s) = (\rho \downarrow x)(0) + \int_0^s (\rho \downarrow \dot{x})(s')ds' \\
,\ \forall x \in C : \quad (\rho \downarrow x, \rho \downarrow \dot{x}) \in G \\
,\ \forall_{s \in [0,t]} : \quad \rho(s)(\mathsf{time}) = \sigma(\mathsf{time}) + s \\
\}
\end{array}
$$

We do not further explain the meaning of the solution function Ω, because it is defined based on the solution function Ω_{FG}, which is already explained in detail in Chapter 3. The way we obtain the solution function Ω from the solution function Ω_{FG} is by filling in the set L, which is empty, and restricting the $\text{dom}(\sigma)$ in such a way that it does not contain any discrete variable. This corresponds to the restriction on χ_{sub}.

5.3.2 Hybrid automata definition

In this section, the syntax and semantics of hybrid automata is given.

5.3.2.1 Syntax

A hybrid automaton *HA* consists of the following components:

- A finite set of (real-valued) variables $X = \{x_1, \ldots, x_n\}$, the set $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ which denotes the first derivatives of the variables w.r.t. time, and the set $X' = \{x'_1, \ldots, x'_n\}$ which denotes the primed variables that represent values at the conclusion of a discrete change.
- A finite directed multi-graph (V, E), where V denotes a set of vertices (also referred to as control modes or locations) and E denotes a set of edges (control switches).
- Three vertex labeling functions init, inv, and flow that assign to each location $v \in V$ a predicate for initial conditions, invariants and flow conditions, respectively. The free variables of the initial and invariant predicates are from X. The free variables of the flow predicates are from $X \cup \dot{X}$.
- An edge labeling function jump, that assigns to each edge $e \in E$ a set of variables ($\subseteq X$) which are allowed to change and a jump condition which is a predicate whose free variables are from $X \cup X'$.
- An edge labeling function guard, that assigns to each edge $e \in E$ a guard which is a predicate whose free variables are from X.
- An edge labeling function urgent $\in E \to \{\text{true}, \text{false}\}$, that assigns to each edge a boolean: true for an urgent edge, and false for a non-urgent edge.
- A finite set Σ of events, and an edge labeling function event $\in E \to \Sigma$ that assigns to each edge an event.

Usually, an edge e is represented as $e = (v, v')$, which identifies a source location $v \in V$ and a target location $v' \in V$. This representation cannot be used in case of multi-edges (multiple edges with the same source location and target location). To deal with these, two additional functions are defined: function source $\in E \to V$ returns the source location of a given edge, and function target $\in E \to V$ returns the target location of a given edge. This results in the following hybrid automata definition: $HA = (X, V, \text{init}, \text{inv}, \text{flow}, E, \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \Sigma, \text{event})$.

5.3.2.2 Semantics

The semantics of a hybrid automaton is defined in terms of a timed transition system. In this transition system, two kinds of transition relations are defined: action transitions and time transitions.

- an action transition is labelled with an action label from an action label set to indicate that the transition may take place on performing that action;

- a time transition is labelled with an non-negative real number to indicate that the transition takes place on idling for that number of time units.

Let $HA = (X, V, \text{init}, \text{inv}, \text{flow}, E, \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \Sigma, \text{event})$ be a hybrid automaton. Then a *state* of HA is a pair $(v, \alpha) \in V \times (X \mapsto \Lambda)$. A state (v, α) of HA is *admissible* if $\alpha \models \text{inv}(v)$, and a state (v, α) of HA is *initial* if it is admissible and $\alpha \models \text{init}(v)$. Here, notation $\alpha \models \varphi$ denotes the truth value obtained by evaluating the predicate φ in α, i.e. replacing in φ all occurrences of all variables $x \in X$ by $\alpha(x)$.

The transition system interpretation of HA, written $[\![HA]\!]$, is the timed transition system $(Q, Q^0, \Sigma, \rightarrow, \longmapsto)$, where

- Q is the set of admissible states of HA;
- Q^0 is the set of initial states of HA;
- Σ is the set of events;
- $\rightarrow \subseteq Q \times \Sigma \times Q$ is the action transition relation. For $l \in \Sigma$, $(v, \alpha), (v', \alpha') \in Q$,

$$(v, \alpha) \xrightarrow{l} (v', \alpha') \Leftrightarrow \exists_{e \in E} \left(\begin{array}{l} \text{source}(e) = v, \text{target}(e) = v', \text{event}(e) = l, \\ \alpha \models \text{guard}(e), (\alpha, \alpha') \models \text{jump}(e) \end{array} \right)$$

- $\longmapsto \subseteq Q \times \mathbb{R}_{\geq 0} \times Q$ is the time transition relation. For $r \in \mathbb{R}_{\geq 0}$, $(v, \alpha), (v', \alpha') \in Q$: $(v, \alpha) \xmapsto{r} (v', \alpha')$ iff
  - $v = v'$,
  - $\exists_{\rho:[0,r] \rightarrow (X \cup \dot{X} \mapsto \Lambda)}$ such that
    - $\rho_\alpha(0) = \alpha, \rho_{\alpha'}(r) = \alpha'$,
    - $\forall_{t \in [0,r]} \left\{ \begin{array}{l} \rho(t) \models \text{inv}(v) \wedge \text{flow}(v) \\ \forall_{x \in X} \, (\rho \downarrow x)(t) = (\rho \downarrow x)(0) + \int_0^t (\rho \downarrow \dot{x})(s) ds \end{array} \right.$
    - $\forall_{e \in E} \, (\text{source}(e) = v \wedge \text{urgent}(e)) \implies \forall_{t \in [0,r)} \, \rho(t) \models \neg \text{guard}(e)$,

where notation $(\alpha, \alpha') \models \text{jump}(e)$ is defined as follows: Let $\text{jump}(e) = (W, \text{pred})$, then notation $(\alpha, \alpha') \models \text{jump}(e)$ is defined as $(\alpha, \alpha') \models \text{pred} \wedge \alpha \restriction (\text{dom}(\alpha) \setminus W) = \alpha' \restriction (\text{dom}(\alpha') \setminus W)$, where notation $(\alpha, \alpha') \models \varphi$ denotes the truth value obtained by evaluating the predicate φ by replacing in φ all occurrences of all variables $x \in X$ by $\alpha(x)$, and by replacing in φ all occurrences of all variables $x \in X'$ by $\alpha'(x)$.

5.3.3 The translation

The translation function $\mathcal{HA} \in \chi_{\text{sub}} \rightarrow \text{HA}$ is defined in terms of function $\mathcal{T} \in \mathcal{P}(\mathcal{V}) \times P \rightarrow HA_{\text{frag}}$ that translates a χ_{sub} process term p with a set of jumping continuous variables J to a corresponding hybrid automaton fragment HA_{frag}. Function $\mathcal{T}$ is further defined below.

Hybrid automaton fragment A hybrid automaton fragment HA_{frag} is a tuple (V, v_0, inv, flow, done, E, source, target, urgent, guard, jump, Σ, event), where V, inv, flow, E, source, target, guard, urgent, jump, Σ, and event are defined as in the hybrid automation definition. Location $v_0 \in V$ is the initial location of the hybrid automaton fragment, and function done $\in V \to \{\text{true}, \text{false}\}$ assigns to each location $v \in V$ a status (also known as done condition) that partitions the locations into terminating locations (status is true), and non-terminating locations (status is false). The distinction between terminating and non-terminating locations is needed in the definition of the translation of some χ_{sub} operators (e.g. sequential composition and repetition). Note that hybrid automaton fragment is defined at the level of χ_{sub} process terms, there is no transition system generated by a hybrid automaton fragment.

Auxiliary functions on hybrid automaton fragments In the translation of hybrid chi process terms with a set of jumping continuous variables to hybrid automaton fragments, we frequently combine hybrid automaton fragments for which the sets of node names are not disjoint and for which the sets of edge names are not disjoint. This presents us with a technical problem in case we simply wish to use such nodes in the combination. To overcome this technicality we introduce functions L_* and R_* that rename nodes and edges in such a way that for any two hybrid automata fragments HA_p and HA_q we have that the nodes of $L_*(HA_p)$ and $R_*(HA_q)$ are disjoint and that the edges of these hybrid automaton fragments are disjoint.

Let $HA = (V, v_0, \text{inv}, \text{flow}, \text{done}, E, \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \Sigma, \text{event})$ be a hybrid automaton fragment. Then we define $L_*(HA) = (V_l, v_{l0}, \text{inv}_l, \text{flow}_l, \text{done}_l, E_l, \text{source}_l, \text{target}_l, \text{urgent}_l, \text{guard}_l, \text{jump}_l, \Sigma_l, \text{event}_l)$ where $V_l = \{(l, v) \mid v \in V\}$, $v_{l0} = (l, v_0)$, for all $v \in V$, $\text{inv}_l(l, v) = \text{inv}(v)$, $\text{flow}_l(l, v) = \text{flow}(v)$, and $\text{done}_l(l, v) = \text{done}(v)$, $E_l = \{(l, e) \mid e \in E\}$, for all $e \in E$, $\text{source}_l(l, e) = (l, \text{source}(e))$, $\text{target}_l(l, e) = (l, \text{target}(e))$, $\text{urgent}_l(l, e) = \text{urgent}(e)$, $\text{guard}_l(l, e) = \text{guard}(e)$, $\text{jump}_l(l, e) = \text{jump}(e)$, $\Sigma_l = \Sigma$ and $\text{event}_l(l, e) = \text{event}(e)$. The function R_* is defined similarly.

Graphical representation In our graphical representation of hybrid automaton fragments, only the initial location of the hybrid automaton fragment has an incoming arrow. The terminating locations are drawn with double circles. Single arrows represent non-urgent edges, and double arrows represent urgent edges.

Variables used in the hybrid automaton A channel has a type. This defines a number of expressions to be sent or a number of variables to be received in (e.g. $h\,!!\,1, 2$ and $h\,??\,x, y$) via a channel. We refer to the number of expressions to be sent or the number of variables to be received via a channel as the number of arguments of the channel. For simplicity, the set of channel names H with the number of arguments of each channel (denoted by $\text{ar}(h)$ for $h \in H$) which is used in the χ specification under consideration is assumed. Using this set, valuation σ and the set of continuous variables C, the set of variables of the hybrid automaton is defined as $X = \text{dom}(\sigma) \cup DC \cup H_{\text{var}}$. The set of variables DC consists of

the variables of C prefixed with $'d'$: $DC = \{dc \mid c \in C\}$. The set H_{var} consists of $\text{ar}(h)$ additional variables for each channel from H: $H_{\text{var}} = \{h'_1, \cdots, h'_{\text{ar}(h)} \mid h \in H\}$. It is assumed that the sets $\text{dom}(\sigma)$, DC, and H_{var} are pairwise disjoint.

Translation function $\mathcal{HA}$ Function $\mathcal{HA}$ is now defined as follows: Let $\mathcal{T}_J(p) = (V_p, v_{0_p}, \text{inv}_p, \text{flow}_p, \text{done}_p, E_p, \text{source}_p, \text{target}_p, \text{urgent}_p, \text{guard}_p, \text{jump}_p, \Sigma_p, \text{event}_p)$ be the hybrid automaton fragment of p with the set of jumping continuous variables J, and X as defined before. Then the hybrid automaton corresponding to the χ_{sub} process $\langle p, \sigma, (C, J, \emptyset, H, \emptyset)\rangle$ is $\mathcal{HA}(\langle p, \sigma, (C, J, \emptyset, H, \emptyset)\rangle) = (X, V_p, \text{init}, \text{inv}_p, \text{flow}, E_p, \text{source}_p, \text{target}_p, \text{urgent}_p, \text{guard}_p, \text{jump}_p, \Sigma_p, \text{event}_p)$, where

$$\begin{aligned} \forall_{v \in V_p}: \quad & \text{init}(v) = \begin{cases} \mathcal{M}(\sigma) & \text{if } v = v_{0_p}, \\ \text{false} & \text{otherwise}, \end{cases} \\ \forall_{v \in V_p}: \quad & \text{flow}(v) = \text{flow}_p \wedge \dot{\mathsf{time}} = 1. \end{aligned}$$

Function $\mathcal{M}$ maps a valuation $\{x_0 \mapsto c_0, \ldots, x_n \mapsto c_n\}$ to a predicate $x_0 = c_0 \wedge \cdots \wedge x_n = c_n$. E.g. $\mathcal{M}(\{x \mapsto 1, y \mapsto 2\}) = (x = 1 \wedge y = 2)$.

5.3.3.1 Translation of atomic process terms of χ_{sub}

In this section, the translation of the atomic process terms of χ_{sub} to the corresponding hybrid automaton fragments is defined. Notation X_{aux} is defined as $X_{\text{aux}} = DC \cup H_{\text{var}}$.

Delay predicate u A delay predicate u restricts the allowed behavior of the variables in such a way that the value of the predicate remains true over time. Since u can only perform time transitions, $\mathcal{T}_J(u)$ has only one location without outgoing edges.

$$\mathcal{T}_J(u) = (\{v_0\}, v_0, \text{inv}, \text{flow}, \text{done}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset),$$

where $\text{inv}(v_0) = u[DC/\dot{C}]$, $\text{flow}(v_0) = u$, and $\text{done}(v_0) = \text{false}$. Predicate $u[DC/\dot{C}]$ is the predicate u where all occurrences of $\dot{c}$ are replaced by a variable dc from DC. E.g. $(\dot{x} = -x + 1 \wedge x \geq 0 \wedge y \in [0, 1])[DC/\dot{C}] = (dx = -x + 1 \wedge x \geq 0 \wedge y \in [0, 1])$.

In χ, it is not possible to reach a state in which the delay predicate evaluates to false, while in hybrid automata, it is possible to reach a state in which the flow condition does not hold. For example, in the semantics of χ_{sub}, the delay predicate $\dot{x} = 0 \wedge \dot{x} = 1$ denotes an inconsistent process, i.e., a process that cannot be reached. To overcome this semantical difference, the invariant is used to prevent entrance in case there is no solution for the delay predicate. As invariants cannot contain dotted variables, these dotted variables are replaced by variables from DC ($\text{inv}(v_0) = u[DC/\dot{C}]$).

Consistent deadlock δ and guarded inconsistent process term $b \to \bot$ Since process term δ can neither perform any action transitions, nor time transitions, $\mathcal{T}_J(\delta)$ has one location with flow condition false, invariant true, and no outgoing edges.

$$\mathcal{T}_J(\delta) = (\{v_0\}, v_0, \text{inv}, \text{flow}, \text{done}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset),$$

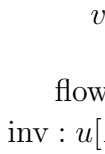

Figure 5.3: Hybrid automaton fragment for delay predicate u.

where $\text{inv}(v_0) = \text{true}$, $\text{flow}(v_0) = \text{false}$, and $\text{done}(v_0) = \text{false}$.

Figure 5.4: Hybrid automaton fragments for δ and $b \rightarrow \bot$.

Let b denote a guard, and $b \rightarrow \bot$ denote a guarded inconsistent process term $b \rightarrow \bot$. We know (see Proposition 3.5.4) that the guarded inconsistent process term $b \rightarrow \bot$ is equivalent to the negation of the guard b. Then

$$\mathcal{T}_J(b \rightarrow \bot) = \mathcal{T}_J(\neg b).$$

Since δ cannot perform any kind of transitions, the process term $b \rightarrow \delta$ can only perform an arbitrary time transition while its guard is false. This process term is not useful for the purpose of modeling. Hence, the process term $b \rightarrow \delta$ is not translated.

Guarded action predicate Let b denote a guard and let $W : r \gg l_{\text{a}}$ denote an action predicate. Then guarded action predicate $b \rightarrow W : r \gg l_{\text{a}}$ behaves as the action predicate $W : r \gg l_{\text{a}}$ if $b = \text{true}$. If $b = \text{false}$, it can perform arbitrary time transitions. An action predicate $W : r \gg l_{\text{a}}$ allows instantaneous changes to the variables from the set $W \cup J$ in such a way that the predicate r is satisfied. The values of variables not in the set $W \cup J$ remain unchanged.

$$\begin{aligned}\mathcal{T}_J(b \rightarrow W : r \gg l_{\text{a}}) \;=\; & (\{v_0, v_1\}, v_0, \text{inv}, \text{flow}, \text{done},\\ & \{e\}, \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \{l_{\text{a}}\}, \text{event}),\end{aligned}$$

where $\text{inv}(v_0) = \text{true}$, $\text{inv}(v_1) = \text{true}$, $\text{source}(e) = v_0$,
$\text{flow}(v_0) = \text{true}$, $\text{flow}(v_1) = \text{false}$, $\text{target}(e) = v_1$,
$\text{done}(v_0) = \text{false}$, $\text{done}(v_1) = \text{true}$, $\text{urgent}(e) = \text{true}$,
$\text{guard}(e) = b$,
$\text{jump}(e) = (X_{\text{aux}} \cup W \cup J, \zeta_{W \cup J}(r))$,
$\text{event}(e) = l_{\text{a}}$.

A guarded action predicate can only delay while its guard is false. Only for the end-point of the trajectory the guard may become true. Therefore, the flow condition of location v_0 is true, and the urgent edge e is guarded with b.

The difference in notation of a jump predicate in the χ_{sub} language and the syntax of hybrid automata defined in Section 5.3.2.1 requires function $\zeta_{W\cup J}$. Function $\zeta_{W\cup J}$ renames variables of $W \cup J$ in r to variables with superscript "′", and replaces variables occurring with a "−" superscript in r to variables without any superscript. E.g. $\zeta_{\{x\}}(x+y = x^- + y^- + 5)$ becomes $x' + y = x + y + 5$. The set of variables which is allowed to change is given by $X_{\text{aux}} \cup W \cup J$. Edge e is labelled with the action label l_{a}.

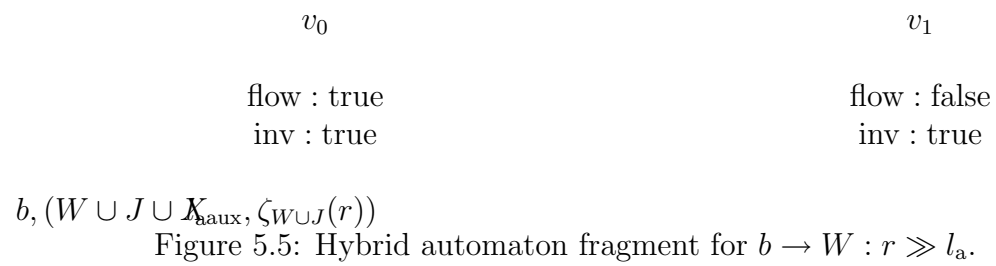

Figure 5.5: Hybrid automaton fragment for $b \to W : r \gg l_{\text{a}}$.

Guarded send and guarded receive Guarded send and guarded receive process terms $b \to h\,!!\mathbf{e}_n$ and $b \to h\,??\,\mathbf{x}_n$ behave as $h\,!!\,\mathbf{e}_n$ and $h\,??\,\mathbf{x}_n$ if $b = \text{true}$. If $b = \text{false}$, they can perform arbitrary time transitions. Process terms $h\,!!\,\mathbf{e}_n$ and $h\,??\,\mathbf{x}_n$ denote undelayable sending of expression $\mathbf{e}_n$ via channel h, and undelayable receiving of information via channel h into variable(s) $\mathbf{x}_n$, respectively. Let h denote an arbitrary channel, $\mathbf{h'}_n$ denote the variables $h'_1, \ldots, h'_n$, and $\mathbf{e}_n$ denote the expressions $e_1, \ldots, e_n$.

$$\begin{aligned}\mathcal{T}_J(b \to h\,!!\,\mathbf{e}_n) \;=\; & (\{v_0, v_1\}, v_0, \text{inv}, \text{flow}, \text{done},\\ & \{e\}, \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \{\text{isa}(h)\}, \text{event}),\end{aligned}$$

where
$$\begin{array}{lll} \text{inv}(v_0) = \text{true}, & \text{inv}(v_1) = \text{true}, & \text{source}(e) = v_0,\\ \text{flow}(v_0) = \text{true}, & \text{flow}(v_1) = \text{false}, & \text{target}(e) = v_1,\\ \text{done}(v_0) = \text{false}, & \text{done}(v_1) = \text{true}, & \text{urgent}(e) = \text{true},\\ & & \text{guard}(e) = b,\\ & & \text{jump}(e) = \left(J \cup X_{\text{aux}}, \bigwedge_{i=1}^{n} e_i = h'_i\right),\\ & & \text{event}(e) = \text{isa}(h).\end{array}$$

The above translation results in a timed transition system that differs from the χ-semantics in the sense that the label of the transition is different. In χ, besides the channel name, also the value of the expressions is an argument of the isa-action. The only way to achieve the same for the hybrid automaton, is to introduce a potentially infinite number of edges, one for each possible value for $\mathbf{e}_n$. This is not allowed in hybrid automata. Therefore, we have introduced auxiliary variables $\mathbf{h'}_n$ (with n the arity of the channel h). The sole purpose of these variables is to store the values of the expressions $\mathbf{e}_n$.

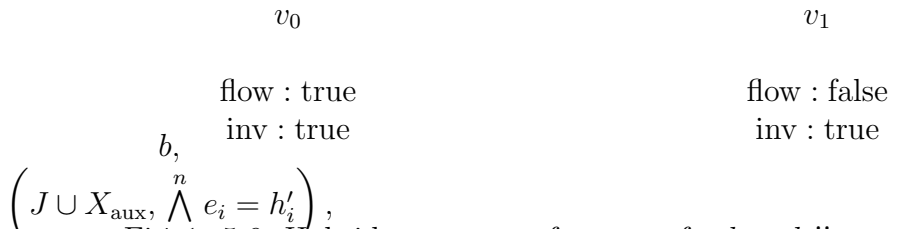

Figure 5.6: Hybrid automaton fragment for $b \to h \,!!\, \mathbf{e}_n$.

Let $\mathbf{x}_n$ denote the variables $x_1, \ldots, x_n$. Then

$$\begin{aligned}\mathcal{T}_J(b \to h \,??\, \mathbf{x}_n) \;=\; & (\{v_0, v_1\}, v_0, \text{inv}, \text{flow}, \text{done}, \{e\},\\ & \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \{\text{ira}(h, \{\mathbf{x}_n\})\}, \text{event}),\end{aligned}$$

where
$\text{init} = (\emptyset, \text{true})$,
$\text{inv}(v_0) = \text{true}$, $\text{inv}(v_1) = \text{true}$,
$\text{flow}(v_0) = \text{true}$, $\text{flow}(v_1) = \text{false}$,
$\text{done}(v_0) = \text{false}$, $\text{done}(v_1) = \text{true}$,
$\text{source}(e) = v_0$,
$\text{target}(e) = v_1$,
$\text{urgent}(e) = \text{true}$,
$\text{guard}(e) = b$,
$\text{jump}(e) = (\{\mathbf{x}_n\} \cup J \cup X_{\text{aux}}, \bigwedge_{i=1}^{n} h'_i = x'_i)$,
$\text{event}(e) = \text{ira}(h, \{\mathbf{x}_n\})$.

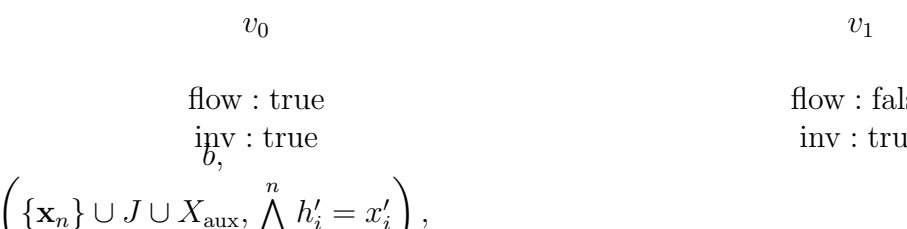

Figure 5.7: Hybrid automaton fragment for $b \to h \,??\, \mathbf{x}_n$.

5.3.3.2 Translation of operators of χ_{sub}

In this section, the translation of the χ_{sub} operators to hybrid automaton fragments is defined. We let p, q be closed process terms, and we use $\mathcal{T}_J(i) = (X_i, V_i, v_{0_i}, \text{inv}_i, \text{flow}_i, \text{done}_i, E_i, \text{source}_i, \text{target}_i, \text{urgent}_i, \text{guard}_i, \text{jump}_i, \Sigma_i, \text{event}_i)$ to denote the hybrid automaton fragment of i, for $i \in \{p, q\}$.

In the translations, notation f_{pq} is used as an abbreviation for $f_p \cup f_q$, where $f \in \{\text{inv}, \text{flow}, \text{done}, E, \text{source}, \text{target}, \text{urgent}, \text{guard}, \text{jump}, \Sigma, \text{event}\}$ and operator $\cup$ is defined as follows: If f and g are functions with $\text{dom}(f) \cap \text{dom}(g) = \emptyset$, then $f \cup g$ denotes the unique

function h with $\mathrm{dom}(h) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ satisfying the condition: for each $c \in \mathrm{dom}(h)$, if $c \in \mathrm{dom}(f)$ then $h(c) = f(c)$, and $h(c) = g(c)$ otherwise.

Any delay operator By means of the any delay operator $[p]$, time transitions of arbitrary duration are allowed for the behavior of p. Time transitions of p itself are neglected. The any delay operator does not affect the action behavior of p. Let $L_*(\mathcal{T}_J(p)) = (V_p, v_{p0}, \mathrm{inv}_p, \mathrm{flow}_p, \mathrm{done}_p, E_p, \mathrm{source}_p, \mathrm{target}_p, \mathrm{urgent}_p, \mathrm{guard}_p, \mathrm{jump}_p, \Sigma_p, \mathrm{event}_p)$ and $v_{0_p} = v_{p0}$, then

$$\begin{aligned}\mathcal{T}_J([p]) \;=\; & (\{v'_{0_p}\} \cup V_p, v'_{0_p}, \mathrm{inv} \cup \mathrm{inv}_p, \mathrm{flow} \cup \mathrm{flow}_p, \mathrm{done} \cup \mathrm{done}_p,\\ & E \cup E_p, \mathrm{source} \cup \mathrm{source}_p, \mathrm{target} \cup \mathrm{target}_p, \mathrm{urgent} \cup \mathrm{urgent}_p,\\ & \mathrm{guard} \cup \mathrm{guard}_p, \mathrm{jump} \cup \mathrm{jump}_p, \Sigma_p, \mathrm{event} \cup \mathrm{event}_p),\end{aligned}$$

where $\mathrm{dom}(\mathrm{inv}) = \mathrm{dom}(\mathrm{flow}) = \mathrm{dom}(\mathrm{done}) = \{v'_{0_p}\}$,
$\mathrm{inv}(v'_{0_p}) = \mathrm{true}, \mathrm{flow}(v'_{0_p}) = \mathrm{true}, \mathrm{done}(v'_{0_p}) = \mathrm{done}_p(v_{0_p})$,
$E = \{e' \mid e \in E_p, \mathrm{source}_p(e) = v_{0_p}\},\ E \cap E_p = \emptyset$,
$\mathrm{dom}(\mathrm{source}) = \mathrm{dom}(\mathrm{target}) = \mathrm{dom}(\mathrm{urgent}) = \mathrm{dom}(\mathrm{guard}) =$
$\mathrm{dom}(\mathrm{jump}) = \mathrm{dom}(\mathrm{event}) = E$,

$$\begin{aligned}\forall_{e' \in E}:\ & \mathrm{source}(e') = v'_{0_p},\\ & \mathrm{target}(e') = \mathrm{target}_p(e),\\ & \mathrm{urgent}(e') = \mathrm{false},\\ & \mathrm{guard}(e') = \mathrm{guard}_p(e),\\ & \mathrm{jump}(e') = \mathrm{jump}_p(e),\\ & \mathrm{event}(e') = \mathrm{event}_p(e).\end{aligned}$$

The automaton fragment $\mathcal{T}_J([p])$ may start with an arbitrary delay. We introduce an additional location v'_{0_p} to $\mathcal{T}_J(p)$ to play the role of the original initial location but now with arbitrary initial delay. Therefore, the invariant and the flow condition of v'_{0_p} are set to true. Also v'_{0_p} becomes the initial location of $\mathcal{T}_J([p])$. Observe that any transition from v'_{0_p} ends up in a location of $\mathcal{T}_J(p)$. Also, any urgent edge from v'_{0_p} is turned into a non-urgent one.

Sequential composition operator The sequential composition of process terms p and q behaves as process term p until p terminates, and then continues to behave as process term q. Let $L_*(\mathcal{T}_J(p)) = (V_p, v_{p0}, \mathrm{inv}_p, \mathrm{flow}_p, \mathrm{done}_p, E_p, \mathrm{source}_p, \mathrm{target}_p, \mathrm{urgent}_p, \mathrm{guard}_p, \mathrm{jump}_p, \Sigma_p, \mathrm{event}_p)$, $R_*(\mathcal{T}_J(q)) = (V_q, v_{q0}, \mathrm{inv}_q, \mathrm{flow}_q, \mathrm{done}_q, E_q, \mathrm{source}_q, \mathrm{target}_q, \mathrm{urgent}_q, \mathrm{guard}_q, \mathrm{jump}_q, \Sigma_q, \mathrm{event}_q)$, $v_{0_p} = v_{p0}$, and $v_{0_q} = v_{q0}$, then

$$\begin{aligned}\mathcal{T}_J(p;\, q) \;=\; & (V, v_{0_p}, \mathrm{inv}_{pq} \restriction V, \mathrm{flow}_{pq} \restriction V, \mathrm{done}_{pq} \restriction V, E_{pq}, E_{pq}, \mathrm{source}_{pq},\\ & \mathrm{target} \cup \mathrm{target}_q, \mathrm{urgent}_{pq}, \mathrm{guard}_{pq}, \mathrm{jump}_{pq}, \Sigma_{pq}, \mathrm{event}_{pq}),\end{aligned}$$

where $V = \{v \mid v \in V_p, \neg\mathrm{done}_p(v)\} \cup V_q$, $\mathrm{dom}(\mathrm{target}) = E_p$

$$\forall_{e \in E_p}:\ \mathrm{target}(e) = \begin{cases} v_{0_q} & \text{if } \mathrm{done}_p(\mathrm{target}_p(e)),\\ \mathrm{target}_p(e) & \text{otherwise.}\end{cases}$$

The initial location of $\mathcal{T}_J(p;\ q)$ is the initial location of $\mathcal{T}_J(p)$. The end-points of the edges that go to terminating locations of $\mathcal{T}_J(p)$ are reconnected to the initial location of $\mathcal{T}_J(q)$ (i.e. v_{0_q}). The terminating locations of $\mathcal{T}_J(p)$ are removed. This is safe since we never create hybrid automaton fragments with outgoing edges in terminating locations. The behavior of $\mathcal{T}_J(p;\ q)$ is straightforward: first p is executed, then q. Upon termination of p, the invariants of the initial location of q must hold.

From the above definition it follows that if the hybrid automaton fragment for p has no terminating locations, i.e. $\forall_{v \in V_p} : \neg \mathrm{done}_p(v)$, then the locations from $\mathcal{T}_J(q)$ are unreachable. Hence, $\mathcal{T}_J(p;\ q)$ and $\mathcal{T}_J(p)$ are identical apart from the unreachable locations/edges.

Alternative composition operator The delay behavior of $\mathcal{T}_J(p \,[\!]\, q)$ is the intersection of the respective delay behaviors of p and q. The action behavior is a non-deterministic choice between the first action allowed by p and the first action allowed by q. Let $L_*(\mathcal{T}_J(p)) = (V_p,\ v_{p0},\ \mathrm{inv}_p,\ \mathrm{flow}_p,\ \mathrm{done}_p,\ E_p,\ \mathrm{source}_p,\ \mathrm{target}_p,\ \mathrm{urgent}_p,\ \mathrm{guard}_p,\ \mathrm{jump}_p,\ \Sigma_p,\ \mathrm{event}_p)$, $R_*(\mathcal{T}_J(q)) = (V_q,\ v_{q0},\ \mathrm{inv}_q,\ \mathrm{flow}_q,\ \mathrm{done}_q,\ E_q,\ \mathrm{source}_q,\ \mathrm{target}_q,\ \mathrm{urgent}_q,\ \mathrm{guard}_q,\ \mathrm{jump}_q,\ \Sigma_q,\ \mathrm{event}_q)$, $v_{0_p} = v_{p0}$, and $v_{0_q} = v_{q0}$, then

$$\begin{aligned}\mathcal{T}_J(p \,[\!]\, q) \quad = \quad & (\{v_0\} \cup V_{pq}, v_0, \mathrm{inv} \cup \mathrm{inv}_{pq}, \mathrm{flow} \cup \mathrm{flow}_{pq}, \mathrm{done} \cup \mathrm{done}_{pq},\\ & E \cup E_{pq}, \mathrm{source} \cup \mathrm{source}_{pq}, \mathrm{target} \cup \mathrm{target}_{pq},\\ & \mathrm{urgent} \cup \mathrm{urgent}_{pq}, \mathrm{guard} \cup \mathrm{guard}_{pq}, \mathrm{jump} \cup \mathrm{jump}_{pq}, \Sigma_{pq},\\ & \mathrm{event} \cup \mathrm{event}_{pq}),\end{aligned}$$

where $\mathrm{dom}(\mathrm{inv}) = \mathrm{dom}(\mathrm{flow}) = \mathrm{dom}(\mathrm{done}) = \{v_0\}$,
$\mathrm{inv}(v_0) = \mathrm{inv}_p(v_{0_p}) \wedge \mathrm{inv}_q(v_{0_q}), \mathrm{flow}(v_0) = \mathrm{flow}_p(v_{0_p}) \wedge \mathrm{flow}_q(v_{0_q})$,
$\mathrm{done}(v_0) = \mathrm{done}_p(v_{0_p}) \wedge \mathrm{done}_q(v_{0_q})$,
$E = \{e' \mid e \in E_{pq}, \mathrm{source}_{pq}(e) \in \{v_{0_p}, v_{0_q}\}\},\ E \cap E_{pq} = \emptyset$,
$\mathrm{dom}(\mathrm{source}) = \mathrm{dom}(\mathrm{target}) = \mathrm{dom}(\mathrm{urgent}) = \mathrm{dom}(\mathrm{guard}) =$
$\mathrm{dom}(\mathrm{jump}) = \mathrm{dom}(\mathrm{event}) = E$,
$\forall_{e' \in E} :\ \mathrm{source}(e') = v_0, \mathrm{target}(e') = \mathrm{target}_{pq}(e)$,
$\mathrm{urgent}(e') = \mathrm{urgent}_{pq}(e), \mathrm{guard}(e') = \mathrm{guard}_{pq}(e)$,
$\mathrm{jump}(e') = \mathrm{jump}_{pq}(e), \mathrm{event}(e') = \mathrm{event}_{pq}(e)$.

The invariant, flow and done conditions of the initial location v_0 of $\mathcal{T}_J(p \,[\!]\, q)$ are the conjunction of the invariants, flow conditions and done conditions of v_{0_p} and v_{0_q}, respectively. The recursion condition of v_0 is the union of the recursion conditions of v_{0_p} and v_{0_q}. All outgoing edges from the original initial nodes, i.e., v_{0_p} an v_{0_q}, are copied to the new initial node with their original target.

It can be the case that the original initial nodes are not reachable anymore. In that case, they can of course be removed from the hybrid automaton fragment.

Parallel composition operator The parallel composition of process terms p and q has as its behavior with respect to action transitions the interleaving of the behaviors of p and q. The parallel composition allows the synchronization of matching send and receive actions. A send action $\mathrm{isa}(h, cs)$ and a receive action $\mathrm{ira}(h', cs', W)$ match iff $h = h'$ and $cs = cs'$;

i.e. the channels used for sending and receiving are the same, and also the values sent and the values received are identical. The time transitions of the process terms that are put in parallel have to synchronize to obtain the time transition (with the same time step t and trajectory ρ) of their parallel composition

Let $\gamma : \Sigma_p \times \Sigma_q \to \{\mathrm{ca}(h) \mid h \in H\}$ be defined as follows: for any channel name h and any $\mathbf{x}_n$

$$\begin{array}{rcll} \gamma(\mathrm{isa}(h), \mathrm{ira}(h, \{\mathbf{x}_n\})) & = & \mathrm{ca}(h) & \text{if } \mathrm{isa}(h) \in \Sigma_p \text{ and } \mathrm{ira}(h, \{\mathbf{x}_n\}) \in \Sigma_q, \\ \gamma(\mathrm{ira}(h, \{\mathbf{x}_n\}), \mathrm{isa}(h)) & = & \mathrm{ca}(h) & \text{if } \mathrm{ira}(h, \{\mathbf{x}_n\}) \in \Sigma_p \text{ and } \mathrm{isa}(h) \in \Sigma_q, \end{array}$$

and undefined otherwise.

Also, let $L_*(\mathcal{T}_J(p)) = (V_p, v_{p0}, \mathrm{inv}_p, \mathrm{flow}_p, \mathrm{done}_p, E_p, \mathrm{source}_p, \mathrm{target}_p, \mathrm{urgent}_p, \mathrm{guard}_p, \mathrm{jump}_p, \Sigma_p, \mathrm{event}_p)$, $R_*(\mathcal{T}_J(q)) = (V_q, v_{q0}, \mathrm{inv}_q, \mathrm{flow}_q, \mathrm{done}_q, E_q, \mathrm{source}_q, \mathrm{target}_q, \mathrm{urgent}_q, \mathrm{guard}_q, \mathrm{jump}_q, \Sigma_q, \mathrm{event}_q)$, $v_{0_p} = v_{p0}$, and $v_{0_q} = v_{q0}$, then

$$\begin{array}{rcl} \mathcal{T}_J(p \parallel q) & = & (V, (v_{0_p}, v_{0_q}), \mathrm{inv} \cup \mathrm{inv}_{pq}, \mathrm{flow} \cup \mathrm{flow}_{pq}, \mathrm{done} \cup \mathrm{done}_{pq}, \\ & & E, \mathrm{source} \cup \mathrm{source}_{pq}, \mathrm{target} \cup \mathrm{target}_{pq}, \mathrm{urgent} \cup \mathrm{urgent}_{pq}, \\ & & \mathrm{guard} \cup \mathrm{guard}_{pq}, \mathrm{jump} \cup \mathrm{jump}_{pq}, \Sigma, \mathrm{event} \cup \mathrm{event}_{pq}), \end{array}$$

where
$$\begin{array}{rcl} V_p^{\mathrm{done}} & = & \{v_p \in V_p \mid \mathrm{done}(v_p)\} \\ V_q^{\mathrm{done}} & = & \{v_q \in V_q \mid \mathrm{done}(v_q)\} \\ V' & = & (V_p \setminus V_p^{\mathrm{done}}) \times (V_q \setminus V_q^{\mathrm{done}}) \cup V_p \cup V_q \end{array}$$
$v_{\mathrm{done}} \notin V', V = V' \cup \{v_{\mathrm{done}}\}$,
$\mathrm{dom}(\mathrm{inv}) = \mathrm{dom}(\mathrm{flow}) = \mathrm{dom}(\mathrm{done}) = V \setminus V_{pq}$,
$\mathrm{inv}(v_{\mathrm{done}}) = \mathrm{true}, \mathrm{flow}(v_{\mathrm{done}}) = \mathrm{false}, \mathrm{done}(v_{\mathrm{done}}) = \mathrm{true}$,

$$\begin{array}{ll} \forall_{(v_p, v_q) \in (V_p \setminus V_p^{\mathrm{done}}) \times (V_q \setminus V_q^{\mathrm{done}})} & \mathrm{inv}(v_p, v_q) = \mathrm{inv}_p(v_p) \wedge \mathrm{inv}_q(v_q), \\ & \mathrm{flow}(v_p, v_q) = \mathrm{flow}_p(v_p) \wedge \mathrm{flow}_q(v_q), \\ & \mathrm{done}(v_p, v_q) = \mathrm{done}_p(v_p) \wedge \mathrm{done}_q(v_q), \end{array}$$

$$\begin{array}{rcl} E & = & (E_p \times (V_q \setminus V_q^{\mathrm{done}})) \cup ((V_p \setminus V_p^{\mathrm{done}}) \times E_q) \cup E_p \cup E_q \\ & \cup & \{(e_p, e_q) \in E_p \times E_q \mid \gamma(\mathrm{event}(e_p), \mathrm{event}(e_q)) \text{ defined}\}, \end{array}$$
$\mathrm{dom}(\mathrm{source}) = \mathrm{dom}(\mathrm{target}) = \mathrm{dom}(\mathrm{urgent}) = \mathrm{dom}(\mathrm{guard}) =$
$\mathrm{dom}(\mathrm{jump}) = \mathrm{dom}(\mathrm{event}) = E \setminus E_{pq}$,
$\Sigma = \Sigma_p \cup \Sigma_q \cup \mathrm{range}(\gamma)$,

$$\begin{array}{ll} \forall_{(e_p, v_q) \in E} : & \mathrm{source}(e_p, v_q) = (\mathrm{source}_p(e_p), v_q), \\ & \mathrm{target}(e_p, v_q) = \begin{cases} (\mathrm{target}_p(e_p), v_q) & \text{if } \neg \mathrm{done}_p(\mathrm{target}_p(e_p)), \\ v_q & \text{if } \mathrm{done}_p(\mathrm{target}_p(e_p)), \end{cases} \\ & \mathrm{urgent}(e_p, v_q) = \mathrm{urgent}_p(e_p), \mathrm{guard}(e_p, v_q) = \mathrm{guard}_p(e_p), \\ & \mathrm{jump}(e_p, v_q) = \mathrm{jump}_p(e_p), \mathrm{event}(e_p, v_q) = \mathrm{event}_p(e_p), \end{array}$$

$$
\begin{aligned}
\forall_{(v_p,e_q)\in E}: \quad & \mathrm{source}(v_p,e_q) = (v_p, \mathrm{source}_q(e_q)),\\
& \mathrm{target}(v_p,e_q) = \begin{cases} (v_p, \mathrm{target}_q(e_q)) & \text{if } \neg\mathrm{done}_q(\mathrm{target}_q(e_q)),\\ v_p & \text{if } \mathrm{done}_q(\mathrm{target}_q(e_q)),\end{cases}\\
& \mathrm{urgent}(v_p,e_q) = \mathrm{urgent}_q(e_q), \mathrm{guard}(v_p,e_q) = \mathrm{guard}_q(e_q),\\
& \mathrm{jump}(v_p,e_q) = \mathrm{jump}_q(e_q), \mathrm{event}(v_p,e_q) = \mathrm{event}_q(e_q),
\end{aligned}
$$

$$
\begin{aligned}
&\forall_{(e_p,e_q)\in E}:\\
&\quad \mathrm{source}(e_p,e_q) = (\mathrm{source}_p(e_p), \mathrm{source}_q(e_q)),\\
&\quad \mathrm{target}(e_p,e_q) = \begin{cases}
(\mathrm{target}_p(e_p), \mathrm{target}_q(e_q)) & \text{if } \neg\mathrm{done}_p(\mathrm{target}_p(e_p)) \text{ and}\\
& \neg\mathrm{done}_q(\mathrm{target}_q(e_q)),\\
\mathrm{target}_q(e_q) & \text{if } \mathrm{done}_p(\mathrm{target}_p(e_p)) \text{ and}\\
& \neg\mathrm{done}_q(\mathrm{target}_q(e_q)),\\
\mathrm{target}_p(e_p) & \text{if } \neg\mathrm{done}_p(\mathrm{target}_p(e_p)) \text{ and}\\
& \mathrm{done}_q(\mathrm{target}_q(e_q)),\\
v_{\mathrm{done}} & \text{if } \mathrm{done}_p(\mathrm{target}_p(e_p)) \text{ and}\\
& \mathrm{done}_q(\mathrm{target}_q(e_q)),
\end{cases}\\
&\quad \mathrm{urgent}(e_p,e_q) = \mathrm{urgent}_p(e_p) \vee \mathrm{urgent}_q(e_q),\\
&\quad \mathrm{guard}(e_p,e_q) = \mathrm{guard}_p(e_p) \wedge \mathrm{guard}_q(e_q),\\
&\quad \mathrm{jump}(e_p,e_q) = \mathrm{jump}_p(e_p) \wedge \mathrm{jump}_q(e_q),\\
&\quad \mathrm{event}(e_p,e_q) = \gamma(\mathrm{event}_p(e_p), \mathrm{event}(e_q)).
\end{aligned}
$$

Let $\mathrm{jump}_p(e_p) = (W_p, r_p)$ and $\mathrm{jump}_q(e_q) = (W_q, r_q)$, then notation $\mathrm{jump}_p(e_p) \wedge \mathrm{jump}_q(e_q)$ is defined as $(W_p \cup W_q, r_p \wedge r_q)$.

In this translation, an additional terminating location v_{done} is introduced. Besides the terminating locations, locations of two hybrid automaton fragments ($\mathcal{T}_J(p)$ and $\mathcal{T}_J(q)$) are conjoined. The conjunction of the invariants, and the conjunction of the flow and done conditions apply. Action transitions from the components are interleaved, apart from the synchronization of matching send and receive actions, in which they are executed simultaneously. If the synchronization of matching send and receive actions can terminate successfully, the control of the hybrid automaton fragment ($\mathcal{T}_J(p \parallel q)$) ends up in the terminating location v_{done}.

Repetition operator Process term $*p$ represents the infinite repetition of process term p. Let $L_*(\mathcal{T}_J(p)) = (V_p,\ v_{p_0},\ \mathrm{inv}_p,\ \mathrm{flow}_p,\ \mathrm{done}_p,\ E_p,\ \mathrm{source}_p,\ \mathrm{target}_p,\ \mathrm{urgent}_p,\ \mathrm{guard}_p,\ \mathrm{jump}_p,\ \Sigma_p,\ \mathrm{event}_p)$ and $v_{0_p} = v_{p_0}$, then

$$
\begin{aligned}
\mathcal{T}_J(*p) \quad = \quad & (V, v_{0_p}, \mathrm{inv}_p \restriction V, \mathrm{flow}_p \restriction V, \mathrm{done}_p \restriction V,\\
& E_p, \mathrm{source}_p, \mathrm{target}, \mathrm{urgent}_p, \mathrm{guard}_p, \mathrm{jump}_p, \Sigma_p, \mathrm{event}_p),
\end{aligned}
$$

where $V = \{v \mid v \in V_p, \neg\mathrm{done}_p(v)\}, \mathrm{dom}(\mathrm{target}) = E_p$

$$
\forall_{e\in E_p}:\ \mathrm{target}(e) = \begin{cases} v_{0_p} & \text{if } \mathrm{done}_p(\mathrm{target}_p(e)),\\ \mathrm{target}_p(e) & \text{otherwise.}\end{cases}
$$

The end-points of the edges that go to terminating locations are reconnected to the initial location v_{0_p}. The terminating locations are removed. As mentioned previously, this is safe, because we never create hybrid automaton fragments with outgoing edges in terminating locations.

Jump enabling operator The jump enabling operator applied to a process term p with set J^+ ($\iota_{J^+}(p)$) behaves the same as its argument in an environment where the variables from set J^+ become jumping variables.

Action encapsulation operator The behavior of the action encapsulation applied to a process term $\partial_{\mathcal{A}}(p)$ is the same as the behavior of its argument with the restriction that actions from the set A ($A \subseteq \mathcal{A} \setminus \{\tau\}$) cannot be executed. Let $L_*(\mathcal{T}_J(p)) = (V_p,\ v_{p0},\ \mathrm{inv}_p,\ \mathrm{flow}_p,\ \mathrm{done}_p,\ E_p,\ \mathrm{source}_p,\ \mathrm{target}_p,\ \mathrm{urgent}_p,\ \mathrm{guard}_p,\ \mathrm{jump}_p,\ \Sigma_p,\ \mathrm{event}_p)$ and $v_{0_p} = v_{p0}$, then

$$\begin{aligned}\mathcal{T}_J(\partial_{\mathcal{A}}(p)) \quad = \quad & (V_p, v_{0_p}, \mathrm{inv}_p, \mathrm{flow}_p, \mathrm{done}_p,\\ & E_p, \mathrm{source}_p, \mathrm{target}_p, \mathrm{urgent}_p, \mathrm{guard}_p, \mathrm{jump}_p, \Sigma_p, \mathrm{event}_p),\end{aligned}$$

where $\forall_{e \in E_p}:\ \mathrm{jump}(e) = \begin{cases}(\emptyset, \mathrm{false}) & \text{if } \mathrm{event}_p(e) \in A,\\ \mathrm{jump}_p(e) & \text{otherwise.}\end{cases}$

If the event label of an edge is in the set of A, then the jump condition of that edge is replaced by a predicate false with an empty set of variables that are allowed to change.

Urgent communication operator The urgent communication operator $\upsilon_H(p)$ gives communication actions via channels from set $H \subseteq \mathcal{H}$ a higher priority than time transitions. Action behavior and consistency are not affected by the urgent communication operator. Time transitions are allowed only if at each intermediate state while delaying no communication actions via channels from H are possible. Let $L_*(\mathcal{T}_J(p)) = (V_p,\ v_{p0},\ \mathrm{inv}_p,\ \mathrm{flow}_p,\ \mathrm{done}_p,\ E_p,\ \mathrm{source}_p,\ \mathrm{target}_p,\ \mathrm{urgent}_p,\ \mathrm{guard}_p,\ \mathrm{jump}_p,\ \Sigma_p,\ \mathrm{event}_p)$ and $v_{0_p} = v_{p0}$, then

$$\begin{aligned}\mathcal{T}_J(\upsilon_H(p)) \quad = \quad & (V_p, v_{0_p}, \mathrm{inv}_p, \mathrm{flow}_p, \mathrm{done}_p,\\ & E_p, \mathrm{source}_p, \mathrm{target}_p, \mathrm{urgent}, \mathrm{guard}_p, \mathrm{jump}_p, \Sigma_p, \mathrm{event}_p),\end{aligned}$$

where $\forall_{e \in E_p}:\ \mathrm{urgent}(e) = \begin{cases}\mathrm{true} & \text{if } \mathrm{event}_p(e) = \mathrm{ca}(h) \text{ for some } h \in H,\\ \mathrm{urgent}_p(e) & \text{otherwise.}\end{cases}$

If the event label of an edge is $\mathrm{ca}(h)$, where $h \in H$, then the edge becomes urgent.

5.3.4 Correctness of the translation

In this section, it is proved that any transition of a χ specification can be mimicked by a transition in the corresponding hybrid automaton model and vice versa. This indicates that the translation as defined in this chapter is correct.

In the operational semantics of χ (see also Chapter 3), a consistency predicate plays an important role. The following theorems state the relationship between consistency in χ and admissibility of states in hybrid automata.

Theorem 5.3.1 *Let p be a closed process term, v_0 be the initial location of $\mathcal{T}_J(p)$, α and σ be valuations such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$, ξ be an extended valuation such that $\sigma = \xi \restriction \mathrm{dom}(\sigma)$, and $E = (C, J, \emptyset, H, \emptyset)$ be an environment. Then*

$$(v_0, \alpha) \text{ is an admissible state of } [\![\mathcal{HA}(\langle p, \sigma, E \rangle)]\!] \quad \Leftrightarrow \quad \langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$$

for some extended valuation ξ.

PROOF. See Appendix C.2.

Theorem 5.3.2 *Let p be a closed process term, σ be a valuation, ξ, ξ' be extended valuations, $E = (C, J, \emptyset, H, \emptyset)$ be an environment, a be an action label, $t \in T$, and ρ be a trajectory. Then*

$$\langle p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \quad \Rightarrow \quad (v_0, \alpha) \text{ is an admissible state of } [\![\mathcal{HA}(\langle p, \sigma, E \rangle)]\!]$$

$$\langle p, \sigma, E \rangle \overset{t, \rho}{\longmapsto} \quad \Rightarrow \quad (v_0, \alpha) \text{ is an admissible state of } [\![\mathcal{HA}(\langle p, \sigma, E \rangle)]\!]$$

where v_0 is the initial location of $\mathcal{T}_J(p)$ and α is any valuation such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$.

PROOF. (Sketch) These follow from Lemma 3.5.2 and Lemma 3.5.3 (using Lemma 3.5.1) and Theorem 5.3.1.

The following theorem states that the solution concepts of χ and the definition of hybrid automata that is used in this chapter are closely related.

Theorem 5.3.3 *$\rho \in \Omega_{FG}(\sigma, C, \emptyset, u, t)$ if and only if ρ' is a solution of the delay predicate u in the hybrid automaton model, where $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$.*

PROOF. (Sketch) This follows from the function Ω and the semantics of a hybrid automaton.

The following theorems state that for any transition in the hybrid transition system associated with a χ process, there is a corresponding transition in the timed transition system of the hybrid automaton that is obtained by the translation defined in this chapter, and vice versa. The proofs of the Theorems 5.3.4 and 5.3.5 are given in Appendices C.3 and C.4.

According to the result of each χ deduction rule in Section 3.3 and Section 3.4 , the environment associating to a χ process is never changed in a transition. Hence, we consider this fact for the following theorems.

Theorem 5.3.4 *Let p and p' be closed process terms, $\sigma, \sigma', \alpha, \alpha'$ be valuations such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, ξ, ξ' be extended valuations, $E = (C, J, \emptyset, H, \emptyset)$ be an environment, and v_0 be the initial location of $\mathcal{T}_J(p)$. Then*

1. *for any non-communication action a*

$$\langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle \Rightarrow [\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{a} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha')$$

2. $$\langle p, \sigma, E\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle \Rightarrow \begin{array}{l} [\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{isa}(h)} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha') \\ \qquad \wedge\, \alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n \end{array}$$

3. $$\langle p, \sigma, E\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,Y), \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle \Rightarrow$$
$$[\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{ira}(h,Y)} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha')$$

4. $$\langle p, \sigma, E\rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle \Rightarrow \begin{array}{l} [\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{ca}(h)} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha') \\ \qquad \wedge\, \alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n \end{array}$$

5. $$\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle \Rightarrow [\![\, \mathcal{HA}(\langle p', \sigma, E\rangle)\,]\!] \models \rho' : (v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$$

where v_0' is a terminating location of $\mathcal{T}_J(p)$, v_0'' is the initial location of $\mathcal{T}_J(p')$ and $\rho' : [0, t] \to (X \mapsto \Lambda)$ is a trajectory such that $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$.

Theorem 5.3.5 *Let p be a closed process term, $\sigma, \sigma', \alpha, \alpha'$ be valuations such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, and $E = (C, J, \emptyset, H, \emptyset)$ be an environment. Let v_0, v_0' and v_0'' be the initial location, a terminating location and a non-terminating location of $\mathcal{T}_J(p)$, respectively. Then*

1. *for any non-communication action a*

$$[\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{a} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha') \;\Rightarrow\; \exists_{\xi, \xi'}\, \langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle$$

2. $$[\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{isa}(h)} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha') \Rightarrow$$
$$\exists_{\xi, \xi', cs}\, \langle p, \sigma, E\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle \wedge \alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$$

3. $$[\![\, \mathcal{HA}(\langle p, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{ira}(h,Y)} (\begin{smallmatrix} v_0'\\ v_0''\end{smallmatrix}, \alpha') \Rightarrow \exists_{\xi, \xi', cs} : \langle p, \sigma, E\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,Y), \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle$$

4. $[\![\mathcal{HA}(\langle p,\sigma,E\rangle)]\!] \models (v_0,\alpha) \xrightarrow{\mathrm{ca}(h)} (\begin{smallmatrix} v_0' \\ v_0'' \end{smallmatrix},\alpha') \Rightarrow$
$\exists_{\xi,\xi',cs} : \langle p,\sigma,E\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix},\sigma',E\rangle \wedge \alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$

5. $[\![\mathcal{HA}(\langle p,\sigma,E\rangle)]\!] \models \rho' : (v_0,\alpha) \overset{t}{\longmapsto} (v_0,\alpha') \Rightarrow \langle p,\sigma,E\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',E\rangle$

where p' is some closed process term such that v_0'' is the initial location of $\mathcal{T}_J(p')$ and $\rho' : [0,t] \to (X \mapsto \Lambda)$ is a trajectory such that $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$.

5.3.5 Example: Bottle filling system

The bottle filling system from Figure 5.8 consists of a liquid storage tank and a bottle filling line.

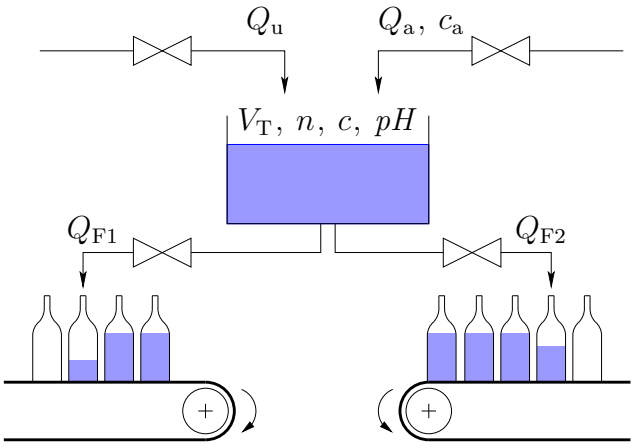

Figure 5.8: The bottle filling system.

The bottles are filled with liquid from the storage tank. A control system keeps the volume V in the storage tank between 1 and 10, and the pH level (acidity) of the liquid in the storage tank between 7 and 7.1. The liquid in the storage tank slowly becomes less acidic (pH level increases). To correct this, a strong acid is dribbled into the storage tank when the acidity of the liquid becomes too low ($pH \geq 7.1$).

The acid and liquid supply processes are not modeled, since we consider the acid always to be available, and we are not interested in the amount of acid that is used. The χ specification of the bottle filling system is as follows, where symbols Q_seta, Q_setu, , V_setb, t_tr, c_a, c_u, and K_loss denote constants:

$\langle$ disc $t_\mathrm{next}, Q_\mathrm{a}, Q_\mathrm{u}, Q_\mathrm{F}$, cont $n, V, V_\mathrm{b},\ pH, c$
, $t_\mathrm{next} = 0, Q_\mathrm{a} = 0, Q_\mathrm{u} = 0, Q_\mathrm{F} = 0, V = 0, V_\mathrm{b} = 2, pH = 7$

$$
\begin{array}{ll}
\langle & \dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F}} \\
, & \dot{n} = c_{\mathrm{u}} Q_{\mathrm{u}} + c_{\mathrm{a}} Q_{\mathrm{a}} - c Q_{\mathrm{F}} - K_{\mathrm{loss}} V \\
, & n = cV \\
, & pH = -\log c/1000 \\
, & \dot{V}_{\mathrm{b}} = Q_{\mathrm{F}} \\
\| & *(\ pH \geq 7.1 \rightarrow Q_{\mathrm{a}} := Q_{\mathrm{seta}};\ pH \leq 7 \rightarrow Q_{\mathrm{a}} := 0\ ) \\
\| & *(\ V \leq 1 \rightarrow Q_{\mathrm{u}} := Q_{\mathrm{setu}};\ V \geq 10 \rightarrow Q_{\mathrm{u}} := 0\ ) \\
\| & *(\ [V < 2 \rightarrow \mathit{start}\,!!\,1 \,[\!]\, V \geq 2 \rightarrow \mathit{start}\,!!\,2]\ ) \\
\| & *(\ \mathit{start}\,?\,Q_{\mathrm{setF}};\ V_{\mathrm{b}} \geq V_{\mathrm{setb}} \rightarrow Q_{\mathrm{F}}, t_{\mathrm{next}} := 0, \mathsf{time} + t_{\mathrm{tr}} \\
& \quad ;\ \mathsf{time} \geq t_{\mathrm{next}} \rightarrow V_{\mathrm{b}} := 0 \\
& \quad) \\
\rangle &
\end{array}
$$

The storage tank and the bottle filling line are connected by means of variable Q_{F}. Depending on the volume in the tank, the bottles are filled with a different rate.

The molar quantity and molar concentration of the acid in the storage tank are denoted by n and c, respectively, where $n = cV$. The incoming flows of liquid and acid to the liquid storage tank are denoted by Q_{u} and Q_{a}, respectively. The outgoing flow to the bottle filling process is denoted by Q_{F}.

It is assumed that the liquids are incompressible, and that the volumes of the fluids remain the same when they are mixed. In such a case, the volume V of the mixed liquid equals the sum of its components which leads to the following equation

$$\dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F}}.$$

Next, the mass balance (actually mol balance) for the dissolved substance is derived. Acid comes into the tank by means of the flows Q_{u} and Q_{a}. Acid leaves the tank in outgoing flow Q_{F}. Because the concentrations are in [mol/m^3], they can be directly multiplied with the flows (in [m^3/s]), which leads to

$$\dot{n} = c_{\mathrm{u}} Q_{\mathrm{u}} + c_{\mathrm{a}} Q_{\mathrm{a}} - c Q_{\mathrm{F}},$$

where c_{u} and c_{a} denote the concentrations of acid in the flows Q_{u} and Q_{a}.

The gradual reduction of the acidity of the liquid is modeled by means of a constant K_{loss}, which leads to

$$\dot{n} = c_{\mathrm{u}} Q_{\mathrm{u}} + c_{\mathrm{a}} Q_{\mathrm{a}} - c Q_{\mathrm{F}} - K_{\mathrm{loss}} V.$$

It is assumed that the acid is completely decomposed. Taking into account that the units of c are in [mol/m^3] instead of [mol/l], the pH is given by

$$pH = -\log c/1000.$$

The behavior of the pH controller model is explained as follows. Initially, the pH of the liquid in the storage tank equals 7. It is assumed that the pH level of the incoming liquid is 7 or more, since the acidity controller can only make the acidity of the storage

tank increase, causing the pH to decrease. If the pH value exceeds the maximum value ($pH \geq 7.1$), the acid valve is opened ($Q_{\mathrm{a}} := Q_{\mathrm{seta}}$) so that acid is dribbled into the tank. Dribbling of the acid continues until the pH value comes back at 7, where after the valve is closed ($Q_{\mathrm{a}} := 0$). In a similar way, the volume controller tries to keep the level of the storage tank between 5 and 10.

The behavior of the filling controller model is explained as follows. Depending on the volume in the tank, the bottles filling rate is adjusted ($[V < 2 \rightarrow \mathit{start} \,!!\, 1 \,[\!]\, V \geq 2 \rightarrow \mathit{start} \,!!\, 2]$). The filled rate is communicated to the bottle filling process via channel *start*.

The bottle filling process is started ($Q_{\mathrm{F}} := Q_{\mathrm{setF}}$) when the volume to be filled is received from the filling controller ($\mathit{start} \,?\, V_{\mathrm{setb}}$). When the bottle is full ($V \geq V_{\mathrm{setb}}$), the valve is closed, and the arrival time of the next bottle (t_{next}) is calculated $Q_{\mathrm{F}}, t_{\mathrm{next}} := 0, \mathsf{time} + t_{\mathrm{tr}}$. The time needed to place a new bottle under the filling nozzle is given by t_{tr}. At the arrival time of the next bottle, the bottle volume is reset to 0, which models the arrival of a new bottle, and the filling process is repeated.

The χ model of the bottle filling system cannot be translated directly. Using properties as defined in Section 3.5.4, guarded (multi-) assignments are rewritten to guarded action predicates ($b \rightarrow \mathbf{x}_n := \mathbf{e}_n \triangleq b \rightarrow \{\mathbf{x}_n\} : x_1 = e_1^- \wedge \cdots \wedge x_n = e_n^- \gg \tau$), unguarded (multi-) assignments are rewritten to guarded action predicates with guard true and delayable receive process terms are rewritten in terms of the guarded undelayable receive process term and the any delay operator ($h \,?\, \mathbf{x}_n \triangleq [\mathrm{true} \rightarrow h \,??\, \mathbf{x}_n]$). Discrete variables are rewritten to continuous variables, and for each discrete variable x, there is an additional equation of the form $\dot{x} = 0$. In principle, algebraic variables may jump arbitrarily during action transitions, while continuous variables may not jump unless specified in a action predicate. If there are no discontinuities in the trajectories of an algebraic variable, this variable can also be declared as a jumping continuous variable. Since there are no discontinuities in the trajectories of the algebraic variables pH and c, and these variables do not jump during the action transitions, they can be declared as (non-jumping) continuous variables. This results in the following χ specification:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ t_{\mathrm{next}}, Q_{\mathrm{a}}, Q_{\mathrm{u}}, Q_{\mathrm{F}}, n, V, V_{\mathrm{b}}, pH, c \\
,\ t_{\mathrm{next}} = 0, Q_{\mathrm{a}} = 0, Q_{\mathrm{u}} = 0, Q_{\mathrm{F}} = 0, V = 0, V_{\mathrm{b}} = 2, pH = 7 \\
|\quad \dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F}} \\
\wedge\quad \dot{n} = c_{\mathrm{u}} Q_{\mathrm{u}} + c_{\mathrm{a}} Q_{\mathrm{a}} - c Q_{\mathrm{F}} - K_{\mathrm{loss}} V \\
\wedge\quad n = cV \\
\wedge\ pH = -\log c/1000 \\
\wedge\ \dot{V}_{\mathrm{b}} = Q_{\mathrm{F}} \\
\wedge \dot{t}_{\mathrm{next}} = 0 \wedge \dot{Q}_{\mathrm{a}} = 0 \wedge \dot{Q}_{\mathrm{u}} = 0 \wedge \dot{Q}_{\mathrm{F}} = 0 \\
\|\ *(\ pH \geq 7.1 \rightarrow \{Q_{\mathrm{a}}\} : Q_{\mathrm{a}} = Q_{\mathrm{seta}};\ pH \leq 7 \rightarrow \{Q_{\mathrm{a}}\} : Q_{\mathrm{a}} = 0\ ) \\
\|\ *(\ V \leq 1 \rightarrow \{Q_{\mathrm{u}}\} : Q_{\mathrm{u}} = Q_{\mathrm{setu}};\ V \geq 10 \rightarrow \{Q_{\mathrm{u}}\} : Q_{\mathrm{u}} = 0\ ) \\
\|\ *(\ [V < 2 \rightarrow \mathit{start} \,!!\, 1 \,[\!]\, V \geq 2 \rightarrow \mathit{start} \,!!\, 2]\ )
\end{array}
$$

$$
\begin{array}{l}
\| *(\ [\text{true} \rightarrow \mathit{start} \,??\, Q_{\mathrm{F}}];\ V_{\mathrm{b}} \geq V_{\mathrm{setb}} \rightarrow \{Q_{\mathrm{F}}, t_{\mathrm{next}}\} : Q_{\mathrm{F}} = 0 \wedge t_{\mathrm{next}} = \mathsf{time} + tr \\
\quad ;\ \mathsf{time} \geq t_{\mathrm{next}} \rightarrow \{V_{\mathrm{b}}\} : V_{\mathrm{b}} = 0 \\
\quad) \\
\rangle
\end{array}
$$

Hybrid automaton of the bottle filling system The hybrid automaton of the complete bottle filling system consists of 16 reachable locations and 16 edges connecting them. For the purpose of illustration, only the hybrid automaton fragments of the subcomponents (pH, volume and filling controllers, the bottle filling process) are shown instead of their parallel composition. In the figures, a graphical notation similar to the notation for hybrid automata as defined in [HHWT95] is used. In this notation, the unreachable locations and edges are omitted, the initial location of the hybrid automaton fragment is represented as a circle with an incoming unlabelled arrow, terminating locations are represented as double circles, and the double edges represent urgent edges. In the examples below, the translations of process terms to hybrid automata fragments are given for the case $J = \emptyset$.

The hybrid automaton fragments of the pH controller and the volume controller are shown in Figures 5.9 and 5.10, respectively. Figures 5.11 and 5.12 show the hybrid

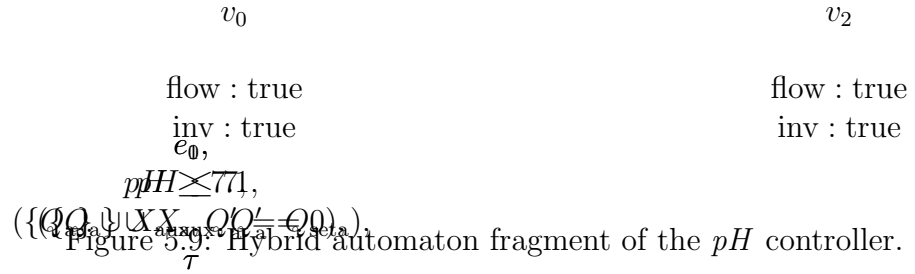

Figure 5.9: Hybrid automaton fragment of the pH controller.

automaton fragments of the filling controller and the bottle filling process. The hybrid automaton fragment of the parallel composition of the filling controller and the filling line is shown in Figure 5.13, where EQ denotes $\dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F}} \wedge \dot{n} = c_{\mathrm{u}}Q_{\mathrm{u}} + c_{\mathrm{a}}Q_{\mathrm{a}} - cQ_{\mathrm{F}} - K_{\mathrm{loss}}V \wedge n = cV \wedge pH = -\log c/1000 \wedge \dot{V}_{\mathrm{b}} = Q_{\mathrm{F}} \wedge \dot{t}_{\mathrm{next}} = 0 \wedge \dot{Q}_{\mathrm{a}} = 0 \wedge \dot{Q}_{\mathrm{u}} = 0 \wedge \dot{Q}_{\mathrm{F}} = 0$, EQ_{DC} denotes $dV = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F}} \wedge dn = c_{\mathrm{u}}Q_{\mathrm{u}} + c_{\mathrm{a}}Q_{\mathrm{a}} - cQ_{\mathrm{F}} - K_{\mathrm{loss}}V \wedge n = cV \wedge pH = -\log c/1000 \wedge dV_{\mathrm{b}} = Q_{\mathrm{F}} \wedge dt_{\mathrm{next}} = 0 \wedge dQ_{\mathrm{a}} = 0 \wedge dQ_{\mathrm{u}} = 0 \wedge dQ_{\mathrm{F}} = 0$, and X_{aux} denotes $\{dV,\ dn,\ pH,\ dV_{\mathrm{b}},\ dt_{\mathrm{next}},\ dQ_{\mathrm{a}},\ dQ_{\mathrm{u}},\ dQ_{\mathrm{F}},\ start_1\}$.

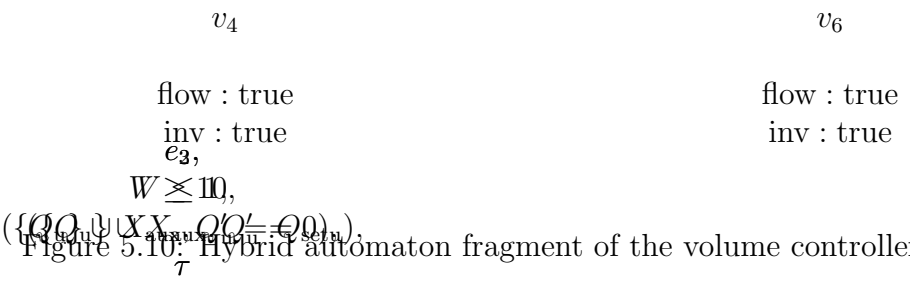

Figure 5.10: Hybrid automaton fragment of the volume controller.

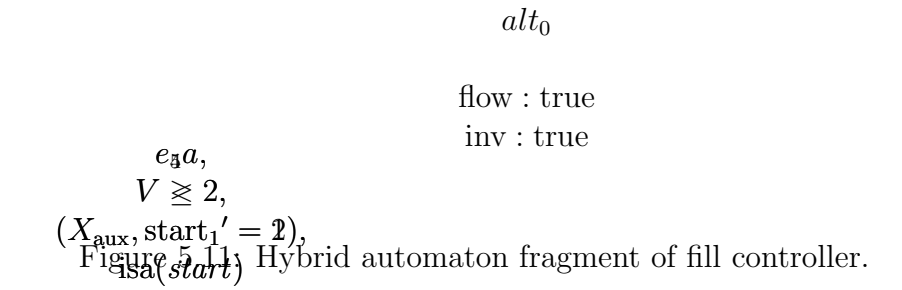

Figure 5.11: Hybrid automaton fragment of fill controller.

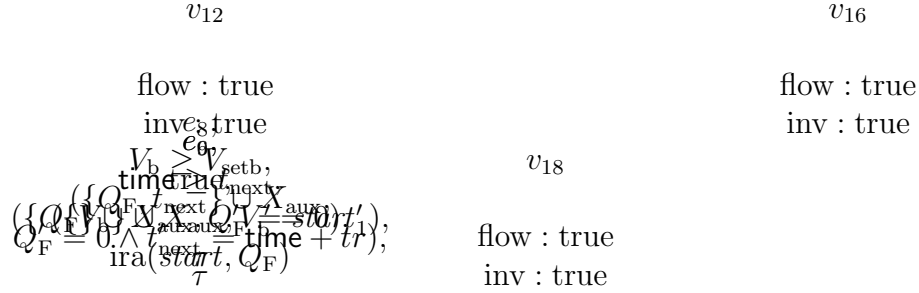

Figure 5.12: Hybrid automaton fragment of filling line.

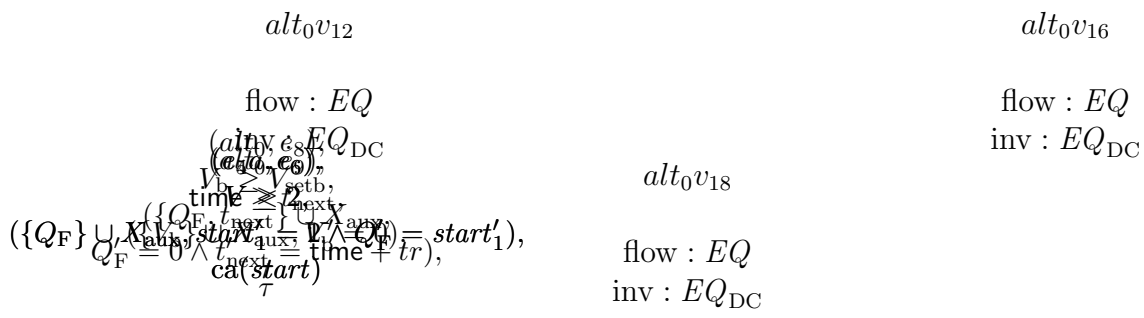

Figure 5.13: Hybrid automaton fragment of parallel composition of fill controller and filling line.

CHAPTER

SIX

Tool support

In this chapter, tools developed for the hybrid χ formalism are described. We do not describe the implementation of the χ toolset but confine ourselves to its theoretical foundations and its architecture. At a global level, the χ toolset consists of the following components:

- Stepper The stepper computes the semantics of a χ process as defined in Chapter 3. That is, given a process, the stepper computes the set of possible steps (see Section 6.1 for the definition of steps) and transitions.
- Simulator The simulator provides functionality to simulate χ specifications.
- Chi2HA translator The Chi2HA translator implements the translation function from Section 5.3.3.

The χ toolset is integrated with third-party tools for

- computation of solutions of action predicates and delay predicates,
- visualization of simulation results,
- visualization of the hybrid automata that are obtained using the Chi2HA translator,
- verification of properties of the hybrid automata obtained using the Chi2HA translator, and thus the translated χ specifications.

This chapter is organized as follows. In Section 6.1, the stepper is formally defined. The symbolic simulator is described in Section 6.2, and in Section 6.3 the Chi2HA translator is described.

6.1 Formal definition of stepper

The stepper computes the semantics of a χ process as defined in Chapter 3. That is, given a process, the stepper computes the set of possible steps. The stepper consists of three main functions: function $\mathcal{S}$ which returns a set of steps given a χ process, function Tr

which returns a set of transitions given a step, and function Tr' which returns a reduced set of transitions.

Steps are generalised transitions. Analogous to transitions, there are two types of steps:

- action steps,
- time steps.

An action step represents zero or more action transitions and a time step represents zero or more time transitions. An action step $(p, \sigma, E, c_p, W_p, r_p, l_{\mathrm{a}p}, C_p^b, C_p^a, p')$ consists of the χ process itself (p, σ, E), the condition (guards) c_p that should hold, the set of variables W_p that may change, the predicate r_p describing the discrete updates, the performed action label $l_{\mathrm{a}p}$, the consistency requirements before the action C_p^b, the consistency requirements after the action C_p^a, and the resulting process term p'. A time step $(p, \sigma, E, c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p')$ consist of the χ process itself (p, σ, E), the predicate $c_p^{[0]}$ that should hold at the startpoint of the time transitions, the predicate $c_p^{(0,t)}$ that should hold at all time points between the start- and endpoint of the time transitions, the predicate $c_p^{[t]}$ that should hold at the endpoint of the time transitions, the predicate $c_p^{[0,t]}$ that should hold at all timepoints (including the start- and endpoint) of the time transitions, the predicate c_p that should hold at least at one timepoint of the time transition, and the resulting process term p'.

Function Tr returns the set of transitions given a χ process. In general, the set of transitions of a χ specification is infinite. For instance, action predicate $\{x\} : \mathrm{true} \gg \tau$ has an infinite number of solutions for variable x (assuming that the type of x contains an infinite number of elements). The same holds for delay predicates that can have an infinite number of solutions. The number of time transitions a χ process can have might also be infinite: if a process can delay for t time units, then, for every $0 \leq t' \leq t$, it can delay for t' time units too. Since the time domain of χ is the set of real numbers, there are infinitely many time transitions.

Function Tr' returns a reduced set of transitions. Instead of returning all time transitions of a time step, for each trajectory only the time transition with longest duration is returned. Although this reduced set of transitions can still be infinite, in practice, this is rarely the case.

The subset χ_{sub} of the χ language for which the stepper functions are defined consists of processes $\langle p, \sigma, (C, J, L, H, R)\rangle$, where $p \in P_{\mathrm{T}}$. Here, P_{T} is defined by:

$$
\begin{array}{rcl}
P_{\mathrm{T}} & ::= & W : r \gg l_{\mathrm{a}} \;\mid\; h \,!!\, \mathbf{e}_n \;\mid\; h \,??\, \mathbf{x}_n \;\mid\; u \;\mid\; \delta \;\mid\; \bot \;\mid\; [P_{\mathrm{T}}] \\
 & \mid & u \curvearrowright P_{\mathrm{T}} \;\mid\; P_{\mathrm{T}};\, P_{\mathrm{T}} \;\mid\; b \rightarrow P_{\mathrm{T}} \;\mid\; P_{\mathrm{T}} \,[\!]\, P_{\mathrm{T}} \;\mid\; P_{\mathrm{T}} \parallel P_{\mathrm{T}} \\
 & \mid & \partial_A(P_{\mathrm{T}}) \;\mid\; \upsilon_H(P_{\mathrm{T}}) \;\mid\; X \;\mid\; \iota_{J^+}(P_{\mathrm{T}})
\end{array}
$$

The syntax of the χ processes is restricted such that there are no scope operators, and the set A of actions to be encapsulated is restricted as follows: $\forall_{x,y \in \Lambda^*, h \in H}\ \mathrm{isa}(h, [x]) \in A \implies \mathrm{isa}(h, [y]) \in A$ and $\forall_{x,y \in \Lambda^*, h \in H, v_{\mathrm{set}} \subseteq \mathcal{P}(\mathcal{V})}\ \mathrm{ira}(h, [x], v_{\mathrm{set}}) \in A \implies \mathrm{ira}(h, [y], v_{\mathrm{set}}) \in A$. The intuition of this restriction is that action encapsulation is based on channel names and

not on the values sent or received. Finally, the use of recursion is restricted to guarded recursion [Mil80, Mil82].

An implementation of the stepper functions may impose additional restrictions on the χ_{sub} syntax. For instance, for an implementation of function Tr' a (symbolic) solver is needed to compute the solutions of action predicates, the solution of delay predicates, and the maximum duration of a time transition. Depending on the solver that is used, additional restrictions may be required.

6.1.1 Function $\mathcal{S}$

Function $\mathcal{S}$ returns the set of steps given a χ_{sub} process. It is defined as $\mathcal{S}(p,\sigma,E) = \{(p,\sigma,E,f_{\mathrm{a}})|f_{\mathrm{a}} \in \mathcal{S}_{\mathrm{a}}(p,E)\} \cup \{(p,\sigma,E,f_{\mathrm{d}})|f_{\mathrm{d}} \in \mathcal{S}_{\mathrm{d}}(p,E)\}$. Given a χ process, functions $\mathcal{S}_{\mathrm{a}}$ and $\mathcal{S}_{\mathrm{d}}$ return the action step fragments and the time step fragments, respectively. The (only) difference between an action (time) step fragment and an action (time) step is that in the latter the χ_{sub} process (p,σ,E) is included. Step fragments are introduced in order to increase the readability of the definitions of functions $\mathcal{S}_{\mathrm{a}}$ and $\mathcal{S}_{\mathrm{d}}$ which are defined below.

Function $\mathcal{S}_{\mathrm{a}}$ Function $\mathcal{S}_{\mathrm{a}}$ returns a set of action step fragments of a χ_{sub} process. Formally, it is defined as follows.

$$
\begin{array}{lcl}
\mathcal{S}_{\mathrm{a}}(\langle W : r \gg l_{\mathrm{a}}, E\rangle) & = & \{(\mathrm{true}, W, r, l_{\mathrm{a}}, \mathrm{true}, \mathrm{true}, \checkmark)\} \\
\mathcal{S}_{\mathrm{a}}(\langle h \,!!\, \mathbf{e}_n, E\rangle) & = & \{(\mathrm{true}, \emptyset, \mathrm{true}, \mathrm{isa}(h, [\mathbf{e}_n]), \mathrm{true}, \mathrm{true}, \checkmark)\} \\
\mathcal{S}_{\mathrm{a}}(\langle h \,??\, \mathbf{x}_n, E\rangle) & = & \{(\mathrm{true}, \{\mathbf{x}_n\}, \mathrm{true}, \mathrm{ira}(h, [\mathbf{x}_n])), \mathrm{true}, \mathrm{true}, \checkmark)\} \\
\mathcal{S}_{\mathrm{a}}(\langle u, E\rangle & = & \emptyset \\
\mathcal{S}_{\mathrm{a}}(\langle \delta, E\rangle & = & \emptyset \\
\mathcal{S}_{\mathrm{a}}(\langle \bot, E\rangle & = & \emptyset \\
\mathcal{S}_{\mathrm{a}}(\langle [p], E\rangle) & = & \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle) \\
\mathcal{S}_{\mathrm{a}}(\langle u \curvearrowright p, E\rangle) & = & \{(u \wedge c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p') \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p') \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle)\} \\
\mathcal{S}_{\mathrm{a}}(\langle p;\ q, E\rangle) & = & \{(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p \wedge \mathcal{C}_{\mathrm{c}}(q,E), q) \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, \checkmark) \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle)\} \\
 & \cup & \{(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p';\ q) \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p') \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle), p' \neq \checkmark\} \\
\mathcal{S}_{\mathrm{a}}(\langle b \rightarrow p, E\rangle) & = & \{(b \wedge c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p') \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p') \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle)\} \\
\mathcal{S}_{\mathrm{a}}(\langle p \,[\!]\, q, E\rangle) & = & \{(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p \wedge \mathcal{C}_{\mathrm{c}}(q,E), C^a_p, p') \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, p') \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle)\} \\
 & \cup & \{c_q, W_q, r_q, l_{\mathrm{a}q}, C^b_q \wedge \mathcal{C}_{\mathrm{c}}(p,E), C^a_q, q') \\
 & & |(c_q, W_q, r_q, l_{\mathrm{a}q}, C^b_q, C^a_q, q') \in \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle)\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{S}_{\mathrm{a}}(\langle p \parallel q, E\rangle) & = & \{(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p \wedge \mathcal{C}_{\mathrm{c}}(q,E), C^a_p \wedge \mathcal{C}_{\mathrm{c}}(q,E), \begin{smallmatrix} q \\ p' \parallel q \end{smallmatrix}) \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle), p' \neq \checkmark\} \\
 & \cup & \{(c_q, W_q, r_q, l_{\mathrm{a}q}, C^b_q \wedge \mathcal{C}_{\mathrm{c}}(p,E), C^a_q \wedge \mathcal{C}_{\mathrm{c}}(p,E), \begin{smallmatrix} p \\ p \parallel q' \end{smallmatrix}) \\
 & & |(c_q, W_q, r_q, l_{\mathrm{a}q}, C^b_q, C^a_q, \begin{smallmatrix} \checkmark \\ q' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle), q' \neq \checkmark\} \\
 & \cup & \{(c_p \wedge c_q, \{\mathbf{x}_n\}, r_p \wedge r_q \wedge \mathbf{x}_n = \mathbf{e}^-_n, \mathrm{ca}(h, [\mathbf{e}_n]), C^b_p \wedge C^b_q, C^a_p \wedge C^a_q, \begin{smallmatrix} \checkmark \\ p' \\ q' \\ p' \parallel q' \end{smallmatrix}) \\
 & & |(c_p, W_p, r_p, \mathrm{isa}(h, [\mathbf{e}_n]), C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ p' \\ \checkmark \\ p' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle p, (C, J \cup \{\mathbf{x}_n\}, L, H, R)\rangle) \\
 & & , (c_q, W_q, r_q, \mathrm{ira}(h, [\mathbf{x}_n]), C^b_q, C^a_q, \begin{smallmatrix} \checkmark \\ q' \\ q' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle) \\
 & & , p' \neq \checkmark, q' \neq \checkmark\} \\
 & \cup & \{(c_p \wedge c_q, \{\mathbf{x}_n\}, r_p \wedge r_q \wedge \mathbf{x}_n = \mathbf{e}^-_n, \mathrm{ca}(h, [\mathbf{e}_n]), C^b_p \wedge C^b_q, C^a_p \wedge C^a_q, \begin{smallmatrix} \checkmark \\ p' \\ q' \\ q' \parallel p' \end{smallmatrix}) \\
 & & |(c_p, W_p, r_p, \mathrm{ira}(h, [\mathbf{x}_n]), C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ p' \\ \checkmark \\ p' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle) \\
 & & , (c_q, W_q, r_q, \mathrm{isa}(h, [\mathbf{e}_n]), C^b_q, C^a_q, \begin{smallmatrix} \checkmark \\ q' \\ q' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle q, (C, J \cup \{\mathbf{x}_n\}, L, H, R)\rangle) \\
 & & , p' \neq \checkmark, q' \neq \checkmark\} \\
\mathcal{S}_{\mathrm{a}}(\langle \partial_A(p), E\rangle) & = & \{(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ \partial_A(p') \end{smallmatrix}) \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle), l_{\mathrm{a}p} \notin A, p' \neq \checkmark\} \\
\mathcal{S}_{\mathrm{a}}(\langle \upsilon_H(p), E\rangle) & = & \{(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ \upsilon_H(p') \end{smallmatrix}) \\
 & & |(c_p, W_p, r_p, l_{\mathrm{a}p}, C^b_p, C^a_p, \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}) \in \mathcal{S}_{\mathrm{a}}(\langle p, E\rangle), p' \neq \checkmark\} \\
\mathcal{S}_{\mathrm{a}}(\langle X, (C, J, L, H, R)\rangle) & = & \mathcal{S}_{\mathrm{a}}(\langle R(X), (C, J, L, H, R)\rangle) \\
\mathcal{S}_{\mathrm{a}}(\langle \iota_{J^+}(p), (C, J, L, H, R)\rangle) & = & \mathcal{S}_{\mathrm{a}}(\langle p, (C, J \cup J^+, L, H, R)\rangle)
\end{array}
$$

Notation $[\mathbf{e}_n]$ denotes the list of expressions $[e_1, \ldots, e_n]$ for $n \geq 1$. In case that n equals 0, $[\mathbf{e}_n]$ denotes an empty list. Notation $\mathbf{x}_n = \mathbf{e}_n^-$ denotes the predicate $x_1 = e_1^- \wedge \cdots \wedge x_n = e_n^-$.

Function $\mathcal{C}_\mathrm{c}$ returns the consistency predicate which has to be satisfied in order for p to be consistent. Formally, it is defined as follows.

$$
\begin{array}{lcl}
\mathcal{C}_\mathrm{c}(W : r \gg l_\mathrm{a}, E) & = & \text{true} \\
\mathcal{C}_\mathrm{c}(h \,!!\, \mathbf{e}_n, E) & = & \text{true} \\
\mathcal{C}_\mathrm{c}(h \,??\, \mathbf{x}_n, E) & = & \text{true} \\
\mathcal{C}_\mathrm{c}(u, E) & = & u \\
\mathcal{C}_\mathrm{c}(\delta, E) & = & \text{true} \\
\mathcal{C}_\mathrm{c}(\bot, E) & = & \text{false} \\
\mathcal{C}_\mathrm{c}([p], E) & = & \text{true} \\
\mathcal{C}_\mathrm{c}(u \curvearrowright p, E) & = & u \wedge \mathcal{C}_\mathrm{c}(p, E) \\
\mathcal{C}_\mathrm{c}(p;\ q, E) & = & \mathcal{C}_\mathrm{c}(p, E) \\
\mathcal{C}_\mathrm{c}(b \rightarrow p, E) & = & (b \wedge \mathcal{C}_\mathrm{c}(p, E)) \vee \neg b \\
\mathcal{C}_\mathrm{c}(p \,[\!]\, q, E) & = & \mathcal{C}_\mathrm{c}(p, E) \wedge \mathcal{C}_\mathrm{c}(q, E) \\
\mathcal{C}_\mathrm{c}(p \parallel q, E) & = & \mathcal{C}_\mathrm{c}(p, E) \wedge \mathcal{C}_\mathrm{c}(q, E) \\
\mathcal{C}_\mathrm{c}(\partial_A(p), E) & = & \mathcal{C}_\mathrm{c}(p, E) \\
\mathcal{C}_\mathrm{c}(\upsilon_H(p), E) & = & \mathcal{C}_\mathrm{c}(p, E) \\
\mathcal{C}_\mathrm{c}(X, (C, J, L, H, R)) & = & \mathcal{C}_\mathrm{c}(R(X), (C, J, L, H, R)) \\
\mathcal{C}_\mathrm{c}(\iota_{J^+}(p), (C, J, L, H, R)) & = & \mathcal{C}_\mathrm{c}(p, (C, J \cup J^+, L, H, R))
\end{array}
$$

Theorem 6.1.1 *Let $p \in P_\mathrm{T}$, σ be a valuation, ξ be an extended valuation such that $\sigma = \xi \restriction \mathrm{dom}(\sigma)$, and E be an environment. Then*

$$\xi \models \mathcal{C}_\mathrm{c}(p, E) \Rightarrow \langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow} .$$

PROOF. See Appendix D.2.

As a conjecture, we have that $\langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow} \Rightarrow \xi \models \mathcal{C}_\mathrm{c}(p, E)$.

For the definition of the $\mathcal{S}_\mathrm{a}$ function, in addition to the propositions as presented in Section 3.5, the following two lemmas are used.

The first lemma shows that regarding termination and action transitions, the behavior of the guard operator and the signal emission operator is the same.

Lemma 6.1.1 *Let $p \in P$, b be a guard, σ, σ' be valuations, ξ, ξ' be extended valuations, a be an action label and E, E' be environments. Then*

$$\langle b \rightarrow p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}, \sigma', E' \rangle \iff \langle b \curvearrowright p, \sigma, E \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}, \sigma', E' \rangle .$$

PROOF. Since the termination and action transition rules defined for the guard opeartor and the signal emission operator are the same, the lemma holds trivially.

The second lemma shows that, for termination and action transitions, nesting of guards is the same as the conjuction of guards.

Lemma 6.1.2 *Let $p \in P$, b, b' be guards, σ, σ' be valuations, ξ, ξ' be extended valuations, a be an action label and E, E' be environments. Then*

$$\langle b \to b' \to p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E'\rangle \iff \langle b \wedge b' \to p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E'\rangle.$$

PROOF. Since the termination and action transition rules defined for the guard operator and the signal emission operator are the same, it follows directly from Lemma B.2.4 (conditions 1 – 3).

Function $\mathcal{S}_{\mathrm{d}}$ Function $\mathcal{S}_{\mathrm{d}}$ returns a set of time step fragments of a χ_{sub} process. Formally, it is defined as follows.

$$
\begin{array}{lcl}
\mathcal{S}_{\mathrm{d}}(\langle W : r \gg l_{\mathrm{a}}, E\rangle) & = & \emptyset \\
\mathcal{S}_{\mathrm{d}}(\langle h \,!!\, \mathbf{e}_n, E\rangle) & = & \emptyset \\
\mathcal{S}_{\mathrm{d}}(\langle h \,??\, \mathbf{x}_n, E\rangle) & = & \emptyset \\
\mathcal{S}_{\mathrm{d}}(\langle u, E\rangle & = & \{(u, u, u, u, \mathrm{true}, u)\} \\
\mathcal{S}_{\mathrm{d}}(\langle \delta, E\rangle & = & \emptyset \\
\mathcal{S}_{\mathrm{d}}(\langle \bot, E\rangle & = & \emptyset \\
\mathcal{S}_{\mathrm{d}}(\langle [p], E\rangle) & = & \{(\mathrm{true}, \mathrm{true}, \mathrm{true}, \mathrm{true}, \mathrm{true}, [p])\} \\
\mathcal{S}_{\mathrm{d}}(\langle u \curvearrowright p, E\rangle) & = & \{(u \wedge c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle p;\ q, E\rangle) & = & \{(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p';\ q) \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle b \to p, E\rangle) & = & \{(b \wedge c_p^{[0]}, b \wedge c_p^{(0,t)}, b \wedge c_p^{[t]}, b \wedge c_p^{[0,t]}, c_p, b \to p') \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E)\} \\
 & \cup & \{(b \implies c_p^{[0]}, \neg b, b \implies \mathcal{C}_{\mathrm{c}}(\langle p, E\rangle), \mathrm{true}, \neg b, b \to p) \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle p \,[\!]\, q, E\rangle) & = & \{(c_p^{[0]} \wedge c_q^{[0]}, c_p^{(0,t)} \wedge c_q^{(0,t)}, c_p^{[t]} \wedge c_q^{[t]}, c_p^{[0,t]} \wedge c_q^{[0,t]}, c_p \wedge c_q, p' \,[\!]\, q') \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E) \\
 & & , (c_q^{[0]}, c_q^{(0,t)}, c_q^{[t]}, c_q^{[0,t]}, c_q, q') \in \mathcal{S}_{\mathrm{d}}(q, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle p \parallel q, E\rangle) & = & \{(c_p^{[0]} \wedge c_q^{[0]}, c_p^{(0,t)} \wedge c_q^{(0,t)}, c_p^{[t]} \wedge c_q^{[t]}, c_p^{[0,t]} \wedge c_q^{[0,t]}, c_p \wedge c_q, p' \parallel q') \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E) \\
 & & , (c_q^{[0]}, c_q^{(0,t)}, c_q^{[t]}, c_q^{[0,t]}, c_q, q') \in \mathcal{S}_{\mathrm{d}}(q, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle \partial_A(p), E\rangle) & = & \{(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, \partial_A(p')) \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle \upsilon_H(p), E\rangle) & = & \{(c_p^{[0]} \wedge \neg c_p^{\mathrm{u}}, c_p^{(0,t)} \wedge \neg c_p^{\mathrm{u}}, c_p^{[t]}, c_p^{[0,t]}, c_p, \upsilon_H(p')) \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(p, E)\} \\
\mathcal{S}_{\mathrm{d}}(\langle X, (C, J, L, H, R)\rangle) & = & \mathcal{S}_{\mathrm{d}}(\langle R(X), (C, J, L, H, R)\rangle) \\
\mathcal{S}_{\mathrm{d}}(\langle \iota_{J^+}(p), (C, J, L, H, R)\rangle) & = & \{(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, \iota_{J^+}(p')) \\
 & & |(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(\langle p, (C, J \cup J^+, L, H, R)\rangle)\}
\end{array}
$$

Here, c_p^{u} is defined as $c_p^{\mathrm{u}} = \bigvee_{c\,:\,c \in \{c_p \mid (c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c'_p, p_{\mathrm{s}}) \in \mathcal{S}_{\mathrm{d}}(p, E),\ (c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_{p_{\mathrm{s}}}, p') \in \mathcal{S}_{\mathrm{d}}(p_{\mathrm{s}}, E),\ (c_p, W_p, r_p, \mathrm{ca}(h, [\mathbf{e}_n]), C_p^b, C_p^a, p') \in \mathcal{S}_{\mathrm{a}}(\langle p_{\mathrm{s}}, E\rangle)\}} c$, where $[\mathbf{e}_n]$ denotes a list of expressions, and h denotes a channel.

6.1.2 Transition functions

Function Tr returns a set of transitions given a χ_{sub} process. It is defined as $\mathrm{Tr} = \mathrm{Tr}_{\mathrm{a}} \cup \mathrm{Tr}_{\mathrm{d}}$. Function Tr_{a} is defined as follows.

$$
\begin{aligned}
&\mathrm{Tr_a}(p,\sigma,(C,J,L,H,R)) = \\
&\quad\{\langle p,\sigma,(C,J,L,H,R)\rangle \xrightarrow{\xi,\mathcal{M}_{\mathrm{tr}}(\xi,l_{\mathrm{a}p},\xi'),\xi'} \langle p',\xi'_\sigma,(C,J,L,H,R)\rangle \\
&\quad|(c_p,W_p,r_p,l_{\mathrm{a}p},C^b_p,C^a_p,p') \in \mathcal{S}_{\mathrm{a}}(\langle p,E\rangle) \\
&\quad,\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_p, \xi' \in \Xi(\sigma,C,J\cup W_p,L), \xi^- \cup \xi' \models r_p, \xi \models C^b_p, \xi' \models C^a_p,\}
\end{aligned}
$$

Function $\mathcal{M}_{\mathrm{tr}}$ is defined as follows, where $[\mathbf{e}_n]$ denotes a list of expressions, $[\mathbf{c}_n]$ denotes a list of values, $[\mathbf{x}_n]$ denotes a list of variables, and h denotes a channel:

$$
\mathcal{M}_{\mathrm{tr}}(\xi,l_{\mathrm{a}},\xi') = \begin{cases} \mathrm{isa}(h,[\xi(\mathbf{e}_n)]) & \text{if } l_{\mathrm{a}} \equiv \mathrm{isa}(h,[\mathbf{e}_n]) \\ \mathrm{ira}(h,[\xi'(\mathbf{x}_n)],\{\mathbf{x}_n\}) & \text{if } l_{\mathrm{a}} \equiv \mathrm{ira}(h,[\mathbf{x}_n]) \\ \mathrm{ca}(h,[\xi(\mathbf{e}_n)]) & \text{if } l_{\mathrm{a}} \equiv \mathrm{ca}(h,[\mathbf{e}_n]) \\ l_{\mathrm{a}} & \text{otherwise.} \end{cases}
$$

Function $\mathrm{Tr_d}$ is defined as follows.

$$
\begin{aligned}
&\mathrm{Tr_d}(p,\sigma,(C,J,L,H,R)) = \\
&\quad\{\langle p,\sigma,(C,J,L,H,R)\rangle \overset{t,\rho}{\longmapsto} \langle p',\rho_\sigma,(C,J,L,H,R)\rangle \\
&\quad|(c_p^{[0]},c_p^{(0,t)},c_p^{[t]},c_p^{[0,t]},c_p,p') \in \mathcal{S}_{\mathrm{d}}(\langle p,E\rangle) \\
&\quad,\rho \in \Omega_{FG}(\sigma,C,L,c_p^{[0,t]},t), \rho(0) \models c_p^{[0]}, \forall_{s\in(0,t)}\rho(s) \models c_p^{(0,t)}, \rho(t) \models c_p^{[t]}, \exists_{s\in[0,t]}\rho(s) \models c_p\}
\end{aligned}
$$

Function Tr' is defined as $\mathrm{Tr}' = \mathrm{Tr_a} \cup \mathrm{Tr}'_{\mathrm{d}}$, where function $\mathrm{Tr}'_{\mathrm{d}}$ is defined as follows.

$$
\begin{aligned}
&\mathrm{Tr}'_{\mathrm{d}}(p,\sigma,(C,J,L,H,R)) = \\
&\quad\{\langle p,\sigma,(C,J,L,H,R)\rangle \overset{t_{\max},\rho}{\longmapsto} \langle p',\rho_\sigma,(C,J,L,H,R)\rangle \\
&\quad|\langle p,\sigma,(C,J,L,H,R)\rangle \overset{t_{\max},\rho}{\longmapsto} \langle p',\rho_\sigma,(C,J,L,H,R)\rangle \in \mathrm{Tr_d}(p,\sigma,(C,J,L,H,R)) \\
&\quad,\nexists_{t>t_{\max},\rho'}\ (\ \langle p,\sigma,(C,J,L,H,R)\rangle \overset{t,\rho'}{\longmapsto} \langle p'',\rho'_\sigma,(C,J,L,H,R)\rangle \in \mathrm{Tr_d}(p,\sigma,(C,J,L,H,R)) \\
&\qquad\qquad,\rho' \restriction [0,t_{\max}] = \rho) \\
&\}.
\end{aligned}
$$

Theorem 6.1.2 *Let p, $p' \in P_{\mathrm{T}}$, σ, σ' be valuations, ξ, ξ' be extended valuations, a be an action label, and E be an environment. Then*

$$
\langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',E\rangle \in \mathrm{Tr_a}(p,\sigma,E) \Rightarrow \langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',E\rangle.
$$

PROOF. See Appendix D.3.

As a conjecture, we have that $\langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',E\rangle \Rightarrow \langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',E\rangle \in \mathrm{Tr_a}(p,\sigma,E)$.

Conjecture 6.1.1 *Let p, $p' \in P_{\mathrm{T}}$, σ, σ' be valuations, $t \in T$, ρ be a trajectory, and E be an environment. Then*

$$
\langle p,\sigma,E\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',E\rangle \in \mathrm{Tr_d}(p,\sigma,E) \Rightarrow \langle p,\sigma,E\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',E\rangle.
$$

PROOF. See Appendix D.4.

As another conjecture, we have that $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle \Rightarrow \langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle \in \mathrm{Tr_d}(p, \sigma, E)$.

6.2 Simulator

Simulation is a powerful method to analyse the dynamic behavior of a model. In this section, a simulator is defined which is mainly based on the stepper functions from Section 6.1. Using pseudocode, the simulator is defined as follows.

```
Simulate(⟨p, σ, E⟩) =
                    while p ≠ ✓ do
                      transitions := Tr'(⟨p, σ, E⟩)
                      if transitions = ∅ then
                        return deadlock
                      else
                        transition := pick(transitions)
                        ⟨p, σ, E⟩ := GetState(transition)
                      endif
                    endwhile
                    return simulation_ended
```

Given a χ_{sub} process $\langle p_{\mathrm{s}}, \sigma, E\rangle$, function Tr' returns the set of possible transitions *transitions*. The process deadlocks (**return** *deadlock*) if the set of transitions is empty. Otherwise, a transition (*transition*) is selected (non-deterministically) from the set of transitions. The resulting state of the transition is obtained by means of function GetState. If the process is terminated ($p = \checkmark$), the simulation is succesfully terminated (**return** *simulation_ended*).

6.3 Chi2HA translator

The translation from χ to hybrid automata with urgency is formally defined in Section 5.3. Since a manual translation of a χ specification is quite laborous and therefore error-prone, this translation has been automated.

For the purpose of visualization of the hybrid automata obtained by this translation and automatic verification of some properties of them using tools based on hybrid automata, codegenerators have been developed.

6.4 Third party tools

The χ toolset is designed in such a way that it can be integrated with third party tools. The main reason for this is to reuse existing applications and libraries. For the implementation

of the tools, the Python [Pyt05] programming language has been used.

Currently, the χ tools are integrated with four different tools: Maple, Gnuplot, Graphviz, and PHAver.

For the computation of solutions of action predicates and delay predicates, and the maximum duration of time transitions, the symbolic solving capabilities of the mathematical package Maple [Map] are used.

For the vizualization of simulation results, in particular the trajectories of time transitions, the portable command-line driven interactive data and function plotting utility Gnuplot [Gnu] is used.

For visualization of the hybrid transition systems obtained by means of the simulator and the hybrid automata that are obtained by means of the Chi2HA translator, the Graphviz tools are used. Graphviz [EN00] is an open toolkit for graph visualization. It is developed at AT&T Labs-Research. The Graphviz tools use a common language to specify attributed graphs. This language is called *Libgraph*, but is probably better known as the *dot* format, after its best-known application. Graphviz provides tools for graph filtering and graph rendering. The filtering tools can be batch-oriented as well as interactive. For our application, visualization of hybrid automata, we only need a small part of the functionality offered by Graphviz. For instance, in the hybrid automaton definition there are only tree types of nodes/locations(initial, termination and normal) and two different types of edges (urgent and non-urgent) in our graphs.

For verification purposes, PHAVer (Polyhedral Hybrid Automaton Verifyer) [Fre05] is used (for details we refer to Section 7.2). PHAVer is a tool for analyzing linear hybrid I/O-automata.

CHAPTER

SEVEN

Analysis of hybrid systems: Case studies

7.1 Case study using simulator

Figure 7.1 shows a bottle filling line taken from [**?**]. It consists of a storage tank that is continuously filled with a flow Q_{in}, a conveyor belt that supplies empty bottles, and a valve that is opened when an empty bottle is below the filling nozzle, and is closed when the bottle is full. When a bottle has been filled, the conveyor starts moving to put the next bottle under the filling nozzle, which takes one unit of time. When the storage tank is not empty, the bottle filling flow Q equals Q_{set}. When the storage tank is empty, the bottle filling flow equals the flow Q_{in}. We assume $Q_{\text{in}} < Q_{\text{set}}$.

Figure 7.1: Filling Line

The model is defined below. The constants V_{T0}, V_{Tmax}, and V_{Bmax} define the initial volume of the storage tank, and the maximum volume of the storage tank, and the filling volume of the bottles, respectively. The constants Q_{in}, and Q_{set}, define the value of the flow that is used to fill the storage tank, and the maximum value of the bottle filling flow Q.

$\langle$ disc x, Q
, cont $V_{\text{T}}, V_{\text{B}}$
, $x = 0.0, Q = 0.0, V_{\text{T}} = V_{\text{T0}}, V_{\text{B}} = 0.0$

$$
\begin{array}{lll}
, \mathit{closed} & \mapsto & \dot{V}_{\mathrm{T}} = Q_{\mathrm{in}}, V_{\mathrm{T}} \leq V_{\mathrm{Tmax}} \,[\!]\, \mathit{open}\,?\,Q;\ \mathit{opened} \\
, \mathit{opened} & \mapsto & ((\dot{V}_{\mathrm{T}} = Q_{\mathrm{in}} - Q, V_{\mathrm{T}} \geq 0.0 \parallel V_{\mathrm{T}} \leq V_{\mathrm{Tmax}}) \\
 & & [\!]\ \mathit{close}\,?\,Q;\ \mathit{closed} \\
 & & [\!]\ [\{Q\} : Q = 1.5 \gg c];\ \mathit{openedempty} \\
 & &) \\
, \mathit{openedempty} & \mapsto & (\dot{V}_{\mathrm{T}} = 0.0, V_{\mathrm{T}} = 0.0 \\
 & & [\!]\ \mathit{close}\,?\,Q;\ \mathit{closed} \\
 & &) \\
, \mathit{moving} & \mapsto & (\{V_{\mathrm{B}}, x\} : V_{\mathrm{B}} = 0.0, x = \mathsf{time} + 1.0 \gg a \\
 & & ;\ \mathsf{time} \geq x \rightarrow \mathrm{skip};\ \mathit{open}\,!\,Q_{\mathrm{set}};\ \mathit{filling} \\
 & &) \\
, \mathit{filling} & \mapsto & (V_{\mathrm{B}} \geq V_{\mathrm{Bmax}} \rightarrow \mathit{close}\,!\,0.0;\ \mathit{moving}) \\
\mid \mathit{closed} \parallel \mathit{moving} \parallel \dot{V}_{\mathrm{B}} = Q & & \\
\rangle & &
\end{array}
$$

The storage tank is modelled by means of recursion variables / modes: *closed*, *opened*, and *openedempty* that correspond to the valve being open, the valve being closed, and the valve being opened while the storage tank is empty. In the mode *opened*, the storage tank is usually not empty. When the storage tank is empty in mode *opened*, the delayable skip statement [skip] may be executed causing the next mode to be *openedempty*. Due to the consistent equation semantics, the skip statement can be executed only if the delay predicate in the next mode *openedempty* holds. This means, among others, that $V_{\mathrm{T}} = 0.0$ must hold. Therefore, the transition to mode openedempty can be taken only when the storage tank is empty. The initial mode is *closed*.

The conveyor is modelled by means of recursion variables / modes *moving* and *filling*. In mode *moving*, the conveyor supplies an empty bottle in 1 unit of time ($\{V_{\mathrm{B}}, x\} : V_{\mathrm{B}} = 0.0, x = \mathsf{time} + 1.0 \gg a$; $\mathsf{time} \geq x \rightarrow$ skip. Then it synchronizes with the storage tank process by means of the send statement $\mathit{open}!Q_{\mathrm{set}}$, and it proceeds in mode *filling*. When the bottle is filled in mode filling ($V_{\mathrm{B}} \geq V_{\mathrm{Bmax}}$), the process synchronizes with the storage tank to close the valve and returns to mode moving. The initial mode is *moving*. Figure 7.2 shows the result of a simulation run of the model using the following values for the constants: $V_{\mathrm{T0}} = 5.0$, $V_{\mathrm{Bmax}} = 10.0$, $V_{\mathrm{Tmax}} = 20.0$, $Q_{\mathrm{in}} = 1.5$, and $Q_{\mathrm{set}} = 3.0$,

7.2 Analysis of χ_{sub} specifications using PHAVer

Hybrid systems can be modeled as hybrid automata and verified using reachability analysis techniques. Over the past years, various hybrid systems techniques (hybrid automata based) and tools (e.g. HyTech [HHWT97], PHAVer [Fre05] and d/dt [ABDM00]) have been successfully introduced into the verification of hybrid systems.

PHAVer (Polyhedral Hybrid Automaton Verifyer) is a tool for analyzing linear hybrid I/O-automata (i.e., a sub-class of hybrid I/O-automata and see the definition in Section 7.2.2) with the following characteristics:

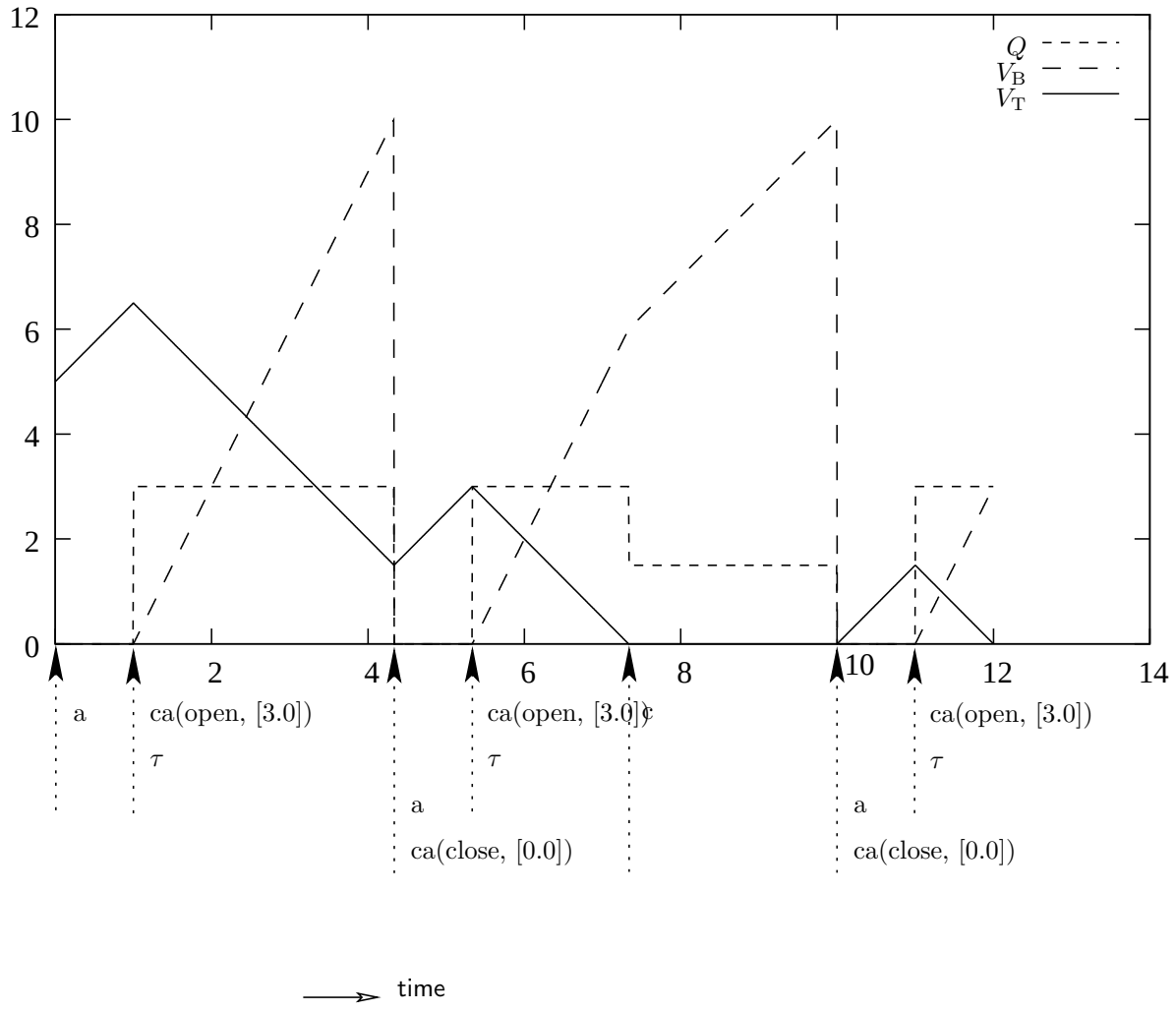

Figure 7.2: Simulation results of model *FillingLine*.

1. exact and robust arithmetic based on the Parma Polyhedra Library,
2. on-the-fly overapproximation of piecewise affine dynamics,
3. conservative limiting of bits and constraints in polyhedral computations,
4. support for compositional and assume-guarantee reasoning.

PHAVer appears to be well suited for verifying χ_{sub} specifications. In Section 5.3, a class of hybrid automata has been defined that is expressive enough to be used to give the interpretations of χ_{sub} in hybrid automata. This class of hybrid automata is closely related to various classes of hybrid automata, e.g., hybrid I/O-automata [LSV03]. This class of hybrid automata is in the following referred to as HA_{u}, where the u represents urgency. A formal translation of χ_{sub} to hybrid automata HA_{u} has been defined in Section 5.3. This

translation enables verification of χ_{sub} specifications using existing hybrid automata based verification tools. Currently, we use PHAVer as an verification engine for χ_{sub} specifications.

7.2.1 Hybrid I/O-automata

In this section, the syntax and semantics of hybrid I/O-automata (based on [FHK04, Fre04a]) are given.

In the definition of hybrid I/O-automata, some functions and notations may be used. Given a set Var of variables, a *valuation* $\beta \in \text{Var} \to \mathbb{R}$ maps a real number to each variable. Let V(Var) denote the set of valuations over Var. An *activity* is a function $f \in \mathbb{R}_{\geq 0} \to V$ in C^∞ (i.e. a function is C^∞ if the n-th derivative exists and is continuous for all n) and describes the change of valuations over time. Let act(Var) denote the set of activities over Var. Let $f + t$ for $f \in \text{act}(\text{Var})$ and $t \in \mathbb{R}_{\geq 0}$ be defined by $(f + t)(d) = f(d + t)$, $d \in \mathbb{R}_{\geq 0}$. A set $S \subseteq \text{act}(\text{Var})$ of activities is *time invariant* if for all $f \in S, t \in \mathbb{R}_{\geq 0} : f + t \in S$.

A hybrid I/O-automaton [Fre04a] $\mathit{HIOA} = (\text{Loc}, \text{Var}_\text{S}, \text{Var}_\text{I}, \text{Var}_\text{O}, \text{Lab}, \to, \text{Act}, \text{Inv}, \text{Init})$ consists of the following components:

- A finite set Loc of locations.
- Finite and disjoint sets of state and input variables, Var_S and Var_I, and of output variables $\text{Var}_\text{O} \subseteq \text{Var}_\text{S}$, and let $\text{Var} = \text{Var}_\text{S} \cup \text{Var}_\text{I}$.
- A finite set Lab of labels.
- A finite set of discrete transitions $\to \subseteq \text{Loc} \times \text{Lab} \times 2^{\text{V(Var)} \times \text{V(Var)}} \times \text{Loc}$. A transition $(l, a, \mu, l') \in \to$ is also written as $l \xrightarrow{a,\mu}_H l'$.
- A mapping (a labelling function) $\text{Act} : \text{Loc} \to 2^{\text{act(Var)}}$ from locations to time invariant sets of activities.
- A mapping $\text{Inv} : \text{Loc} \to 2^{\text{V(Var)}}$ from locations to sets of valuations.
- A set $\text{Init} \subseteq \text{Loc} \times \text{V(Var)}$ of initial states.

The semantics of a hybrid I/O-automaton is defined in terms of a timed transition system. Let $\mathit{HIOA} = (\text{Loc}, \text{Var}_\text{S}, \text{Var}_\text{I}, \text{Var}_\text{O}, \text{Lab}, \to, \text{Act}, \text{Inv}, \text{Init})$ be a hybrid I/O-automaton. A *state* of *HIOA* is a pair $(l, v) \in \text{Loc} \times \text{V(Var)}$ of a location and a valuation. The transition system interpretation of *HIOA*, written $[\![\mathit{HIOA}]\!]$, is the timed transition system $(\text{Loc}, \text{Var}_\text{S}, \text{Var}_\text{I}, \text{Var}_\text{O}, \Sigma', \to_{LH}, \text{Init})$, where $\Sigma' = \text{Lab} \cup \mathbb{R}_{\geq 0} \cup \{\epsilon\}$ and $\to_{LH}$ is the union of $\xrightarrow{a}$, for $a \in \Sigma'$;

- $(l, v) \xrightarrow{a}_{LH} (l', v')$ iff $l \xrightarrow{a,\mu}_H l'$, $(v, v') \in \mu$, $v \in \text{Inv}(l)$, $v' \in \text{Inv}(l')$ (discrete transitions),
- $(l, v) \xrightarrow{t}_{LH} (l', v')$ iff $l = l'$ and there exists $f \in \text{Act}(l)$, $f(0) = v$, $f(t) = v'$, and $\forall t'$, $0 \leq t' \leq t : f(t') \in \text{Inv}(l)$ (timed transitions),

- $(l, v) \xrightarrow{\epsilon}_{LH} (l', v')$ iff $l = l'$, $v \restriction \text{Var}_\text{S} = v' \restriction \text{Var}_\text{S}$, and $v, v' \in \text{Inv}(l)$ (environment transitions).

In this transition system, three kinds of transition relations are defined. They are differentiated by their label, the time elapse involved, and a special label ϵ that represents changes in the input variables by the environment.

7.2.2 Relating hybrid automata HA_u to hybrid I/O-automata

This section outlines the main differences (in both syntax and semantics) between hybrid automata HA_u and hybrid I/O-automata.

There are several syntactical and semantical differences between hybrid automata HA_u and hybrid I/O-automata. They can be summarized as follows:

- Some syntactical differences involve the definition of the invariants, initial conditions, etc. In the definition of hybrid automata HA_u, we introduce predicates (e.g. for invariants and initial conditions) into the locations of automata to restrict the behaviors of locations. Predicates are introduced as symbolic representations of sets of valuations. For instance, we consider predicate $x > 5$ as a symbolic representation of an infinite set of valuations $\{x \mapsto 6.5, x \mapsto 7.1, x \mapsto 8.3, \ldots\}$. On the other hand, in the definition of hybrid I/O-automata, sets of valuations are explicitly introduced into the locations of automata to restrict the behaviors of locations.

- Urgent transitions are defined in the semantics of hybrid automata HA_u, but they are not defined in the semantics of hybrid I/O-automata.

- Action transitions can also be guarded in the semantics of hybrid automata HA_u. This feature is not explicitly defined in the semantics of hybrid I/O-automata. In the definition of hybrid I/O-automata, the guard is implicitly embedded in the reset map.

- A special transition (labelled with ϵ) is defined in the semantics of hybrid I/O-automata. It represents changes in input variables by the environment (see also [FHK04]).

The equivalence between a sub-class of hybrid automata HA_u and a sub-class of hybrid I/O-automata is given in this section.

Theorem 7.2.1 *Let v_1, v_2 be locations, σ, σ' be valuations such that* range(σ), range(σ') $\in \mathbb{R}$, *$t \in \mathbb{R}_{\geq 0}$, ρ be a trajectory such that $\rho_\sigma(0) = \sigma$,and $\rho_\sigma(t) = \sigma'$, $HA_\text{u} = (X, V,$* init, inv, flow, *E,* source, target, urgent, guard, jump, Σ, event*) be a hybrid automaton HA_u as defined in Section 5.3, and HIOA* $= (\text{Loc}, \text{Var}_\text{S}, \text{Var}_\text{I}, \text{Var}_\text{O}, \text{Lab}, \rightarrow, \text{Act}, \text{Inv}, \text{Init})$ *be a hybrid I/O-automaton (as defined in Section 7.2.1). If*

1. $X = \text{Var}_\text{S}$,

2. $V = \mathrm{Loc}$,

3. $\mathrm{Var_I} = \mathrm{Var_O} = \emptyset$,

4. $\mathrm{Lab} = \Sigma$,

5. $\forall_{e \in E} : \mathrm{urgent}(e) = \mathrm{false}$,

6. $\forall_{\sigma,\sigma'} : \mathrm{source}(e) = l,\ (\sigma, \sigma') \models (\mathrm{jump}(e) \wedge \mathrm{guard}(e))$
$\Rightarrow (\mathrm{source}(e), \mathrm{event}(e),\ (\sigma, \sigma'), \mathrm{target}(e)) \in \rightarrow$,

7. $\forall_{l,l' \in \mathrm{Loc}, a \in \mathrm{Lab}, \sigma, \sigma'} :\ (l, a, (\sigma, \sigma'), l') \in \rightarrow\ \Rightarrow\ \exists_{e \in E} :\ \mathrm{source}(e) = l, \mathrm{target}(e) = l', (\sigma, \sigma') \models$
$\mathrm{jump}(e) \wedge \mathrm{guard}(e), \mathrm{event}(e) = a$,

8. $\forall_{\sigma, v \in V} :\ \sigma \models \mathrm{init}(v) \Leftrightarrow (v, \sigma) \in \mathrm{Init}(v)$,

9. $\forall_{\sigma, v \in V} :\ \sigma \models \mathrm{inv}(v) \Leftrightarrow (v, \sigma) \in \mathrm{Inv}(v)$,

10. $\forall_{\rho, t \in \mathrm{dom}(\rho), v \in V} : \rho(t) \models \mathrm{inv}(v) \wedge \mathrm{flow}(v)(t) \Rightarrow \rho \in \mathrm{Act}(v)$,

11. $\forall_{\rho, v \in V} : \rho \in \mathrm{Act}(v) \Rightarrow \forall_{t \in \mathrm{dom}(\rho)} \rho(t) \models \mathrm{inv}(v) \wedge \mathrm{flow}(v)(t)$,

then we have

1. $[\![\, HA_{\mathrm{u}} \,]\!] \models (v_1, \sigma) \xrightarrow{a} (v_2, \sigma') \ \Leftrightarrow\ [\![\, HIOA \,]\!] \models (v_1, \sigma) \xrightarrow{a}_{LH} (v_2, \sigma')$,

2. $[\![\, HA_{\mathrm{u}} \,]\!] \models (v_1, \sigma) \longmapsto (v_1, \sigma') \ \Leftrightarrow\ [\![\, HIOA \,]\!] \models (v_1, \sigma) \xrightarrow{t}_{LH} (v_1, \sigma')$.

PROOF. (Sketch). Theorem 7.2.1.1. First, we assume to have $[\![\, HA_{\mathrm{u}} \,]\!] \models (v_1, \sigma) \xrightarrow{a} (v_2, \sigma')$, which means that there exists an edge e with source location v_1, target location v_2, event a, and a guard and a jump condition that hold for α and α'. It follows from the hypothesis, there exists a transition $\in \rightarrow$ such that $[\![\, HIOA \,]\!] \models (v_1, \sigma) \xrightarrow{a}_{LH} (v_2, \sigma')$ holds. Second, the proof for $[\![\, HIOA \,]\!] \models (v_1, \sigma) \xrightarrow{a}_{LH} (v_2, \sigma') \Rightarrow [\![\, HA_{\mathrm{u}} \,]\!] \models (v_1, \sigma) \xrightarrow{a} (v_2, \sigma')$ is similar to the first case.

Theorem 7.2.1.2. First, we assume to have $[\![\, HA_{\mathrm{u}} \,]\!] \models (v_1, \sigma) \longmapsto (v_1, \sigma')$, which means that there is a ρ such that $\forall_{t \in \mathrm{dom}(\rho)} \rho(t) \models \mathrm{inv}(v_1) \wedge \mathrm{flow}(v_1)(t)$. It follows from the hypothesis that $\rho \in \mathrm{Act}(v_1)$. Hence, $(v_1, \sigma) \xrightarrow{t}_{LH} (v_1, \sigma')$ is in $[\![\, HIOA \,]\!]$. Second, the proof for $[\![\, HIOA \,]\!] \models (v_1, \sigma) \xrightarrow{a}_{LH} (v_2, \sigma') \Rightarrow [\![\, HA_{\mathrm{u}} \,]\!] \models (v_1, \sigma) \xrightarrow{a} (v_2, \sigma')$ is similar to the first case.

Since PHAVer is a tool for analyzing linear hybrid I/O-automata (i.e., a sub-class of hybrid I/O-automata), this section gives the definition of a linear constraint, a linear formula (based on [FHK04]), a linear hybrid automaton HA_{u}, and a linear hybrid I/O-automaton (taken from [FHK04]).

A linear constraint over a set of variables X is of the form $\sum_i a_i v_i + b \lozenge 0$, with $a_i, b \in \mathbb{Z}$, $v_i \in X$, and $\lozenge \in \{<, \leq, =\}$. For a given valuation σ of X such that $\mathrm{range}(\sigma) \in \mathbb{R}$, a linear constraint ϕ yields a boolean value on whether ϕ is satisfied ($\sigma \models \phi$) or not. A linear

formula (also known as linear predicate) is a boolean combination of linear constraints. A linear hybrid I/O-automaton is a hybrid I/O-automaton (as defined in Section 7.2.1) in which the invariants are given by linear formulas over Var, the state transformations are given by by a linear formulas over $\text{Var} \cup \text{Var}'$ ($\text{Var}' = \{x' \mid x \in \text{Var}\}$), and the activities are given by linear formulas (also known as linear predicate) over the time derivatives of the state variables, i.e., over $\dot{\text{Var}}_s = \{\dot{x} \mid x \in \text{Var}_s\}$.

Similarly, a linear hybrid automaton HA_{u} can be obtained from a hybrid automaton HA_{u} by restricting the predicates involved in the invariants, jump conditions, guards and flow conditions to be linear.

It is not hard to see that Theorem 7.2.1 also holds for the linear hybrid automaton HA_{u} and linear hybrid I/O-automaton trivially, because we only restrict the automata HA_{u} and *HIOA* to be linear.

Next, we discuss the relations between (linear) hybrid automata, (linear) hybrid I/O-automata and the input language of PHAVer.

Theorem 7.2.1 describes the relation between (linear) hybrid automata HA_{u} and (linear) hybrid I/O-automata. The input language of PHAVer (not formally defined) is a straightforward textual representation of linear hybrid I/O-automata [Fre04b]. Furthermore, the syntax of PHAVer's representation of automata (the input language) additionally allows guarded action transitions between two locations, i.e. a linear formula *trans_rel* over $\text{Var} \cup \text{Var}'$ is guarded by another linear formula *guard* over Var, and the conjunction of the two linear formulas $\mathit{guard} \wedge \mathit{trans\_rel}$ correspondes to the linear formula μ as defined in Section 7.2.2. The syntax of the input language of PHAVer (allowing guarded action transitions) makes it easy to give a straightforward textual representation of linear hybrid I/O-automata in PHAVer.

7.2.3 Example

The verification of a χ_{sub} specification using PHAVer is illustrated by means of an example: the water level monitor, which is taken from [ACH+95]. We first model the water-level monitor in χ_{sub}, then we translate the water-level monitor in χ_{sub} to a hybrid automaton HA_{u}.

The water level in a tank is controlled through a monitor, which continuously senses the water level and turns a pump on and off. When the pump is off, the water level, denoted by variable the y, drops by 2 units per second; when the pump is on, the water level rises by 1 per second. There is a time delay of 2 time units between the time that the monitor signals to change the status of the pump and time that the change becomes effective (this is modeled by the variable x). Initially the water level is 1 and the pump is turned on. The water-level monitor is modelled in χ_{sub} as follows:

$\langle$ cont x, y
, $x = 0, y = 1$
| $\dot{x} = 1$

$$
\begin{array}{l}
\| * (\ (\ \dot{y} = 1 \quad \wedge \quad y \leq 10 \quad [\!] \quad [y \geq 10 \rightarrow \{x\} : x = 0 \gg \tau]\ ) \\
\quad ;\ (\ \dot{y} = 1 \quad \wedge \quad x \leq 2 \quad [\!] \quad [x \geq 2 \rightarrow \{\emptyset\} : \text{true} \gg \tau]\ ) \\
\quad ;\ (\ \dot{y} = -2 \quad \wedge \quad y \geq 5 \quad [\!] \quad [y \leq 5 \rightarrow \{x\} : x = 0 \gg \tau]\ ) \\
\quad ;\ (\ \dot{y} = -2 \quad \wedge \quad x \leq 2 \quad [\!] \quad [x \geq 2 \rightarrow \{\emptyset\} : \text{true} \gg \tau]\ ) \\
\quad)\\
\rangle
\end{array}
$$

This specification is translated into a hybrid automaton HA_{u}, the graphical representation of the hybrid automaton HA_{u} is shown in Figure 7.3. This graphical representation is similar to the one as it was shown in [ACH+95], see Figure 7.4. The graphical representation of the hybrid automaton HA_{u} contains more predicates as a result of the translation. However these additional predicates do not influence the behavior of the hybrid automaton HA_{u}.

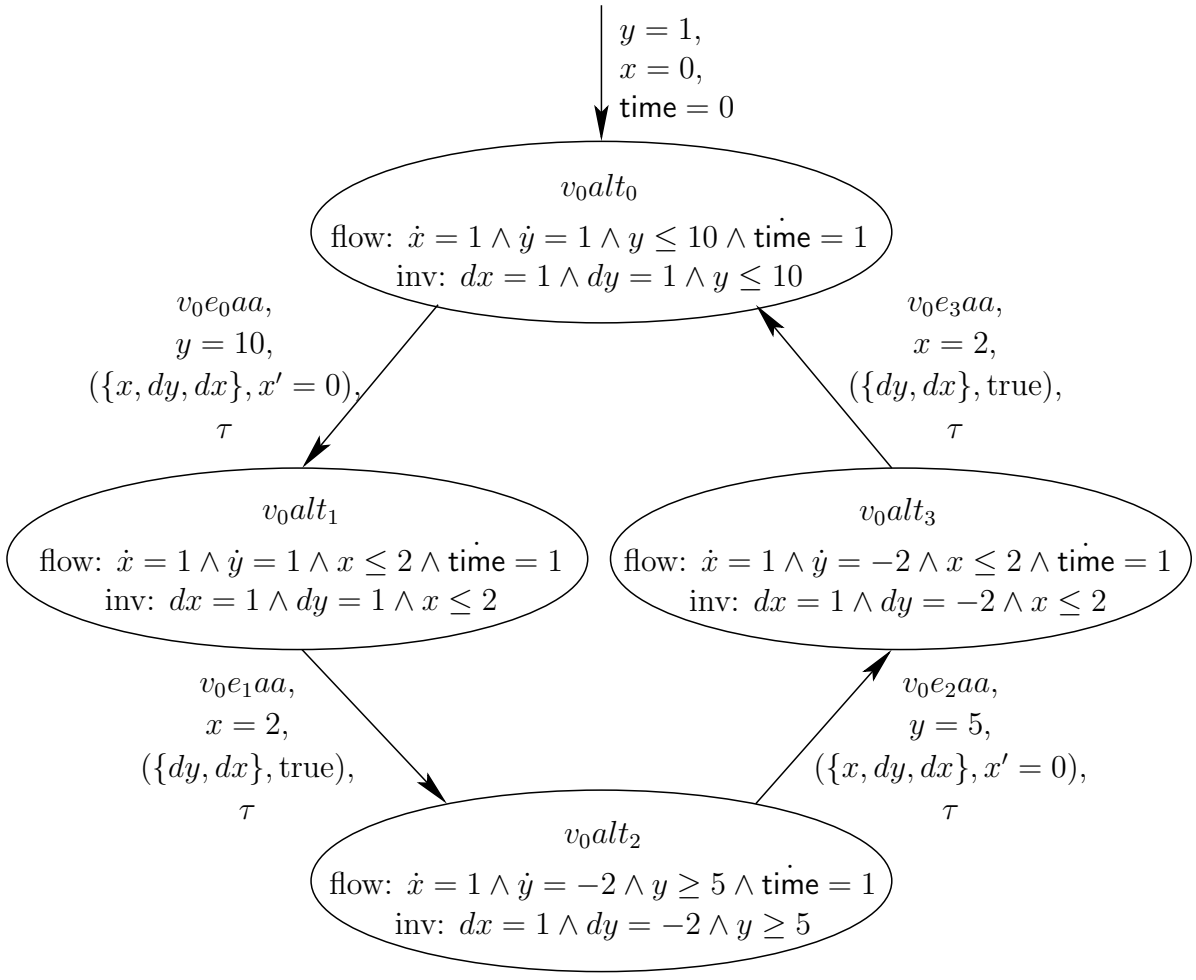

Figure 7.3: Generated water-level monitor automaton.

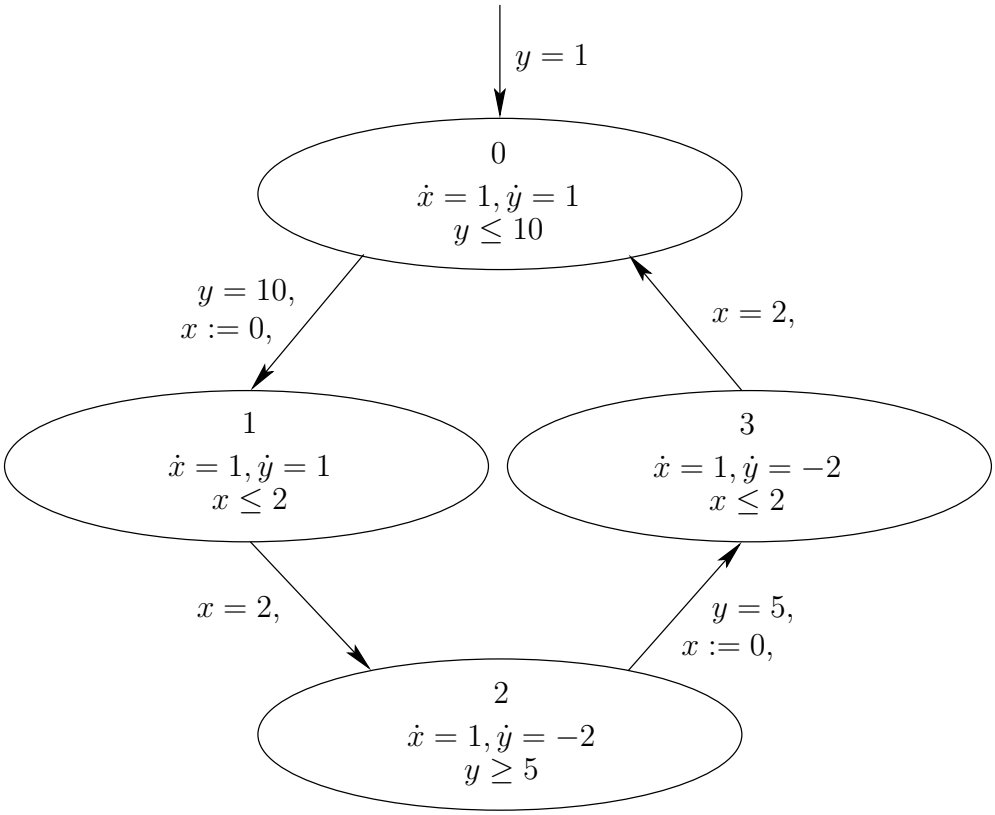

Figure 7.4: Water-level monitor automaton [ACH+95].

This hybrid automaton HA_{u} is linear, which means there exists a corresponding linear hybrid I/O-automaton (see also Theorem 7.2.1) that can be verified using PHAVer. Figure 7.5 shows such a corresponding linear hybrid I/O-automaton of the hybrid automaton HA_{u} described in the PHAVer input language. The safety property that the water level has to be kept between $1 < y < 12$ has been verified using PHAVer. PHAVer reported that this safety property held in all locations of the corresponding linear hybrid I/O-automaton. By Theorem 7.2.1, we know that this safety property also holds in the hybrid automaton HA_{u}. From Section 5.3.4, we know that any transition of a χ_{sub} specification can be mimicked by a transition in the corresponding hybrid automaton HA_{u} and vice versa, which indicates this safety property also holds in the original χ_{sub} specification.

```
automaton watermonitor
  state_var: y, x, time, dy, dx;
  synclabs: t;

  loc v0alt0: while dx == 1 & dy == 1 & y <= 10
    wait {x == 1 & y == 1 & y' <= 10 & time == 1};
    when y == 10 do {x' == 0 & y' == y & time' == time} sync t goto v0alt1;

  loc v0alt1: while dx == 1 & dy == 1 & x <= 2
    wait {x == 1 & y == 1 & x' <= 2 & time == 1};
    when x == 2 do {y' == y & x' == x & time' == time} sync t goto v0alt2;

  loc v0alt2: while dx == 1 & dy == -2 & y >= 5
    wait {x == 1 & y == -2 & y' >= 5 & time == 1};
    when y == 5 do {x' == 0 & y' == y & time' == time} sync t goto v0alt3;

  loc v0alt3: while dx == 1 & dy == -2 & x <= 2
    wait {x == 1 & y == -2 & x' <= 2 & time == 1};
    when x == 2 do {y' == y & x' == x & time' == time} sync t goto v0alt0;

  initially: v0alt0 & y == 1 & x == 0;
end

sys = watermonitor;
reg = sys.reachable;
reg.print;

forbidden = sys.{v0alt0 & y>12|y<0 , v0alt1 & y>12|y<0 ,
                 v0alt2 & y>12|y<0 , v0alt3 & y>12|y<0};

/* Safety property: the water level has to be kept for 1< y <12 */

echo "";
reg.intersection_assign(forbidden);
echo "Intersection with forbidden states:";
reg.is_empty;
```

Figure 7.5: PHAVer code of the water-level monitor.

CHAPTER

EIGHT

Elimination in Chi

As we have seen in Section 7.2, the formal translation from a reasonable subset (χ_{sub}) of χ to hybrid automata HA_{u} enables verification of χ_{sub} specifications using existing hybrid automaton based verification tools. The full χ formalism, however, is much richer.

The fact that the χ formalism is such a rich language potentially complicates the development of tools for χ, because the implementations have to deal with all possible combinations of the χ atomic process terms and the operators that are defined on them. This is, where the process algebraic approach of equational reasoning, that allows rewriting specifications to a simpler form, is essential.

Thus, we investigate the possibilties to develop efficient algorithms for the linearization of hybrid χ processes. In process algebras, linearization is a transformation of a recursive specification into a linear representation, i.e., a kind of normal form that is convenient for many forms of analysis. Note that these linear representations are expressed as recursive specifications as well, but they use only a small subset of the full process algebra. In general, such linear representations can also be considered very compact representations of a possibly infinite state space. The original recursive specification and its transformation are required to be bisimilar, which ensures that the relevant specification properties are preserved. Related works in this direction can be found in [Use02], [vdBRC04] and [BvBR06].

It is a quite difficult task to develop efficient algorithms for the linearization of hybrid χ processes. Our first attempt in this direction is to prove an elimination theorem for χ. Like in many process algebras (e.g. in ACP), it is possible to define a set of basic terms (without recursion definitions) and prove that all closed process terms are derivably equal to some basic terms, which makes it easier to perform inductive reasoning on the structure of a closed term. This is the so-called elimination thereom. Elimination in χ can also be regarded as our first step towards the linearization of χ processes.

In this chapter, we present a sub-language of χ (which we call χ_{S} for simplicity), which allows us to define a set of basic terms into which each closed term of χ_{S} can be rewritten. This eases the proofs of properties and the analysis of specifications described in χ_{S}, because many operators can be eliminated and any closed process term of χ_{S} can be rewritten into an equivalent term that uses only a few basic operators. We also show that parallel composition can be eliminated from all closed terms of χ_{S}. After the elimination of the parallel composition, we can use algebraic reasoning to analyze relevant properties of the

specification that we are interested in.

However, we are unable to define a set of basic terms into which each closed term of χ can be rewritten. One reason for this is the fact that the semantics of the guard operator leads to problems for the proof of the elimination thereom (see Section 8.3).

Whether it is possible to define a set of basic terms into which each closed term of χ can be rewritten, and to develop algorithms for the linearization of hybrid χ processes, are topics for future research.

8.1 The semantics of communication process term

In χ, the parallel composition allows the synchronization of matching send and receive actions. The result of the synchronization is a communication action. The syntax of the communication is given in terms of other language elements (send process term, receive process term, the parallel composition, and the action encapsulation operator).

With the goal of eliminating parallel composition, we need an atomic process term which represents communication (i.e. without such a communication process term, it is not possible to prove the elimination theorem for the parallel composition).

Let $\mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)$ denote a communication process term, where h is a channel, $\mathbf{e}_n$ denotes the expressions $e_1, \ldots, e_n$; $\mathbf{x}_n$ denotes the (non-dotted) variables $x_1, \ldots, x_n$ such that $\mathsf{time} \notin \{\mathbf{x}_n\}$. We provide the deduction rules for the communication process term $\mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)$ as follows:

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L),\ \xi'(\mathbf{x}_n) = \xi(\mathbf{e}_n)}{(C, J, L, H, R) \Vdash \langle \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n), \sigma\rangle \xrightarrow{\xi, \mathrm{ca}(h, [\xi(\mathbf{e}_n)]), \xi'} \langle \surd, \xi'_\sigma\rangle}\ 54$$

$$\frac{}{(C, J, L, H, R) \Vdash \langle \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n), \sigma\rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 55$$

Then, the following lemma captures the fact that the communication process term can be expressed by other language elements.

Lemma 8.1.1 *For arbitrary channel h, expression(s) $\mathbf{e}_n$, and variable(s) $\mathbf{x}_n$, we have*

$$\mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n) \leftrightarrows \partial_{\{\mathrm{isa}(h,cs), \mathrm{ira}(h,cs,W) \mid h \in \mathcal{H}, cs \in \Lambda^*, W \subseteq \mathcal{V}\}}(h \,!!\, \mathbf{e}_n \parallel h \,??\, \mathbf{x}_n).$$

PROOF. (Sketch) This follows from Rules 28, 32, 54, and 55.

8.2 Sub-language of χ

The set of process terms S is defined by the following grammar for the process terms $s \in S$ and $p_{\mathrm{bs}} \in P_{\mathrm{bs}}$:

$$p_{\mathrm{bs}} ::= W : r \gg l_{\mathrm{a}} \mid h \,!!\, \mathbf{e}_n \mid h \,??\, \mathbf{x}_n \mid \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n) \mid \delta,$$

$$s ::= \quad b \to p_{\mathrm{bs}} \quad | \quad u \quad | \quad [s] \quad | \quad s;\, s \quad | \quad s \,[\!]\, s \quad | \quad s \parallel s \quad | \quad \partial_A(s),$$

where b is a guard, W is a set of variables, l_{a} is an action label; r, u are predicates, h is a channel, $\mathbf{e}_n$ denotes the expressions $e_1, \ldots, e_n$; $\mathbf{x}_n$ denotes the (non-dotted) variables $x_1, \ldots, x_n$ such that $\mathsf{time} \notin \{\mathbf{x}_n\}$, and A is a set of actions.

8.3 Elimination

Definition 8.3.1 *(Basic terms) The set of basic terms N is defined by the following grammar for process terms $n \in N$:*

$$n ::= u \mid b \to p_{\mathrm{bs}} \mid [b \to p_{\mathrm{bs}}] \mid b \to p_{\mathrm{bs}};\, n \mid [b \to p_{\mathrm{bs}};\, n] \mid n \,[\!]\, n.$$

Theorem 8.3.1 *(Elimination) For terms from the set S, there is a basic term which is derivably equal (i.e. $\forall_{s:s\in S}\ \exists_{q:q\in N}\ s \leftrightarrows q$).*

PROOF. This follows from Proposition 8.3.2 to Proposition 8.3.4 with induction on the structure of $s \in S$. The proofs for the cases $s \equiv u$ and $s \equiv b \to p^*$, where u is a predicate, b is a guard, and $p^* \in P_{\mathrm{bs}}$, are trivial.

Proposition 8.3.1 *(Any delay operator) Let $p \in N$, then $\exists_{q:q\in N}\ [p] \leftrightarrows q$.*

PROOF. See Appendix E.1.

Proposition 8.3.2 *(Sequential composition operator) Let $p_1, p_2 \in N$, then $\exists_{q:q\in N}\ p_1; p_2 \leftrightarrows q$.*

PROOF. See Appendix E.2.

Proposition 8.3.3 *(Alternative composition operator) Let $p_1, p_2 \in N$, then $\exists_{q:q\in N}\ p_1 \,[\!]\, p_2 \leftrightarrows q$.*

PROOF. It is trivial, because $[\!]$ of two terms in N is by definition also a term in N.

Proposition 8.3.4 *(Parallel composition operator) Let $p_1, p_2 \in N$, then $\exists_{q:q\in N}\ p_1 \parallel p_2 \leftrightarrows q$.*

PROOF. It follows directly from Lemma 8.4.3.

Proposition 8.3.5 *(Action encapsulation operator) Let A be a set of actions and $p \in N$, then $\exists_{q:q\in N}\ \partial_A(p) \leftrightarrows q$.*

PROOF. See Appendix E.3.

We cannot prove the following proposition for the guard operator: For arbitary guard b_c and $p \in P$, then $\exists_{q:q\in P}\ b \rightarrow p \leftrightarrow q$. The reason is that we have no basic terms for $b \rightarrow (b' \rightarrow p)$, where b' is a guard. One might think that the property $b \rightarrow (b' \rightarrow p) \leftrightarrow b \wedge b' \rightarrow p$ holds. However, it does not hold. As a counter-example (to show that this property does not hold), we consider the following process terms $\mathsf{time} \neq 1 \rightarrow \text{false} \rightarrow \text{skip}$ (first process term), and $\mathsf{time} \neq 1 \wedge \text{false} \rightarrow \text{skip}$ (second process term) for the initial valuation $\{\mathsf{time} \mapsto 1\}$. The first process term cannot perform any time transition except for time zero (i.e. zero-time transition), because the first guard (i.e. $\mathsf{time} \neq 1$) is initially false, but it becomes true for the subsequent time-points. We do not have a deduction rule that can be applied for this case. On the other hand, if we only consider time transitions, the conjunction of guards (i.e. $\mathsf{time} \neq 1 \wedge \text{false}$) of the second process term is always false, because the guard false never holds. According to Rule 22, the second process term can perform arbitray time transitions starting from the initial valuation. Hence, the two process terms are not bisimilar.

We have the intuition that the property $b_c \rightarrow (b'_c \rightarrow p) \leftrightarrow b_c \wedge_{\mathrm{l}} b'_c \rightarrow p$ will hold by introducing the concatenation of trajectories/time property to χ semantics, where b_c and b'_c are closed guards (e.g. $x \geq 3$ and $x = 3$ are closed guards; $x > 3$ and $x \neq 3$ are open guards), and $\wedge_{\mathrm{l}}$ denotes the logical AND operator between two closed guards with the left-to-right evaluation. It is defined as follows:

$$b_c \wedge_{\mathrm{l}} b'_c = b_c \text{ for } b_c \in \{\text{false}, \bot\} \text{ and } b_c \wedge_{\mathrm{l}} b'_c = b'_c \text{ if } b_c = \text{true}.$$

Making use of this property, we believe that we can prove that the following proposition: For arbitary closed guard b_c and $p \in P$, then $\exists_{q:q\in P}\ b_c \rightarrow p \leftrightarrow q$. Nevertheless, the proof for such a proposition for the guard operator is considered as future work.

8.4 Additional properties

The following lemmas are introduced for calculation purposes.

Lemma 8.4.1 *For arbitrary guards b_1, b_2, channel h, expression(s) $\mathbf{e}_n$, variable(s) $\mathbf{x}_n$, and p_1, p_2 such that $(p_1 \equiv h\,!!\,\mathbf{e}_n \wedge p_2 \equiv h\,??\,\mathbf{x}_n) \vee (p_1 \equiv h\,??\,\mathbf{x}_n \wedge p_2 \equiv h\,!!\,\mathbf{e}_n)$, we have*

$$b_1 \rightarrow p_1 \parallel b_2 \rightarrow p_2 \leftrightarrow (b_1 \rightarrow p_1;\ b_2 \rightarrow p_2) \,[\!]\, (b_2 \rightarrow p_2;\ b_1 \rightarrow p_1) \,[\!]\, [b_1 \wedge b_2 \rightarrow \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)].$$

One might think that the application of the any delay operator to the process term $b_1 \wedge b_2 \rightarrow \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)$ (let us say process term $p_3 \equiv b_1 \wedge b_2 \rightarrow \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)$) breaks the bisimilarity. However, it does not, because for the case that both guards (i.e. b_1 and b_2) of p_3 evaluate to true, such an application adds arbitrary delay behavior to p_3. If this is the case, it is not possible for $(b_1 \rightarrow p_1;\ b_2 \rightarrow p_2) \,[\!]\, (b_2 \rightarrow p_2;\ b_1 \rightarrow p_1) \,[\!]\, [b_1 \wedge b_2 \rightarrow \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)]$ to perform any time transitions, because both process terms $(b_1 \rightarrow p_1;\ b_2 \rightarrow p_2)$ and $(b_2 \rightarrow p_2;\ b_1 \rightarrow p_1)$

cannot perform any time transitions. Also notice that process term $b \to p^*$, for any guard b and an undelayable process term p^*, can perform arbitrary time transitions only if b evaluates to false.

PROOF. See Appendix E.4.

Lemma 8.4.2 *Let A be a set of actions and $p_* \in P_{\mathrm{bs}}$, then $\exists_{q:q\in P_{\mathrm{bs}}}\ \partial_A(p_*) \leftrightarrows q$.*

PROOF. See Appendix E.5.

For ease of proofs of Proposition 8.3.4, the following lemma (in the form of $s_1 \parallel s_2 \leftrightarrows s_3$) is introduced. Since the following lemma is rather complex, we first briefly introduce some notations used in the lemma.

1. From Proposition 3.5.3, we know that the alternative composition is commutative and associative. We can define a generalized alternative composition operator. For a finite index set I, the notation $[\!]_{i\in I}\ x_i$ represents the alternative composition of the process terms x_i for $i \in I$. If $I = \emptyset$, then $[\!]_{i\in I}\ x_i = \mathrm{true}$. Therefore, every basic term can be written in the form:

$$\begin{aligned} N ::= \ & ([\!]_{i\in I}\ u_i) \mathbin{[\!]} ([\!]_{j\in J}\ b_j \to p_j) \mathbin{[\!]} ([\!]_{k\in K}\ [b_k \to p_k]) \mathbin{[\!]} ([\!]_{l\in L}\ b_l \to p_l;\ n_l) \mathbin{[\!]} \\ & ([\!]_{m\in M}\ [b_m \to p_m;\ n_m]), \end{aligned}$$

   for some finite index sets I, J, K, L, M, where u_i are predicates, b_j, b_k, b_l, b_m are guards, $p_j, p_k, p_l, p_m \in P_{\mathrm{bs}}$ and $n_l, n_m \in N$.

2. For any basic term (let us say) s_1, we can write $s_1 \equiv ([\!]_{i\in I}\ u_i) \mathbin{[\!]} ([\!]_{j\in J}\ b_j \to p_j) \mathbin{[\!]} ([\!]_{k\in K}\ [b_k \to p_k]) \mathbin{[\!]} ([\!]_{l\in L}\ b_l \to p_l;\ n_l) \mathbin{[\!]} ([\!]_{m\in M}\ [b_m \to p_m;\ n_m])$, where $[\!]_{i\in I}\ u_i$ represent delay predicates; $[\!]_{j\in J}\ b_j \to p_j$ represent guarded process terms which can perform termination transitions, communicate with other process terms in a parallel context or perform arbitrary time transitions if the guards b_j are false; $[\!]_{k\in K}\ [b_k \to p_k]$ represent guarded process terms which can perform termination transitions, communicate with other process terms in a parallel context or perform arbitrary time transitions; $[\!]_{l\in L}\ b_l \to p_l;\ n_l$ represent guarded process terms which can perform action transitions, communicate with other process terms in a parallel context or perform arbitrary time transitions if the guards b_l are false; and $[\!]_{m\in M}\ [b_m \to p_m;\ n_m]$ represent guarded process terms which can perform action transitions, communicate with other process terms in a parallel context or perform arbitrary time transitions.

3. Process term s_3 is a basic term which is bisimilar to the parallel composition of s_1 and s_2. It consists of four main parts:

   (a) delay predicates;

   (b) transitions from the left argument of the parallel composition (i.e. s_1);

(c) transitions from the right argument of the parallel composition (i.e. s_2);

(d) communications between the left argument (i.e. s_1) and right argument (i.e. s_2) of the parallel composition.

4. In process term s_3, the notation $\Gamma_{A,B}$ is used. For given index sets A and B, we define $\Gamma_{A,B}$ to be the index set of pairs of indices from A and B of which the corresponding process terms can communicate. That is, $\Gamma_{A,B} = \{(a,b) \in A \times B \mid ((p_a \equiv h_a \,!!\, \mathbf{e}_{n_a} \wedge p_b \equiv h_b \,??\, \mathbf{x}_{n_b}) \vee (p_a \equiv h_a \,??\, \mathbf{x}_{n_a} \wedge p_b \equiv h_b \,!!\, \mathbf{e}_{n_b})) \wedge h_a = h_b\}$ for $A, B \in \{J, K, L, M, J^*, K^*, L^*, M^*\}$, where h_a and h_b are channels, $\mathbf{e}_{n_a}$ and $\mathbf{e}_{n_b}$ are expression(s), and $\mathbf{x}_{n_a}$ and $\mathbf{x}_{n_b}$ are variable(s). For $(a,b) \in \Gamma_{A,B}$, $p_a \parallel p_b \leftrightarrows p_a;\ p_b \,[\!]\, p_b;\ p_a \,[\!]\, \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)$ for some $h, \mathbf{e}_n$, and $\mathbf{x}_n$ (this bisimilarity follows directly from Lemma 8.4.1). Then we define $h_{ab} = h$, $\mathbf{e}_{n_{ab}} = \mathbf{e}_n$ and $\mathbf{x}_{n_{ab}} = \mathbf{x}_n$.

Lemma 8.4.3 *For some finite index sets $I, J, K, L, M, I^*, J^*, K^*, L^*, M^*$, arbitrary predicates u_i, u_{i^*}, arbitrary guards $b_j, b_k, b_l, b_m, b_{j^*}, b_{k^*}, b_{l^*}, b_{m^*}$; p_j, p_k, p_l, p_m, p_{j^*}, $p_{k^*}, p_{l^*}, p_{m^*} \in P_{\mathrm{bs}}$; n_l, n_{l^*}, n_m, $n_{m^*} \in N$, $s_1 \equiv (\,[\!]_{i\in I}\, u_i) \,[\!]\, ([\!]_{j\in J}\, b_j \to p_j) \,[\!]\, ([\!]_{k\in K}\, [b_k \to p_k]) \,[\!]\, ([\!]_{l\in L}\, b_l \to p_l;\ n_l) \,[\!]\, ([\!]_{m\in M}\, [b_m \to p_m;\ n_m])$, $s_2 \equiv ([\!]_{i^*\in I^*}\, u_{i^*}) \,[\!]\, ([\!]_{j^*\in J^*}\, b_{j^*} \to p_{j^*}) \,[\!]\, ([\!]_{k^*\in K^*}\, [b_{k^*} \to p_{k^*}]) \,[\!]\, ([\!]_{l^*\in L^*}\, b_{l^*} \to p_{l^*};\ n_{l^*}) \,[\!]\, ([\!]_{m^*\in M^*}\, [b_{m^*} \to p_{m^*};\ n_{m^*}])$, we have $s_1 \parallel s_2 \leftrightarrows s_3$, where*

s_3 (without comments) $\equiv$

$(\,[\!]_{i\in I}\, u_i) \,[\!]\, ([\!]_{i^*\in I^*}\, u_{i^*})$
$[\!]\ ([\!]_{j\in J}\, b_j \to p_j;\ s_2)$
$[\!]\ ([\!]_{k\in K}\, [b_k \to p_k;\ s_2])$
$[\!]\ ([\!]_{l\in L}\, b_l \to p_l;\ (n_l \parallel s_2))$
$[\!]\ ([\!]_{m\in M}\, [b_m \to p_m];\ (n_m \parallel s_2))$
$[\!]\ ([\!]_{j^*\in J^*}\, b_{j^*} \to p_{j^*};\ s_1)$
$[\!]\ ([\!]_{k^*\in K^*}\, [b_{k^*} \to p_{k^*};\ s_1])$
$[\!]\ ([\!]_{l^*\in L^*}\, b_{l^*} \to p_{l^*};\ (n_{l^*} \parallel s_1))$
$[\!]\ ([\!]_{m^*\in M^*}\, [b_{m^*} \to p_{m^*}];\ (n_{m^*} \parallel s_1))$
$[\!]\ ([\!]_{(j,j^*)\in\Gamma_{J,J^*}}\, [b_j \wedge b_{j^*} \to\ \mathrm{ca}(h_{jj^*}, \mathbf{e}_{n_{jj^*}}, \mathbf{x}_{n_{jj^*}})])$
$[\!]\ ([\!]_{(j,k^*)\in\Gamma_{J,K^*}}\, [b_j \wedge b_{k^*} \to \mathrm{ca}(h_{jk^*}, \mathbf{e}_{n_{jk^*}}, \mathbf{x}_{n_{jk^*}})])$
$[\!]\ ([\!]_{(j,l^*)\in\Gamma_{J,L^*}}\, [b_j \wedge b_{l^*} \to \mathrm{ca}(h_{jl^*}, \mathbf{e}_{n_{jl^*}}, \mathbf{x}_{n_{jl^*}});\ n_{l^*}])$
$[\!]\ ([\!]_{(j,m^*)\in\Gamma_{J,M^*}}\, [b_j \wedge b_{m^*} \to \mathrm{ca}(h_{jm^*}, \mathbf{e}_{n_{jm^*}}, \mathbf{x}_{n_{jm^*}});\ n_{m^*}])$
$[\!]\ ([\!]_{(k,j^*)\in\Gamma_{K,J^*}}\, [b_k \wedge b_{j^*} \to \mathrm{ca}(h_{kj^*}, \mathbf{e}_{n_{kj^*}}, \mathbf{x}_{n_{kj^*}})])$
$[\!]\ ([\!]_{(k,k^*)\in\Gamma_{K,K^*}}\, [b_k \wedge b_{k^*} \to \mathrm{ca}(h_{kk^*}, \mathbf{e}_{n_{kk^*}}, \mathbf{x}_{n_{kk^*}})])$
$[\!]\ ([\!]_{(k,l^*)\in\Gamma_{K,L^*}}\, [b_k \wedge b_{l^*} \to \mathrm{ca}(h_{kl^*}, \mathbf{e}_{n_{kl^*}}, \mathbf{x}_{n_{kl^*}});\ n_{l^*}])$
$[\!]\ ([\!]_{(k,m^*)\in\Gamma_{K,M^*}}\, [b_k \wedge b_{m^*} \to \mathrm{ca}(h_{km^*}, \mathbf{e}_{n_{km^*}}, \mathbf{x}_{n_{km^*}});\ n_{m^*}])$
$[\!]\ ([\!]_{(l,j^*)\in\Gamma_{L,J^*}}\, [b_l \wedge b_{j^*} \to \mathrm{ca}(h_{lj^*}, \mathbf{e}_{n_{lj^*}}, \mathbf{x}_{n_{lj^*}});\ n_l])$
$[\!]\ ([\!]_{(l,k^*)\in\Gamma_{L,K^*}}\, [b_l \wedge b_{k^*} \to \mathrm{ca}(h_{lk^*}, \mathbf{e}_{n_{lk^*}}, \mathbf{x}_{n_{lk^*}});\ n_l])$
$[\!]\ ([\!]_{(l,l^*)\in\Gamma_{L,L^*}}\, [b_l \wedge b_{l^*} \to \mathrm{ca}(h_{ll^*}, \mathbf{e}_{n_{ll^*}}, \mathbf{x}_{n_{ll^*}});\ (n_l \parallel n_{l^*})])$
$[\!]\ ([\!]_{(l,m^*)\in\Gamma_{L,M^*}}\, [b_l \wedge b_{m^*} \to \mathrm{ca}(h_{lm^*}, \mathbf{e}_{n_{lm^*}}, \mathbf{x}_{n_{lm^*}});\ (n_l \parallel n_{m^*})])$
$[\!]\ ([\!]_{(m,j^*)\in\Gamma_{M,J^*}}\, [b_m \wedge b_{j^*} \to \mathrm{ca}(h_{mj^*}, \mathbf{e}_{n_{mj^*}}, \mathbf{x}_{n_{mj^*}});\ n_m])$

$[] ([]_{(m,k^*)\in\Gamma_{M,K^*}} [b_m \wedge b_{k^*} \rightarrow \mathrm{ca}(h_{mk^*}, \mathbf{e}_{n_{mk^*}}, \mathbf{x}_{n_{mk^*}});\ n_m])$
$[] ([]_{(m,l^*)\in\Gamma_{M,L^*}} [b_m \wedge b_{l^*} \rightarrow \mathrm{ca}(h_{ml^*}, \mathbf{e}_{n_{ml^*}},\ \mathbf{x}_{n_{ml^*}});\ (n_m \parallel n_{l^*})])$
$[] ([]_{(m,m^*)\in\Gamma_{M,M^*}} [b_m \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{mm^*}, \mathbf{e}_{n_{mm^*}}, \mathbf{x}_{n_{mm^*}});\ (n_m \parallel n_{m^*})])$
$[] ([]_{(j^*,j)\in\Gamma_{J^*,J}} [b_{j^*} \wedge b_j \rightarrow \mathrm{ca}(h_{j^*j}, \mathbf{e}_{n_{j^*j}}, \mathbf{x}_{n_{j^*j}})])$
$[] ([]_{(j^*,k)\Gamma_{J^*,K}} [b_{j^*} \wedge b_k \rightarrow \mathrm{ca}(h_{j^*k}, \mathbf{e}_{n_{j^*k}}, \mathbf{x}_{n_{j^*k}})])$
$[] ([]_{(j^*,l)\in\Gamma_{J^*,L}} [b_{j^*} \wedge b_l \rightarrow\ \mathrm{ca}(h_{j^*l}, \mathbf{e}_{n_{j^*l}}, \mathbf{x}_{n_{j^*l}});\ n_l])$
$[] ([]_{(j^*,m)\in\Gamma_{J^*,M}} [b_{j^*} \wedge b_m \rightarrow \mathrm{ca}(h_{j^*m}, \mathbf{e}_{n_{j^*m}}, \mathbf{x}_{n_{j^*m}});\ n_m])$
$[] ([]_{(k^*,j)\in\Gamma_{K^*,J}} [b_{k^*} \wedge b_j \rightarrow \mathrm{ca}(h_{k^*j}, \mathbf{e}_{n_{k^*j}}, \mathbf{x}_{n_{k^*j}})])$
$[] ([]_{(k^*,k)\in\Gamma_{K^*,K}} [b_{k^*} \wedge b_k \rightarrow \mathrm{ca}(h_{k^*k}, \mathbf{e}_{n_{k^*k}}, \mathbf{x}_{n_{k^*k}})])$
$[] ([]_{(k^*,l)\in\Gamma_{K^*,L}} [b_{k^*} \wedge\ b_l \rightarrow \mathrm{ca}(h_{k^*l},\ \mathbf{e}_{n_{k^*l}}, \mathbf{x}_{n_{k^*l}});\ n_l])$
$[] ([]_{(k^*,m)\in\Gamma_{K^*,M}} [b_{k^*} \wedge b_m \rightarrow \mathrm{ca}(h_{k^*m}, \mathbf{e}_{n_{k^*m}},\ \mathbf{x}_{n_{k^*m}});\ n_m])$
$[] ([]_{(l^*,j)\in\Gamma_{L^*,J}} [b_{l^*} \wedge b_j \rightarrow \mathrm{ca}(h_{l^*j}, \mathbf{e}_{n_{l^*j}}, \mathbf{x}_{n_{l^*j}});\ n_{l^*}])$
$[] ([]_{(l^*,k)\in\Gamma_{L^*,K}} [b_{l^*} \wedge b_k \rightarrow \mathrm{ca}(h_{l^*k}, \mathbf{e}_{n_{l^*k}}, \mathbf{x}_{n_{l^*k}});\ n_{l^*}])$
$[] ([]_{(l^*,l)\in\Gamma_{L^*,L}} [b_{l^*} \wedge b_l \rightarrow \mathrm{ca}(h_{l^*l}, \mathbf{e}_{n_{l^*l}}, \mathbf{x}_{n_{l^*l}});\ (n_{l^*} \parallel n_l)])$
$[] ([]_{(l^*,m)\in\Gamma_{L^*,M}} [b_{l^*} \wedge b_m \rightarrow \mathrm{ca}(h_{l^*m}, \mathbf{e}_{n_{l^*m}}, \mathbf{x}_{n_{l^*m}});\ (n_{l^*} \parallel n_m)])$
$[] ([]_{(m^*,j)\in\Gamma_{M^*,J}} [b_{m^*} \wedge b_j \rightarrow \mathrm{ca}(h_{m^*j}, \mathbf{e}_{n_{m^*j}}, \mathbf{x}_{n_{m^*j}});\ n_{m^*}])$
$[] ([]_{(m^*,k)\in\Gamma_{M^*,K}} [b_{m^*} \wedge b_k \rightarrow \mathrm{ca}(h_{m^*k}, \mathbf{e}_{n_{m^*k}}, \mathbf{x}_{n_{m^*k}});\ n_{m^*}])$
$[] ([]_{(m^*,l)\in\Gamma_{M^*,L}} [b_{m^*} \wedge b_l \rightarrow \mathrm{ca}(h_{m^*l}, \mathbf{e}_{n_{m^*l}}, \mathbf{x}_{n_{m^*l}});\ (n_{m^*} \parallel n_l)])$
$[] ([]_{(m^*,m)\in\Gamma_{M^*,M}} [b_{m^*} \wedge b_m \rightarrow \mathrm{ca}(h_{m^*m}, \mathbf{e}_{n_{m^*m}}, \mathbf{x}_{n_{m^*m}});\ (n_{m^*} \parallel n_m)])$, *and*
s_3 *(with comments)* $\equiv$

- *part* (a)
  $([]_{i\in I}\ u_i)\ [] ([]_{i^*\in I^*}\ u_{i^*})$
  - *the alternative composition of delay predicates from the set* I *and* I^*
- *part* (b)
  $[] ([]_{j\in J}\ b_j \rightarrow p_j;\ s_2)$
  - *the possibility of a process term from the set* J *which performs a termination transition, and then continues as* s_2

  $[] ([]_{k\in K}\ [b_k \rightarrow p_k;\ s_2])$
  - *the possibility of a process term from the set* K *which performs a termination transition, and then continues as* s_2

  $[] ([]_{l\in L}\ b_l \rightarrow p_l;\ (n_l \parallel s_2))$
  - *the possibility of a process term from the set* L *which performs a termination transition, and then continues as* $n_l \parallel s_2$

  $[] ([]_{m\in M}\ [b_m \rightarrow p_m];\ (n_m \parallel s_2))$

– *the possibility of a process term from the set M which performs a termination transition, and then continues as $n_m \parallel s_2$*

- *part (c)*
  $[\!] ([\!]_{j^* \in J^*} b_{j^*} \rightarrow p_{j^*};\ s_1)$

  – *the possibility of a process term from the set J^* which performs a termination transition, and then continues as s_1*

  $[\!] ([\!]_{k^* \in K^*} [b_{k^*} \rightarrow p_{k^*};\ s_1])$

  – *the possibility of a process term from the set K^* which performs a termination transition, and then continues as s_1*

  $[\!] ([\!]_{l^* \in L^*} b_{l^*} \rightarrow p_{l^*};\ (n_{l^*} \parallel s_1))$

  – *the possibility of a process term from the set L^* which performs a termination transition, and then continues as $n_{l^*} \parallel s_1$*

  $[\!] ([\!]_{m^* \in M^*} [b_{m^*} \rightarrow p_{m^*}];\ (n_{m^*} \parallel s_1))$

  – *the possibility of a process term from the set M^* which performs a termination transition, and then continues as $n_{m^*} \parallel s_1$*

- *part (d)*
  $[\!] ([\!]_{(j,j^*) \in \Gamma_{J,J^*}} [b_j \wedge b_{j^*} \rightarrow \ \mathrm{ca}(h_{jj^*}, \mathbf{e}_{n_{jj^*}}, \mathbf{x}_{n_{jj^*}})])$
  $[\!] ([\!]_{(j,k^*) \in \Gamma_{J,K^*}} [b_j \wedge b_{k^*} \rightarrow \mathrm{ca}(h_{jk^*}, \mathbf{e}_{n_{jk^*}}, \mathbf{x}_{n_{jk^*}})])$
  $[\!] ([\!]_{(j,l^*) \in \Gamma_{J,L^*}} [b_j \wedge b_{l^*} \rightarrow \mathrm{ca}(h_{jl^*}, \mathbf{e}_{n_{jl^*}}, \mathbf{x}_{n_{jl^*}});\ n_{l^*}])$
  $[\!] ([\!]_{(j,m^*) \in \Gamma_{J,M^*}} [b_j \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{jm^*}, \mathbf{e}_{n_{jm^*}}, \mathbf{x}_{n_{jm^*}});\ n_{m^*}])$

  – *the possibilities of process terms from the pairs of sets J and R^* which communicate, where $R^* \in \{J^*, K^*, L^*, M^*\}$*

  $[\!] ([\!]_{(k,j^*) \in \Gamma_{K,J^*}} [b_k \wedge b_{j^*} \rightarrow \mathrm{ca}(h_{kj^*}, \mathbf{e}_{n_{kj^*}}, \mathbf{x}_{n_{kj^*}})])$
  $[\!] ([\!]_{(k,k^*) \in \Gamma_{K,K^*}} [b_k \wedge b_{k^*} \rightarrow \mathrm{ca}(h_{kk^*}, \mathbf{e}_{n_{kk^*}}, \mathbf{x}_{n_{kk^*}})])$
  $[\!] ([\!]_{(k,l^*) \in \Gamma_{K,L^*}} [b_k \wedge b_{l^*} \rightarrow \mathrm{ca}(h_{kl^*}, \mathbf{e}_{n_{kl^*}}, \mathbf{x}_{n_{kl^*}});\ n_{l^*}])$
  $[\!] ([\!]_{(k,m^*) \in \Gamma_{K,M^*}} [b_k \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{km^*}, \mathbf{e}_{n_{km^*}}, \mathbf{x}_{n_{km^*}});\ n_{m^*}])$

  – *the possibilities of process terms from the pairs of sets K and R^* which communicate*

  $[\!] ([\!]_{(l,j^*) \in \Gamma_{L,J^*}} [b_l \wedge b_{j^*} \rightarrow \mathrm{ca}(h_{lj^*}, \mathbf{e}_{n_{lj^*}}, \mathbf{x}_{n_{lj^*}});\ n_l])$
  $[\!] ([\!]_{(l,k^*) \in \Gamma_{L,K^*}} [b_l \wedge b_{k^*} \rightarrow \mathrm{ca}(h_{lk^*}, \mathbf{e}_{n_{lk^*}}, \mathbf{x}_{n_{lk^*}});\ n_l])$
  $[\!] ([\!]_{(l,l^*) \in \Gamma_{L,L^*}} [b_l \wedge b_{l^*} \rightarrow \mathrm{ca}(h_{ll^*}, \mathbf{e}_{n_{ll^*}}, \mathbf{x}_{n_{ll^*}});\ (n_l \parallel n_{l^*})])$
  $[\!] ([\!]_{(l,m^*) \in \Gamma_{L,M^*}} [b_l \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{lm^*}, \mathbf{e}_{n_{lm^*}}, \mathbf{x}_{n_{lm^*}});\ (n_l \parallel n_{m^*})])$

– *the possibilities of process terms from the pairs of sets L and R^* which communicate*

$$
\begin{aligned}
&[\!] ([\!]_{(m,j^*)\in\Gamma_{M,J^*}} [b_m \wedge b_{j^*} \rightarrow \mathrm{ca}(h_{mj^*}, \mathbf{e}_{n_{mj^*}}, \mathbf{x}_{n_{mj^*}});\ n_m])\\
&[\!] ([\!]_{(m,k^*)\in\Gamma_{M,K^*}} [b_m \wedge b_{k^*} \rightarrow \mathrm{ca}(h_{mk^*}, \mathbf{e}_{n_{mk^*}}, \mathbf{x}_{n_{mk^*}});\ n_m])\\
&[\!] ([\!]_{(m,l^*)\in\Gamma_{M,L^*}} [b_m \wedge b_{l^*} \rightarrow \mathrm{ca}(h_{ml^*}, \mathbf{e}_{n_{ml^*}},\ \mathbf{x}_{n_{ml^*}});\ (n_m \parallel n_{l^*})])\\
&[\!] ([\!]_{(m,m^*)\in\Gamma_{M,M^*}} [b_m \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{mm^*}, \mathbf{e}_{n_{mm^*}}, \mathbf{x}_{n_{mm^*}});\ (n_m \parallel n_{m^*})])
\end{aligned}
$$

– *the possibilities of process terms from the pairs of sets M and R^* which communicate*

$$
\begin{aligned}
&[\!] ([\!]_{(j^*,j)\in\Gamma_{J^*,J}} [b_{j^*} \wedge b_j \rightarrow \mathrm{ca}(h_{j^*j}, \mathbf{e}_{n_{j^*j}}, \mathbf{x}_{n_{j^*j}})])\\
&[\!] ([\!]_{(j^*,k)\Gamma_{J^*,K}} [b_{j^*} \wedge b_k \rightarrow \mathrm{ca}(h_{j^*k}, \mathbf{e}_{n_{j^*k}}, \mathbf{x}_{n_{j^*k}})])\\
&[\!] ([\!]_{(j^*,l)\in\Gamma_{J^*,L}} [b_{j^*} \wedge b_l \rightarrow\ \mathrm{ca}(h_{j^*l}, \mathbf{e}_{n_{j^*l}}, \mathbf{x}_{n_{j^*l}});\ n_l])\\
&[\!] ([\!]_{(j^*,m)\in\Gamma_{J^*,M}} [b_{j^*} \wedge b_m \rightarrow \mathrm{ca}(h, \mathbf{e}_{n_{j^*m}}, \mathbf{x}_{n_{j^*m}});\ n_m])
\end{aligned}
$$

– *the possibilities of process terms from the pairs of sets J^* and R which communicate, where $R \in \{J,K,L,M\}$*

$$
\begin{aligned}
&[\!] ([\!]_{(k^*,j)\in\Gamma_{K^*,J}} [b_{k^*} \wedge b_j \rightarrow \mathrm{ca}(h_{k^*j}, \mathbf{e}_{n_{k^*j}}, \mathbf{x}_{n_{k^*j}})])\\
&[\!] ([\!]_{(k^*,k)\in\Gamma_{K^*,K}} [b_{k^*} \wedge b_k \rightarrow \mathrm{ca}(h_{k^*k}, \mathbf{e}_{n_{k^*k}}, \mathbf{x}_{n_{k^*k}})])\\
&[\!] ([\!]_{(k^*,l)\in\Gamma_{K^*,L}} [b_{k^*} \wedge\ b_l \rightarrow \mathrm{ca}(h_{k^*l},\ \mathbf{e}_{n_{k^*l}}, \mathbf{x}_{n_{k^*l}});\ n_l])\\
&[\!] ([\!]_{(k^*,m)\in\Gamma_{K^*,M}} [b_{k^*} \wedge b_m \rightarrow \mathrm{ca}(h_{k^*m}, \mathbf{e}_{n_{k^*m}},\ \mathbf{x}_{n_{k^*m}});\ n_m])
\end{aligned}
$$

– *the possibilities of process terms from the pairs of sets K^* and R which communicate*

$$
\begin{aligned}
&[\!] ([\!]_{(l^*,j)\in\Gamma_{L^*,J}} [b_{l^*} \wedge b_j \rightarrow \mathrm{ca}(h_{l^*j}, \mathbf{e}_{n_{l^*j}}, \mathbf{x}_{n_{l^*j}});\ n_{l^*}])\\
&[\!] ([\!]_{(l^*,k)\in\Gamma_{L^*,K}} [b_{l^*} \wedge b_k \rightarrow \mathrm{ca}(h_{l^*k}, \mathbf{e}_{n_{l^*k}}, \mathbf{x}_{n_{l^*k}});\ n_{l^*}])\\
&[\!] ([\!]_{(l^*,l)\in\Gamma_{L^*,L}} [b_{l^*} \wedge b_l \rightarrow \mathrm{ca}(h_{l^*l}, \mathbf{e}_{n_{l^*l}}, \mathbf{x}_{n_{l^*l}});\ (n_{l^*} \parallel n_l)])\\
&[\!] ([\!]_{(l^*,m)\in\Gamma_{L^*,M}} [b_{l^*} \wedge b_m \rightarrow \mathrm{ca}(h_{l^*m}, \mathbf{e}_{n_{l^*m}}, \mathbf{x}_{n_{l^*m}});\ (n_{l^*} \parallel n_m)])
\end{aligned}
$$

– *the possibilities of process terms from the pairs of sets L^* and R which communicate*

$$
\begin{aligned}
&[\!] ([\!]_{(m^*,j)\in\Gamma_{M^*,J}} [b_{m^*} \wedge b_j \rightarrow \mathrm{ca}(h_{m^*j}, \mathbf{e}_{n_{m^*j}}, \mathbf{x}_{n_{m^*j}});\ n_{m^*}])\\
&[\!] ([\!]_{(m^*,k)\in\Gamma_{M^*,K}} [b_{m^*} \wedge b_k \rightarrow \mathrm{ca}(h_{m^*k}, \mathbf{e}_{n_{m^*k}}, \mathbf{x}_{n_{m^*k}});\ n_{m^*}])\\
&[\!] ([\!]_{(m^*,l)\in\Gamma_{M^*,L}} [b_{m^*} \wedge b_l \rightarrow \mathrm{ca}(h_{m^*l}, \mathbf{e}_{n_{m^*l}}, \mathbf{x}_{n_{m^*l}});\ (n_{m^*} \parallel n_l)])\\
&[\!] ([\!]_{(m^*,m)\in\Gamma_{M^*,M}} [b_{m^*} \wedge b_m \rightarrow \mathrm{ca}(h_{m^*m}, \mathbf{e}_{n_{m^*m}}, \mathbf{x}_{n_{m^*m}});\ (n_{m^*} \parallel n_m)]).
\end{aligned}
$$

– *the possibilities of process terms from the pairs of sets M^* and R which communicate*

One might think that the application of the any delay operator to all sub-process terms of s_3 which represent communications (e.g. $[\!]_{(j,j^*)\in\Gamma_{J,J^*}}\ [b_j \wedge b_{j^*} \rightarrow\ \mathrm{ca}(h_{jj^*}, \mathbf{e}_{n_{jj^*}}, \mathbf{x}_{n_{jj^*}})]$) introduces additional behavior to s_3. As in Lemma 8.4.1, it is safe to apply the any delay operator to all sub-process terms of s_3 which represent communications without adding behavior to them. We do this, because it can simplify the proofs for time transitions and consistency remarkably.

PROOF. See Appendix E.6.

8.5 Example

This section is intended to show how to eliminate the parallel composition operators from a χ specification modeling a system GME consisting of a generator G, a machine M and an exit E.

For reasons of brevity, the system is simplified considerably. The generator G sends lots 2 to the machine M via a channel h_1 if the guard b is true. Otherwise, lots 3 are sent instead. The machine M receives lots via channel h_1 and immediately sends them to exit via channel h_2.

The process terms modeling G, M and E are as follows:

$$G \equiv b \rightarrow h_1\,!!\,2\ [\!]\ \neg b \rightarrow h_1\,!!\,3$$

$$M \equiv [\mathrm{true} \rightarrow h_1\,??\,x];\ h_2\,!!\,x,$$

$$E \equiv [\mathrm{true} \rightarrow h_2\,??\,x'].$$

The process term modeling the system GME system is as follows:

$$GME \equiv G \parallel M \parallel E.$$

8.5.1 Rewriting of the system GME

Since $G, M, E \in N$, from Proposition 8.3.4, we know that there exists a process term which is bisimilar to $G \parallel M \parallel E$ in which the parallel composition is eliminated.

First, we write $G \parallel M$ as Q (i.e. $Q \equiv (b \rightarrow h_1\,!!\,2\ [\!]\ \neg b \rightarrow h_1\,!!\,3) \parallel [\mathrm{true} \rightarrow h_1\,??\,x];\ h_2\,!!\,x$). Using the χ properties from Section 3.5.4, it is not hard to see that we can obtain

$$\begin{aligned} Q \leftrightarrow\ & b \rightarrow h_1\,!!\,2;\ M \\ & [\!]\ \neg b \rightarrow h_1\,!!\,3;\ M \\ & [\!]\ [\mathrm{true} \rightarrow h_1\,??\,x];\ (h_2\,!!\,x \parallel G) \\ & [\!]\ [b \rightarrow \mathrm{ca}(h_1, 2, x);\ h_2\,!!\,x] \\ & [\!]\ [\neg b \rightarrow \mathrm{ca}(h_1, 3, x);\ h_2\,!!\,x] \end{aligned}$$

with

$$h_2 \,!!\, x \parallel G \leftrightarrows \begin{array}{l} h_2 \,!!\, x;\ G \\ [\!] \ b \to h_1 \,!!\, 2;\ h_2 \,!!\, x \\ [\!] \ \neg b \to h_1 \,!!\, 3;\ h_2 \,!!\, x. \end{array}$$

We write

$$\begin{array}{rl} Q' \equiv & b \to h_1 \,!!\, 2;\ M \\ & [\!] \ \neg b \to h_1 \,!!\, 3;\ M \\ & [\!] \ [\text{true} \to h_1 \,??\, x];\ (h_2 \,!!\, x;\ G \ [\!] \ b \to h_1 \,!!\, 2;\ h_2 \,!!\, x \ [\!] \ \neg b \to h_1 \,!!\, 3;\ h_2 \,!!\, x) \\ & [\!] \ [b \to \text{ca}(h_1, 2, x);\ h_2 \,!!\, x] \\ & [\!] \ [\neg b \to \text{ca}(h_1, 3, x);\ h_2 \,!!\, x], \end{array}$$

where $Q' \leftrightarrows Q$, and process term Q' does not contain any parallel composition operators. Let $R \equiv h_2 \,!!\, x;\ G \ [\!] \ b \to h_1 \,!!\, 2;\ h_2 \,!!\, x \ [\!] \ \neg b \to h_1 \,!!\, 3;\ h_2 \,!!\, x$, then we can have

$$(G \parallel M) \parallel E \leftrightarrows Q' \parallel E \leftrightarrows \begin{array}{l} b \to h_1 \,!!\, 2;\ (M \parallel E) \\ [\!] \ \neg b \to h_1 \,!!\, 3;\ (M \parallel E) \\ [\!] \ [\text{true} \to h_1 \,??\, x];\ (R \parallel E) \\ [\!] \ [b \to \text{ca}(h_1, 2, x)];\ (h_2 \,!!\, x \parallel E) \\ [\!] \ [\neg b \to \text{ca}(h_1, 3, x)];\ (h_2 \,!!\, x \parallel E) \\ [\!] \ E;\ Q' \end{array}$$

with

$$\begin{array}{lcl} M \parallel E & \leftrightarrows & [\text{true} \to h_1 \,??\, x];\ (h_2 \,!!\, x \parallel E) \ [\!] \ E;\ M \\ R \parallel E & \leftrightarrows & h_2 \,!!\, x;\ (G \parallel E) \\ & & [\!] \ b \to h_1 \,!!\, 2;\ (h_2 \,!!\, x \parallel E) \\ & & [\!] \ \neg b \to h_1 \,!!\, 3;\ (h_2 \,!!\, x \parallel E) \\ & & [\!] \ E;\ R \\ & & [\!] \ [\text{true} \to \text{ca}(h_2, x, x');\ G] \\ h_2 \,!!\, x \parallel E & \leftrightarrows & h_2 \,!!\, x;\ E \ [\!] \ E;\ h_2 \,!!\, x \ [\!] \ [\text{true} \to \text{ca}(h_2, x, x')] \end{array}$$

with

$$G \parallel E \leftrightarrows b \to h_1 \,!!\, 2;\ E \ [\!] \ \neg b \to h_1 \,!!\, 3;\ E \ [\!] \ E;\ G.$$

As we have shown above, there exists a process term which is bisimilar to the process term modeling the system GME $(G \parallel M \parallel E)$ in which the parallel composition is eliminated.

CHAPTER

NINE

Related work

The χ formalism is a hybrid process algebra, and is thus related to the other hybrid process algebras: HyPA [CR05], process algebra for hybrid systems $\text{ACP}_{\text{hs}}^{\text{srt}}$ [BM05], the ϕ-Calculus [RS03], the hybrid formalisms based on CSP [Jif94, CJR96], and the process algebra from [Ver95].

The latter three process algebras [Jif94, CJR96, Ver95] differ from χ in that they do not have shared variables. Shared variables are essential for modular specification of continuous and hybrid systems. The two CSP based formalisms also differ from the other process algebras in that they use a denotational semantics instead of an operational semantics. An operational semantics is generally believed to be more intuitive and easier to understand than a denotational semantics [AFV01].

The ϕ-calculus is a hybrid extension of Milner's π-calculus [Mil99]. The hybrid extension allows processes to interact with continuous environments. The semantics of the ϕ-calculus is based on timed transition systems. The ϕ-calculus differs from the other process algebras in that continuous behavior is not defined by means of predicates in process expressions. Instead, continuous behavior is defined by means of an environment.

In ϕ-calculus, an embedded (hybrid) system is a pair consisting of an environment and a process expression. An environment consists of a state, a dynamic system (specified by differential equations), and an invariant predicate. Environmental actions are used to reset the state, change the dynamic system to a new one, and update the invariant predicate. In this way, the ϕ-calculus can deal with dynamically reconfigurable processes. The resulting differential equations are required to be autonomous. This limits the specification of continuous systems, using the ϕ-calculus, to that of ordinary differential equations (ODEs). In the environment, only time transitions can be executed.

The ϕ-calculus has a maximal progress (or urgent) semantics, which means the system can perform a time transition if and only if the process expression of the system cannot perform any environmental action during such a time transition. This differs from the more flexible concept of urgency as defined in χ, where non-delayable actions have priority over delayable actions.

The relation between χ, hybrid automata, HyPA and $\text{ACP}_{\text{hs}}^{\text{srt}}$ is discussed below. When comparing χ to hybrid automata, it should be kept in mind that many different hybrid automaton definitions exist. Some definitions require solutions for the continuous variables to be differentiable functions, e.g. in [Hen00b, AHH96]. Other definitions allow the more

general case of piecewise differentiable or piecewise continuous functions, e.g. in [vdSS00]. In [LSV03], for each variable a dynamic type can be defined, which allows among others solutions in the form of discontinuous functions. Most definitions of hybrid automata do not define urgent transitions, or they define urgent transitions in a restrictive way (non-guarded), as in [HHWT95]. In [NOSY92], urgent transitions are defined in a general way, using a predicate that defines the maximum sojourn time in a location. However, instead of invariants and flow clauses, evolution functions are used in locations. With respect to the meaning of jump clauses, that define the behavior of the variables in action transitions, differences also occur: where in [Hen00b] the variables can in principle perform arbitrary jumps unless restricted by the jump predicate, in [HHWT95], variables in principle remain unchanged unless changes are enforced by the jump predicate. Most hybrid automata distinguish between flow clauses, or vector fields, and invariants. In [HHWT98], however, invariants and flow clauses are combined into one predicate (as in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, HyPA, and χ). Finally, some hybrid automata have a precisely defined syntax, in particular the input languages of the verification tools PHAver [Fre05] and HyTech [HHWT95]. Many other hybrid automata are mainly semantical models, such as the hybrid automata defined in [LSV03] and [LJS+03].

Where HyPA is a conservative extension of ACP from [BW90], and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ is a conservative extension of a combination of the process algebra with continuous relative timing from [BM02] and the process algebra with propositional signals from [BB97], hybrid χ is not an extension of any previously existing process algebra. Hybrid χ has been proven to be an operational conservative extension of timed χ in [vBMR+05]. The semantics of hybrid χ and timed χ, which is derived from hybrid χ, differs considerably from the semantics of their discrete-event predecessor χ_σ as defined in [BK02]. Where χ_σ has non-delayable guards, a weak time-deterministic alternative composition operator, urgent actions only, and no (global) time variable, the semantics of χ as defined in this paper has delayable guards, a strong time-deterministic alternative composition operator, urgent and non-urgent actions, and a global variable denoting the model time.

The integration between the DC and CS world views in χ was inspired by HyPA. Also, the use of delay predicates as atomic process term was inspired by HyPA. The χ formalism and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ were both strongly influenced by hybrid automata. $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, χ, and hybrid automata share the 'consistent equation semantics'. For a hybrid automaton, the invariant of the current location should hold in the current state, and transitions to a new state and new location are allowed only if the invariant of the new location holds in the new state. Correspondingly, in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and χ, the equations (delay predicates) of the process term should be consistent with the current state, and transitions to a new process term are allowed only if the equations (delay predicates) of the new process term are consistent with the new state. The hybrid automaton defined in [ACH+95] has a different semantics in that it allows transitions to a new location only if the invariant of the *current* location holds in the current state and in the new state. The signal emission operator in χ was inspired by the signal emission operator from $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, which in its turn comes from the process algebra with relative timing from [BM02].

Some differences between χ, hybrid automata, $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, and HyPA are:

- Where some hybrid automata and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ use continuous variables that are allowed to jump arbitrarily in an action transition with a true reset predicate, and other hybrid automata and HyPA use continuous variables that are not allowed to jump in an action transition, unless explicitly specified, χ uses both classes of continuous variables. Furthermore, χ adds discrete and algebraic variables. Some hybrid automata (e.g. see [LJS+03]) also define discrete variables (instead of locations). The behavior of the algebraic variables from χ is related to the external variables from the semantical hybrid automata defined in [LSV03]. The external variables are not part of the state, and they can have a dynamic type that allows discontinuous trajectories. However, discrete transitions (action transitions) are defined only on internal variables, and the concept of internal and external variables is linked to visibility and hiding in [LSV03]. In χ, all variables can be used in action predicates, and the different classes of variables and hiding/abstraction are orthogonal concepts.

- Where in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and the hybrid automaton definition from [HHWT98] the dotted variables (derivatives) are part of the state (valuation), in HyPA, other hybrid automata, and χ they are not. The reason for this in χ is that the valuation together with the process term and the environment represent all that is needed to be able to determine future behavior. The values of the dotted variables are not needed for this purpose. For the same reason, algebraic variables are not part of the valuation in χ. Their values are determined completely by the process term.

- Where HyPA does not specify a solution concept for algebraic differential equations, and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ requires differentiability of the trajectories of the continuous variables, the χ semantics defines a solution concept that is parameterized with the type of trajectories allowed. In this paper, piecewise continuous functions for the trajectories of the algebraic and dotted variables are allowed. The parametrization of the solution concept in χ is related to the dynamic type present in [LSV03]. Of course, since the solution concept of HyPA is a parameter of the semantics, it could use the solution concept defined in χ.

- Where in χ the passage of time cannot make a choice between the operands of alternative composition, in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, the passage of time can enforce such a choice. In HyPA, the passage of time will always make a choice between the operands of the choice operator. This corresponds to the initial behavior of a hybrid automaton: depending on the initial state, a non-deterministic choice can be made for the first location where continuous behavior or discrete behavior may take place. After this first choice, a hybrid automaton cannot change location as a result of time passing, nor can outgoing edges disappear as a result of time passing.

- The syntactic extensions present in χ are unavailable in the other three formalisms, apart from the delay operator, which is also available in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. However, in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, the expression defining the amount of delay cannot contain variables. Furthermore,

the scope operators, and process definition and instantiation process terms for complex system specification are available only in χ, apart from the variable scope operator which is added to HyPA in [vdBRC04].

An interesting question is whether the χ functionality could have been obtained by extending HyPA and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ with the χ scope operators, with the χ urgent communication operator, and with similar syntactic extensions as defined in χ. This approach suffers from fundamental limitations. The most important of these are:

- The algebraic variables present in χ cannot be incorporated in this way, because their functionality is reflected in the operational semantics of several process terms.
- The χ solution concept is quite different from the solution concept in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$.
- The semantics of the guards in χ (delayable) is fundamentally different from the semantics of the guards in HyPA and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ (non-delayable).
- The flexibility of urgency in χ, where non-delayable actions have priority over delayable actions, is obtained by a carefully defined semantics of several operators (alternative composition, parallel composition, guard). It cannot be obtained by means of extensions to $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ or HyPA.
- The consistent equation semantics of χ is fundamentally different from the HyPA semantics, where equations can (temporarily) become inconsistent as a result of actions.

The additional functionality of χ makes axiomatization more difficult, when compared to $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and HyPA. When it comes to tool support, the additional functionality offered by χ probably means additional efforts for implementation. At this moment, it is difficult to further compare the expected efforts required for tool implementations of χ, $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and HyPA.

Other formalisms for hybrid system specification are hybrid Petri nets [DA01, FGM01], and formalisms based on hybrid automata such as Charon [ADE+03] and Masaccio [Hen00a]. There are many differences and similarities between χ and these other formalisms. The main difference, however, between χ and other formalisms, including the process algebras and hybrid automata discussed before, is that we consider χ to be overall better suited to modeling. This may mean that certain phenomena can be modeled in χ whereas they cannot be modeled in another formalism, or that certain phenomena can be modeled more concisely or more intuitively in χ.

Which systems can be modeled in χ and not in other formalisms, or the other way round, is difficult to establish. It also depends on the notion of equivalence. For example, the equation $y = \mathrm{step}(t-1)$, where y is an algebraic variable, t denotes time and step is a discontinuous function such that $\mathrm{step}(x)$ is 0 for $x < 0$ and 1 for $x \geq 0$, cannot be specified, or does not have the required behavior, in many formalisms. The required behavior can however be approximated by introducing an additional action to model the discontinuity.

As another example, steady state initialization, as in $\dot{x} = 0 \curvearrowright \dot{x} = -x + 1$, cannot be expressed in most formalisms. When the equations are straightforward enough, however, the same effect can be obtained by direct initializations. In this example, by initializing variable x to 1.

The following properties make χ highly suited to modeling:

1. The integration between the DC and CS world views as explained in Section 1. In this respect χ differs from the other formalisms mentioned above, apart from HyPA and the hybrid automata such as defined in [vdSS00].

2. The combination of concise and intuitive language primitives, well suited to modeling, with a straightforward semantics, well suited to verification. This was in fact the biggest challenge in the design of χ. After numerous attempts to define the language primitives with associated syntax and semantics, it appeared that either the language was well suited to modeling, but with complex semantics, unsuited to verification; or the semantics was straightforward and elegant, but at the same time the language was cumbersome for modeling. The reason for this apparent contradiction is that the requirements for language primitives for verification and the requirements for language primitives for modeling are not the same.

3. The relatively large number of operators dedicated to the modeling of discrete-event behavior: This makes it easy to abstract from continuous behavior and specify timed discrete-event models, without any continuous variables and without differential (algebraic) equations. In this respect, χ has much in common with the hybrid formalisms based on CSP [Jif94, CJR96], and with $\text{ACP}_{\text{hs}}^{\text{srt}}$.

4. Process instantiation, based on the modeling scope operator: this enables hierarchical composition of processes. It also provides encapsulation and data hiding, and it enables re-use of processes: parameterized processes can be defined once and instantiated many times with the same or different parameters. In this respect, the χ formalism is related to Charon and Masaccio, which allow components to be defined and instantiated. The χ formalism, being a process algebra, does not only allow parallel composition (as Charon and Masaccio) and sequential composition (as Masaccio), but allows in principle any combination of process terms by means of the χ operators.

   Local variables, variable and/or action abstraction are present also in other formalisms. Hybrid I/O automata [LSV03] define both action abstraction and variable abstraction, which are referred to as hiding of external actions and external variables. Hybrid (I/O) automata, however, need to be 'compatible' to allow parallel composition. Hybrid I/O automata, for example, require disjointness of the internal variables of the automata in parallel composition.

   In χ, the concepts of variable abstraction and channel abstraction (comparable with action abstraction in other formalisms) are integrated in the modeling scope operator,

which also provides a local scope for variables, channels, and recursion definitions. In this respect, the χ modeling scope operator is a high level modeling primitive unavailable in the other hybrid formalisms. Also, there are no compatibility restrictions on processes for parallel composition. Modular composition of processes is further supported by means of different interaction mechanisms. Processes can interact in three different ways:

- By means of shared variables, which is the main interaction mechanism for continuous-time processes consisting of systems of differential algebraic equations. Interaction between processes in Charon and Masaccio also takes place by means of shared variables. Synchronization by means of actions is, however, not supported.
- By means of channel based 'handshake synchronization'. It is comparable to actions in (hybrid) (I/O) automata and actions in ACP-based process algebras. A difference is that actions can be used to express synchronization between two or more processes. The synchronization mechanism used in χ is CSP [Hoa78] based. A channel can be shared by any number of processes, but synchronization always occurs on a point-to-point basis, so between exactly two processes. Another difference is that the interaction mechanism in χ also allows synchronous *communication*, as explained below, whereas actions are strictly used for synchronization.
- By means of synchronous communication, which is the CSP interaction mechanism that combines synchronization with data-transfer, as also used in [Jif94, CJR96].

CHAPTER

TEN

Conclusions and future work

The χ formalism differs considerably from other formalisms. On the one hand, it supports the dynamics and control way of hybrid systems modeling by means of discontinuous functions and/or switched equation systems, possibly leading to discontinuous trajectories. On the other hand, it supports the computer science way of hybrid systems modeling, where actions are used to model discontinuities. With respect to the computer science way of modeling, the χ formalism is heavily influenced by hybrid automata. The two formalisms both have a choice mechanism where, apart from initialization in a hybrid automaton, the passage of time cannot result in choices between operands (χ) or choices between locations or outgoing edges (hybrid automata). The χ formalism also shares the consistency concept with many hybrid automata: state changes in χ need to be consistent with delay predicates, which include the invariant and flow clauses of hybrid automata.

The χ formalism combines ease of modeling with a straightforward, formal semantics. Ease of modeling is ensured by means of different classes of variables, such as discrete, non-jumping continuous, jumping continuous and algebraic variables; by means of its delayable guard that ensures that the guard always holds when the first action of the guarded process term occurs; by means of its integration of urgent (non-delayable) and non-urgent (delayable) actions on the one hand, and urgent and non-urgent channels on the other hand; by means of allowing the modeling of differential algebraic equations as a process term as in mathematics; by means of allowing straightforward steady-state initialization; and by means of several user-friendly syntactic extensions.

The χ formalism is suited to modeling, simulation and verification of: (timed) discrete-event systems without (differential) equations, continuous-time systems consisting of ordinary differential equations with algebraic constraints, and combined discrete-event/continuous-time systems. It is especially suited to the specification and analysis of complex systems. This is achieved by means of the process terms for scoping, that integrate abstraction, local variables, local channels and local recursion definitions; by means of the process definition and instantiation syntactic extensions that enable process re-use, encapsulation, hierarchical and/or modular composition of processes; and by means of the different interaction mechanisms, namely handshake synchronization and synchronous communication that are mainly intended for discrete-event processes that do not share variables, and shared variables that are mainly intended for interaction between continuous-time or hybrid processes.

In literature, many formal techniques for reasoning about the correctness of hybrid

systems have been proposed. The goal of these formal techniques is to prove that the hybrid system performs as expected. One of the most successful formalisms for hybrid system verification is the theory of hybrid automata. Since the χ formalism is closely related to theory of hybrid automata, formal translations between them (in both directions) have been defined. The translation from hybrid automata to χ aims to show that the χ formalism is at least as expressive as the theory of hybrid automata. The translation from a reasonable subset of χ to hybrid automata enables verification of χ specifications using existing hybrid automata based verification tools. This translation has also been automated.

As an alternative to analyse χ specifications using hybrid automata based verification tools, χ simulators can be used to simulate χ specifications. Recently, a symbolic simulator has been developed for χ.

Like in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and HyPA, a set of basic terms (in χ) has been defined into which many closed terms can be rewritten using χ properties. This is so-called elimination, which is a useful step for algebraic analysis, because it reduces the complexity of specifications (without recursion variables) by transforming them into simpler forms. The elimination result allows to eliminate the parallel composition from many χ specifications, and it can be regarded as a preprocessing step for the linearization (transformation of a recursive specification into linear form) of χ processes.

BIBLIOGRAPHY

[ABDM00] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In Nancy A. Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control, Third International Workshop*, Lecture Notes in Computer Science 1790, pages 20–31. Springer-Verlag, 2000.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[ADE+03] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. J. Pappas, and O. Sokolsky. Hierarchical modeling and analysis of embedded systems. *Proceedings of the IEEE*, 91(1):11–28, 2003.

[AFV01] Luca Aceto, Willem Jan Fokkink, and Chris Verhoef. Structural operational semantics. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 3, pages 197–292. Elsevier, 2001.

[AHH96] R. Alur, T. A. Henzinger, and P. H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[BB97] J. C. M. Baeten and J. A. Bergstra. Process algebra with propositional signals. *Theoretical Computer Science*, 177(2):381–405, 1997.

[BH04] F. Breitenecker and I. Husinsky, editors. *Simulation News Europe*, chapter ARGESIM Comparisons. Number 0-40. EUROSIM, 1990-2004.

[BK00] V. Bos and J. J. T. Kleijn. Automatic verification of a manufacturing system. *Robotics and Computer Integrated Manufacturing*, 17(3):185–198, 2000.

[BK02] V. Bos and J. J. T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. PhD thesis, Eindhoven University of Technology, 2002.

[BM02] J. C. M. Baeten and C. A. Middelburg. *Process Algebra with Timing*. EACTS Monographs in Theoretical Computer Science. Springer-Verlag, 2002.

[BM05] J. A. Bergstra and C. A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335(2/3):215–280, 2005.

[BV95] J. C. M. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4 (Semantic Modelling), pages 149–268. Oxford University Press, 1995.

[BvBR06] J. C. M. Baeten, D. A. van Beek, and J. E. Rooda. *Handbook of Dynamic System Modeling*, chapter Process Algebra. CRC Press LLC, 2006. invited chapter, in process.

[BW90] J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, United Kingdom, 1990.

[CJR96] Zhou Chaochen, Wang Ji, and Anders P. Ravn. A formal description of hybrid systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III - Verification and Control*, Lecture Notes in Computer Science 1066, pages 511–530. Springer-Verlag, 1996.

[CR05] P. J. L. Cuijpers and M. A. Reniers. Hybrid process algebra. *Journal of Logic and Algebraic Programming*, 62(2):191–245, 2005.

[CRH02] P. J. L. Cuijpers, M. A. Reniers, and W. P. M. H. Heemels. Hybrid transition systems. Technical Report CS-Report 02-12, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2002.

[DA01] R. David and H. Alla. On hybrid Petri nets. *Discrete Event Dynamic Systems: Theory & Applications*, 11(1-2):9–40, 2001.

[EN00] R. Gansner Emden and Stephan C. North. An open graph visualization system and its applications to software engineering. *Software–Pratice and Experience*, 30(11):1203–1233, 2000.

[Fáb99] G. Fábián. *A Language and Simulator for Hybrid Systems*. PhD thesis, Eindhoven University of Technology, 1999.

[FGM01] A. Di Febbraro, A. Giua, and G. Menga, editors. *Special Issue on Hybrid Petri Nets*, volume 11, no. 1 and 2 of *Journal of Discrete Event Dynamic Systems*, 2001.

[FHK04] Goran Frehse, Zhi Han, and Bruce Krogh. Assume-guarantee reasoning for Hybrid I/O-Automata by over-approximation of continuous interaction. In *43rd IEEE Conference on Decision and Control*, pages 479–484, Nassau Bahamas, 2004.

[Fil88] A. F. Filippov. *Differential Equations with Discontinuous Right Hand Sides*. Kluwer Academic Publishers, Dordrecht, 1988.

[Fre04a] G. Frehse. *Compositional verification of hybrid systems using simulation relations*. PhD thesis, Radbout University Nijmegen, 2004.

[Fre04b] G. Frehse. *Language Overview v.0.2.2.1 for PHAVer v.0.2.2*. www.cs.ru.nl/ goranf, 2004.

[Fre05] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control, 8th International Workshop*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer-Verlag, 2005.

[Gnu] Gnuplot website. Gnuplot. http://www.gnuplot.info.

[Hen00a] T. A. Henzinger. Masaccio: A formal model for embedded components. In *First IFIP International Conference on Theoretical Computer Science (TCS)*, Lecture Notes in Computer Science 1872, pages 549–563. Springer-Verlag, 2000.

[Hen00b] T. A. Henzinger. The theory of hybrid automata. In M.K. Inan and R.P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series F: Computer and Systems Science*, pages 265–292. Springer-Verlag, New York, 2000.

[HHWT95] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A user guide to HyTech. In *First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS*, Lecture Notes in Computer Science 1019, pages 41–71. Springer Verlag, 1995.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

[HHWT98] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, 1998.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[HSB01] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[Jif94] He 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, 1994.

[LJS+03] J. Lygeros, K. H. Johansson, S. Simic, J. Zhang, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

[LSV03] Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003.

[Map] MapleSoft. Maple. http://www.maplesoft.com.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil82] George J. Milne. Abstraction and nondeterminism in concurrent systems. In *3rd Int. Conference on Distr. Systems*, pages 358–364, 1982.

[Mil99] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, Cambridge, United Kingdom, 1999.

[MRG05] M. R. Mousavi, M. A. Reniers, and J. F. Groote. Notions of bisimulation and congruence formats for SOS with data. *Information and Computation*, 200(1):107–147, 2005.

[NA98] G. Naumoski and W. Alberts. *A Discrete-Event Simulator for Systems Engineering*. PhD thesis, Eindhoven University of Technology, 1998.

[NOSY92] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Workshop on Theory of Hybrid Systems*, pages 149–178, 1992.

[Par81] D. M. R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5^{th} GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DIAMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pyt05] Python website, 2005. http://www.python.org.

[RS03] W. C. Rounds and H. Song. The ϕ-Calculus: A language for distributed control of reconfigurable embedded systems. In Oded Maler and Amir Pnueli, editors, *Hybrid Systems : Computation and Control, 6th International Workshop*, Lecture Notes in Computer Science 2623, pages 435–449. Springer-Verlag, 2003.

[SvBM+03a] R. R. H. Schiffelers, D. A. van Beek, K. L. Man, M. A. Reniers, and J. E. Rooda. Formal semantics of hybrid Chi. In Kim Guldstrand Larsen and Peter Niebert, editors, *Formal Modeling and Analysis of Timed Systems: First International Workshop, FORMATS 2003*, volume 2791 of *Lecture Notes in Computer Science*, pages 151–165. Springer-Verlag, 2003.

[SvBM+03b] R. R. H. Schiffelers, D. A. van Beek, K. L. Man, M. A. Reniers, and J. E. Rooda. A hybrid language for modeling, simulation and verification. In S. Engell, H. Guéguen, and J. Zaytoon, editors, *IFAC Conference on Analysis and Design of Hybrid Systems*, pages 235–240, Saint-Malo, Brittany, France, 2003.

[Use02] Yaroslav S. Usenko. *Linearization in μCRL*. PhD thesis, Eindhoven University of Technology, 2002.

[Utk92] V. I. Utkin. *Sliding Modes in Control Optimization*. Springer-Verlag, Berlin, 1992.

[vBGR97] D. A. van Beek, S. H. F. Gordijn, and J. E. Rooda. Integrating continuous-time and discrete-event concepts in modelling and simulation of manufacturing machines. *Simulation Practice and Theory*, 5(7-8):653–669, 1997.

[vBMR+05] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Syntax and semantics of timed Chi. Technical Report CS-Report 05-09, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2005.

[vBPNR04] D. A. van Beek, A. Pogromsky, H. Nijmeijer, and J. E. Rooda. Convex equations and differential inclusions in hybrid systems. In *43rd IEEE Conference on Decision and Control*, pages 1424–1429, Nassau Bahamas, 2004.

[vBR00] D. A. van Beek and J. E. Rooda. Languages and applications in hybrid modelling and simulation: Positioning of Chi. *Control Engineering Practice*, 8(1):81–91, 2000.

[vBvdHR02] D. A. van Beek, A. van den Ham, and J. E. Rooda. Modelling and control of process industry batch production systems. In *15th Triennial World Congress of the International Federation of Automatic Control*, Barcelona, 2002. CD-ROM.

[vdBRC04] P. van de Brand, M. A. Reniers, and P. J. L. Cuijpers. Linearization of hybrid processes. Technical Report CS-Report 04-29, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2004.

[vdSS00] A. J. van der Schaft and J. M. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Springer Lecture Notes in Control and Information Sciences*. Springer, 2000.

[Ver95] J. J. Vereijken. A process algebra for hybrid systems. In Bouajjani and Maler, editors, *The Second European Workshop on Real-Time and Hybrid Systems*, Grenoble, France, 1995.

APPENDIX

A

Proofs of properties of the Chi semantics

Since a deduction rule A may consist of some sub-deduction rules, we use the notation Rule A.i.s to indicate the sub-deduction rule that has been applied in the proofs, where A represents a deduction rule number, i represents an index, and s indicates the left or right result.

Consider the following deduction rule A:

$$\frac{\langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} p'_{11} \\ \vdots \\ p'_{1n} \end{matrix}, \sigma', E\rangle, \langle q,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} q'_{11} \\ \vdots \\ q'_{1n} \end{matrix}, \sigma', E\rangle}{\langle l,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} l'_{11} \\ \vdots \\ l'_{1n} \end{matrix}, \sigma', E\rangle, \langle r,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} r'_{11} \\ \vdots \\ r'_{1n} \end{matrix}, \sigma', E\rangle} \text{(A)}$$

Rule A.1.l refers to the following sub-deduction rule of deduction rule A:

$$\frac{\langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle p'_{11},\sigma',E\rangle, \langle q,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle q'_{11},\sigma',E\rangle}{\langle l,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle l'_{11},\sigma',E\rangle}$$

Similarly, Rule A.n.r refers to the following sub-deduction rule of deduction rule A:

$$\frac{\langle p,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle p'_{1n},\sigma',E\rangle, \langle q,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle q'_{1n},\sigma',E\rangle}{\langle r,\sigma,E\rangle \xrightarrow{\xi,a,\xi'} \langle r'_{1n},\sigma',E\rangle}$$

Note that i and/or s can be omitted in the notation Rule A.i.s when there is no such a sub-deduction rule.

A.1 Proof of Lemma 3.5.1

Appendix A. Proofs of properties of the Chi semantics

Let p and p' be closed process terms, σ, σ' be valuations, ξ, ξ' be extended valuations, E and E' be environments, a be an action, ρ be a trajectory, and $t \in T$. Then

$$\begin{array}{lcl} \langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E'\rangle & \Rightarrow & \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma' \\ & & \wedge E = E', \\ \langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle & \Rightarrow & \mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma' \\ & & \wedge E = E', \\ \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} & \Rightarrow & \xi_\sigma = \sigma, \end{array}$$

where $\langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \overset{\checkmark}{p'}, \sigma', E'\rangle$ for some p'.

PROOF. We prove this lemma by induction on the depth of the proof of the transition in the left-hand-side of the implication and case distinction on the deduction rule applied last in such a proof. The proof for the equality $E = E'$ in the right-hand-side of the implication is trivial, because the equality $E = E'$ holds necessarily according to the result of each χ deduction rule. Therefore, we do not give the proof of this equality for each rule. In what follows, we write E' as E.

Firstly, we give the proofs for $\langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E\rangle \Rightarrow \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 1. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$ and $\sigma' = \xi'_\sigma$, where ξ'_σ is an abbreviation for $\xi' \restriction \mathrm{dom}(\sigma)$. The domain of the extended valuation ξ' is given by $\mathrm{dom}(\sigma) \cup \dot{C} \cup L$, and the domain of $\xi' \restriction \mathrm{dom}(\sigma)$ is $\mathrm{dom}(\xi') \cap \mathrm{dom}(\sigma)$. Since $\mathrm{dom}(\sigma') = \mathrm{dom}(\xi'_\sigma)$, it is not hard to see that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. For $\xi = \sigma \cup \xi^{\dot{C}L}$, we obtain $\xi_\sigma = \sigma$. We also have $\sigma' = \xi'_{\sigma'}$, because $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$.

- Rules 5 and 6 are similar to the previous case.

- Rule 10. Then, $p = [q]$ for some q and $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 13. Then $p \equiv u \curvearrowright q$ for some u and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E\rangle$ and $\xi \models u$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 16. Then $p \equiv q_1 ; q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E\rangle$ and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 17. Then $p \equiv q_1 ;\ q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 20. Then $p \equiv b \to q$ for some b and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\xi \models b$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 25. Then $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 28. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E_a\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E_a\rangle$ and $\langle q_2, \sigma, E_b\rangle \xrightarrow{\xi,b,\xi'} \langle \_, \sigma', E_b\rangle$ for some (unimportant) actions a and b, and some (unimportant) environments E_a and E_b. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 29. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 32. Then $p \equiv \partial_A(q)$ for some A and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$, and $a \notin A$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 35. Then $p \equiv \upsilon_H(q)$ for some $\mathcal{H}$ and q, and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 38. Then $p \equiv X$ for some X, $E = (C, J, L, H, R)$ and $\langle R(X), \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction, we have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 41. Then $E = (C, J, L, H, R)$ and $p \equiv \iota_{J^+}(q)$ for some J^+ and q, and $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 44. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi_x,a,\xi_y} \langle \_, \sigma', E\rangle$ for some ξ_x and ξ_y. Then, we have $E = (C, J, L, H, R), p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid\ q\,]\!]$ for some q, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma''\rangle$ for some $\mathbf{d}, \mathbf{d}', \mathbf{x}', \mathbf{g}', \sigma_{\mathrm{d}'\mathrm{x}'}, \sigma''$, $\sigma' = \sigma''_\sigma; \xi, \xi'$such that $\xi_x = \xi \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi_y = \xi' \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. Note that the syntactical equality of p' is not given, because it is irrelevant for this proof.

  - Firstly, we have to show that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma''_\sigma)$. By induction, we know that $\mathrm{dom}(\sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}) = \mathrm{dom}(\sigma) \cup \mathrm{dom}(\sigma_{\mathrm{d}'\mathrm{x}'}) = \mathrm{dom}(\sigma'')$. On the other hand, $\mathrm{dom}(\sigma''_\sigma) = \mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma) = (\mathrm{dom}(\sigma) \cup \mathrm{dom}(\sigma_{\mathrm{d}'\mathrm{x}'})) \cap \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma)$, i.e. $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma''_\sigma)$.
  - Secondly, we have to show that $\xi_x \upharpoonright \mathrm{dom}(\sigma) = \sigma$. By induction, we know that $\xi \upharpoonright \mathrm{dom}(\sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}) = \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}$, then $\xi \upharpoonright \mathrm{dom}(\sigma) = \sigma$ and $\xi \upharpoonright \mathrm{dom}(\sigma_{\mathrm{d}'\mathrm{x}'}) = \sigma_{\mathrm{d}'\mathrm{x}'}$. On the other hand, $\xi_x \upharpoonright \mathrm{dom}(\sigma) = (\xi \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \upharpoonright \mathrm{dom}(\sigma) = \xi \upharpoonright \mathrm{dom}(\sigma) = \sigma$, i.e. $\xi_x \upharpoonright \mathrm{dom}(\sigma) = \sigma$.

  - Thirdly, we have to show that $\xi_y \upharpoonright \mathrm{dom}(\sigma''_\sigma) = \sigma''_\sigma$. By induction, we know that $\xi \upharpoonright \mathrm{dom}(\sigma'') = \sigma''$. On the other hand, $\xi_y \upharpoonright \mathrm{dom}(\sigma''_\sigma) = (\xi' \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \upharpoonright \mathrm{dom}(\sigma''_\sigma) = (\xi' \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \upharpoonright (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma)) = \xi' \upharpoonright (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma))$. From $\sigma'' = \xi \upharpoonright \mathrm{dom}(\sigma'')$, we obtain $\sigma''_\sigma = \sigma'' \upharpoonright \mathrm{dom}(\sigma) = (\xi \upharpoonright \mathrm{dom}(\sigma'')) \upharpoonright \mathrm{dom}(\sigma)$. It is not hard to see that $\xi' \upharpoonright (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma)) = (\xi \upharpoonright \mathrm{dom}(\sigma'')) \upharpoonright \mathrm{dom}(\sigma)$, which also means $\xi_y \upharpoonright \mathrm{dom}(\sigma''_\sigma) = \sigma''_\sigma$.

- Rules 47, 48 and 51. The proofs are similar. We only give the proof for Rule 47. Then $p \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q \,]\!]$ for some $\mathbf{h}$, q, $E = (C, J, L, H, R)$, $(C,\ J,\ L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle q[\mathbf{h}'/\mathbf{h}], \sigma\rangle \xrightarrow{\xi,b,\xi'} \langle \_, \sigma', E\rangle$ for some unimportant action b for this proof, $\mathbf{h}'$ and $h \in \{\mathbf{h}'\}$ for some h. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

The rules that have not been considered could not have been applied last since they have as conclusion a time transition or a consistency predicate.

Secondly, we give the proofs for $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle \Rightarrow \mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 3. Then, $p \equiv u \equiv p'$ for some u, $E = (C, J, L, H, R)$, $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$, and $\sigma' = \rho_\sigma(t)$. Then, by the definition of Ω_{FG}, $\mathrm{dom}(\rho) = [0, t]$, and $\rho(0) \upharpoonright \mathrm{dom}(\sigma) = \rho_\sigma(0) = \sigma$ necessarily. From $\sigma' = \rho_\sigma(t)$, we know that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. Therefore, we also have $\sigma' = \rho_{\sigma'}(t)$.

- Rule 11. Then $p \equiv [q] \equiv p'$ for some q, $\rho \in \Omega_{\sigma Et}$ and $\sigma' = \rho_\sigma(t)$. Then, by the definition of Ω_{FG}, $\mathrm{dom}(\rho) = [0, t]$, and $\rho(0) \upharpoonright \mathrm{dom}(\sigma) = \rho_\sigma(0) = \sigma$ necessarily. From $\sigma' = \rho_\sigma(t)$, we know that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. Therefore, we have also $\sigma' = \rho_{\sigma'}(t)$.

- Rule 14. Then $p \equiv u \curvearrowright q$ for some u and q, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$ and $\rho(0) \models u$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 18. Then $p \equiv q_1;\ q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q'_1, \sigma', E\rangle$ for some q'_1 and $p' \equiv q'_1;\ q_2$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 21. Then $p \equiv b \to q$ for some b and q, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some q' such that $p' \equiv b \to q'$, and $\forall_{s\in[0,t]}\, \rho(s) \models b$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 22. Then $p \equiv b \to q \equiv p'$ for some b and q, $\rho \in \Omega_{\sigma Et}$ and $\sigma' = \rho_\sigma(t)$ (some irrelevant information for the proof is omitted). By the definition of Ω_{FG}, $\mathrm{dom}(\rho) = [0, t]$, and $\rho(0) \upharpoonright \mathrm{dom}(\sigma) = \rho_\sigma(0) = \sigma$ necessarily. From $\sigma' = \rho_\sigma(t)$, we know that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. Therefore, we have also $\sigma' = \rho_{\sigma'}(t)$.

- Rule 26. Then $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q'_1, \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q'_2, \sigma', E\rangle$ for some q'_1 and q'_2, and $p' \equiv q'_1 \,[\!]\, q'_2$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 30. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E\rangle$, for some q_1' and q_2', and $p' \equiv q_1' \parallel q_2'$. By induction we then have $\mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 33. Then $p \equiv \partial_A(q)$ for some A and q, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some q', and $p' \equiv \partial_A(q')$. By induction we then have $\mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 37. Then $p \equiv \upsilon_H(q)$ for some H and q, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some q', and $p' \equiv \upsilon_H(q')$ (some irrelevant information for this proof is omitted). By induction we then have $\mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 39. Then $p \equiv X$ for some X, $E = (C, J, L, H, R)$ and $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 42. Then $p \equiv \iota_{J^+}(q)$ for some q and set J^+, $E = (C, J, L, H, R)$, $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some q', and $p' \equiv \iota_{J^+}(q')$. By induction we then have $\mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 45. We assume $\langle p, \sigma, E\rangle \overset{t,\rho'}{\longmapsto} \langle p', \sigma', E\rangle$ for some ρ'. Then $E = C, J \cup J^+, L, H, R$, $p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\,]\!]$ for some q, $\sigma_{\mathrm{dx}_\perp}$,$\mathbf{x}$,$\mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma''\rangle$ for some q', $\mathbf{d}$, $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, $\sigma_{\mathrm{d}'\mathrm{x}'}$, σ'', $\sigma' = \sigma''_\sigma$, and $\rho' = \rho_{\sigma \dot{C} L} = \rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. Note that the syntactical equality of p' is not given, because it is irrelevant for this proof.

  - Firstly, we have to show that $\mathrm{dom}(\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) = [0,t]$. By induction we know that $\mathrm{dom}(\rho) = [0,t]$. On the other hand, we have $\mathrm{dom}(\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) = \mathrm{dom}(\rho) = [0,t]$.
  - Secondly, we have to show that $\rho' \downarrow \mathrm{dom}(\sigma)(0) = (\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma)(0) = \sigma$. By induction we know that $\rho \downarrow (\mathrm{dom}(\sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}))(0) = \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}$. Then, we have also $\rho \downarrow \mathrm{dom}(\sigma)(0) = \sigma$ and $\rho \downarrow \mathrm{dom}(\sigma_{\mathrm{d}'\mathrm{x}'})(0) = \sigma_{\mathrm{d}'\mathrm{x}'}$. On the other hand, $\rho' \downarrow \mathrm{dom}(\sigma)(0) = (\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma)(0) = \rho \downarrow \mathrm{dom}(\sigma)(0) = \sigma$.
  - Thirdly, we have to show that $\rho' \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = (\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = \sigma''_\sigma$. By induction we know that $\rho \downarrow \mathrm{dom}(\sigma'')(t) = \sigma''$. Then, we have $(\rho \downarrow \mathrm{dom}(\sigma'')) \downarrow \mathrm{dom}(\sigma)(t) = \sigma'' \downarrow \mathrm{dom}(\sigma) = \sigma''_\sigma$. On the other hand, $\rho' \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = ((\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = ((\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma)))(t) = \rho \downarrow (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma))(t) = (\rho \downarrow \mathrm{dom}(\sigma'')) \downarrow \mathrm{dom}(\sigma)(t) = \sigma''_\sigma$.

- Rules 49 and 52. The proofs are similar. We only give the proof for Rule 49. Then $p \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\,]\!]$ for some $\mathbf{h}$, q, $E = (C, J, L, H, R)$, $(C,\ J,\ L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle q[\mathbf{h}'/\mathbf{h}], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some q'. Note that the syntactical equality of p' is not given, because it is irrelevant for this proof. By induction we then have $\mathrm{dom}(\rho) = [0,t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

The rules that have not been considered could not have been applied last since they conclude an action transition or a consistency predicate.

The proof for $\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \Rightarrow \xi_\sigma = \sigma$ is trivial. According to all χ deduction rules for consistency predicates, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$ necessarily. Then we have $\xi_\sigma = \sigma$.

A.2 Proof of Lemma 3.5.2

Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, ξ and ξ' be extended valuations and a be an action. Then

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \implies \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow},$$

where $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E'\rangle$ for some p', σ', and E'.

PROOF. We prove this lemma by induction on the depth of the proof of $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ using case distinction based on the deduction rule applied last. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 1. Then $p \equiv W : r \gg l_{\mathrm{a}}$ for some W, r, l_{a}, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$, and $a = l_{\mathrm{a}}$. Therefore, by Rule 2, we have $\langle W : r \gg l_{\mathrm{a}}, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 5. Then $p \equiv h \,!!\, \mathbf{e}_n$ for some h and $\mathbf{e}_n$, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$, and $a = \mathrm{isa}(h, [\xi(\mathbf{e}_n)])$. Therefore, by Rule 7, we have $\langle h \,!!\, \mathbf{e}_n, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 6. Then $p \equiv h \,??\, \mathbf{x}_n$ for some h and $\mathbf{x}_n$, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$, and $a = \mathrm{ira}(h, [\mathbf{c}_n], \{\mathbf{x}_n\})$ for some $\mathbf{c}_n$. Then, by Rule 8, we have $\langle h \,??\, \mathbf{x}_n, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 10. Then, $p = [q]$ for some q, $E = (C, J, L, H, R)$ and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$. By induction we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 12, we have $\langle [q], \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, and $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$.

- Rule 13. Then $p \equiv u \curvearrowright q$ for some u and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ and $\xi \models u$. By induction $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 15, we have $\langle u \curvearrowright q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 16. Then $p \equiv q_1 ; q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$ and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 19, we have $\langle q_1 ;\, q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 17. Then $p \equiv q_1;\ q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E'\rangle$ for some q_1'. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 19, we have $\langle q_1;\ q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 20. Then $p \equiv b \to q$ for some b and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ and $\xi \models b$. By induction $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 23, we have $\langle b \to q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 25. Then $p \equiv q_1 \between q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 27, we have $\langle q_1 \between q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 28. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E_a\rangle \xrightarrow{\xi,a,\xi'}$ and $\langle q_2, \sigma, E_b\rangle \xrightarrow{\xi,b,\xi'}$ for some (unimportant) actions a and b, and some (unimportant) environments E_a and E_b. By induction $\langle q_1, \sigma, E_a\rangle \overset{\xi}{\rightsquigarrow}$ and $\langle q_2, \sigma, E_b\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 31 and by Lemma 3.5.6, we have $\langle q_1 \parallel q_2, \sigma, E_a\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 29. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 31, we have $\langle q_1 \between q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 32. Then $p \equiv \partial_A(q)$ for some A and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$, and $a \notin A$. By induction we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34, we obtain $\langle \partial_A(q), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 35. Then $p \equiv \upsilon_H(q)$ for some H and q, and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$. By induction we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 36, we obtain $\langle \upsilon_H(q), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 38. Then $p \equiv X$ for some X and $E = (C, J, L, H, R)$ and $\langle R(X),\sigma,E\ \rangle \xrightarrow{\xi,a,\xi'}$. By induction, we have $\langle R(X), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 40, $\langle X, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 41. Then $E = (C, J, L, H, R)$ and $p \equiv \iota_{J^+}(q)$ for some J^+ and q, and $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'}$. By induction we have $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. By Rule 43, we have $\langle \iota_{J^+}(q), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 44. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi_x,a,\xi_y}$ for some ξ_x, and ξ_y. Then, we have that $E = (C, J, L, H, R)$, $p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\ ]\!]$ for some q, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, and $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \xrightarrow{\xi,a,\xi'}$ for some $\mathbf{d}$, $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, $\sigma_{\mathrm{d}'\mathrm{x}'}$, ξ, ξ' such that $\xi_x = \xi \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi_y = \xi' \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. By induction we have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 46, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p\ ]\!], \sigma\rangle \overset{\xi \upharpoonright (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)}{\rightsquigarrow}$.

- Rules 47, 48 and 51. The proofs are similar. We only give the proof for Rule 47. Then $p \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\ ]\!]$ for some $\mathbf{h}$, q, $E = (C, J, L, H, R)$, $\langle q[\mathbf{h}'/\mathbf{h}], \sigma$, $(C,J,L,H \cup$

$\{\mathbf{h}'\},R)\rangle \xrightarrow{\xi,b,\xi'}$ for some unimportant action b for this proof, $\mathbf{h}'$ and $h \in \{\mathbf{h}'\}$ for some h. By induction we then $\langle q[\mathbf{h}'/\mathbf{h}], \sigma,(C,J,L,H \cup \{\mathbf{h}'\},R)\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 50, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\,]\!], \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

The rules that have not been considered could not have been applied last since they have as conclusion a time transition or a consistency predicate.

A.3 Proof of Lemma 3.5.3

Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, $t \in T$, and ρ be a trajectory. Then,

$$\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle \;\;\Rightarrow\;\; \langle p, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}.$$

PROOF. We prove this lemma by induction on the depth of the proof of $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$ using case distinction based on the deduction rule applied last. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 3. Then, $p \equiv u \equiv p'$ for some u, $E = (C, J, L, H, R)$, $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$. Then, by definition, $\rho(0) \models u$ and $\rho(0) \restriction \mathrm{dom}(\sigma) = \sigma$. Thus $\rho(0) = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Therefore, by Rule 4, we have $\langle u, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 11. Then $p \equiv [q]$ for some q and $\rho(0) \in \Omega_{\sigma Et}$. Then, by definition, $\rho(0) \restriction \mathrm{dom}(\sigma) = \sigma$. Thus $\rho(0) = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Therefore, by Rule 12, $\langle [q], \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 14. Then $p \equiv u \curvearrowright q$ for some u and p, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$ and $\rho(0) \models u$. Therefore, by induction, $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. Then, by Rule 15, $\langle u \curvearrowright q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 18. Then $p \equiv q_1;\, q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E'\rangle$ for some q_1', and $p' \equiv q_1';\, q_2$. By induction we have $\langle q_1, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$, and thus by application of Rule 19 we have $\langle q_1;\, q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 21. Then $p \equiv b \to q$ for some b and q, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E'\rangle$ for some q' such that $p' \equiv b \to q'$, and $\forall_{s\in[0,t]}\, \rho(s) \models b$. By induction we have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. Since we also have $\rho(0) \models b$, we have, by Rule 23, $\langle b \to q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 22. Then $p \equiv b \to q$ for some b and q, $\rho \in \Omega_{\sigma E t}$, $\exists_{s\in[0,t]}\, \rho(s) \models \neg b$, $\rho(0) \models b \implies \langle q, \sigma, E\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q', \sigma'', E''\rangle$ for some q', σ'' and E''. In case $\rho(0) \models \neg b$, we also have $\sigma \cup \xi^{\dot{C}L} \models \neg b$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Then, by Rule 24, $\langle b \to q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. In case $\rho(0) \models b$, we have $\langle q, \sigma, E\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q', \sigma'', E''\rangle$. By induction we then have $\langle q, \sigma, E\rangle \overset{\rho\restriction\{0\}(0)}{\rightsquigarrow}$, which gives $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 23 we then have $\langle b \to q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 26. Then $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E'\rangle$ for some q_1', $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E'\rangle$ for some q_2', and $p' \equiv q_1' \,[\!]\, q_2'$. By induction we have $\langle q_1, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$ and $\langle q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$, and thus by application of Rule 27 we have $\langle q_1 \,[\!]\, q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 30. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E'\rangle$ for some q_1', $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E'\rangle$ for some q_2', and $p' \equiv q_1' \parallel q_2'$. By induction we have $\langle q_1, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$ and $\langle q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$, and thus by application of Rule 31 we have $\langle q_1 \parallel q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 33. Then $p \equiv \partial_A(q)$ for some A and q, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some q', and $p' \equiv \partial_A(q')$. By induction we then have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 34, we obtain $\langle \partial_A(q), \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 37. Then $p \equiv \upsilon_H(q)$ for some H and q, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some q', and $p' \equiv \upsilon_H(q')$ (some irrelevant information for the proof is omitted). By induction we then have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 36, we obtain $\langle \upsilon_H(q), \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 39. Then $p \equiv X$ for some X, $E = (C, J, L, H, R)$ and $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$. As the proof for $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$ has smaller depth, by induction we have $\langle R(X), \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. Then, by Rule 40, we have $\langle X, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$ as well.

- Rule 42. Then $p \equiv \iota_{J^+}(q)$ for some term q and set J^+, $E = (C, J, L, H, R)$, $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma, \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some q', and $p' \equiv \iota_{J^+}(q')$. By induction we then have $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$. From Rule 43, we deduce $\langle \iota_{J^+}(q), \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 45. Then $p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q \,]\!]$ for some q, $E = (C,J,L,H,R)$, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma''\rangle$ for some ρ, q', $\mathbf{d}$, $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, $\sigma_{\mathrm{d}'\mathrm{x}'}$, σ'', $\sigma' = \sigma''_\sigma$, and $\rho' = \rho_{\sigma\dot{C}L} = \rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. Note that the syntactical equality of p' is not given, because it is irrelevant for this proof. By induction we then have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 46, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q \,]\!], \sigma\rangle \overset{\rho\downarrow(\mathrm{dom}(\sigma)\cup\dot{C}\cup L)(0)}{\rightsquigarrow}$.

- Rules 49 and 52. The proofs are similar. We only give the proof for Rule 49. Then $p \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\,]\!]$ for some $\mathbf{h}$, q, $E = (C, J, L, H, R)$, $(C,\ J,\ L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle q[\mathbf{h}'/\mathbf{h}], \sigma\rangle \stackrel{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some q'. Note that the syntactical equality of p' is not given, because it is irrelevant for this proof. By induction we then have $\langle q[\mathbf{h}'/\mathbf{h}], \sigma, (C,$ $J,\ L, H \cup \{\mathbf{h}'\}, R)\ \rangle \stackrel{\rho(0)}{\rightsquigarrow}$. By Rule 50, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\,]\!], \sigma\rangle \stackrel{\rho(0)}{\rightsquigarrow}$.

The rules that have not been considered could not have been applied last since they have as conclusion an action transition or a consistency predicate.

A.4 Proof of Lemma 3.5.4

Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, ξ and ξ' be extended valuations and a be an action. Then

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle \implies \langle p', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}.$$

PROOF. We prove this lemma by induction on the depth of the proof of $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$ using case distinction based on the deduction rule applied last. We do not explicitly separate the base cases and the inductive steps. We know that $E = E'$ (see Lemma 3.5.1), in the proofs, we may write E' as E.

The rule applied last is

- Rule 10.2. Then, $p = [q]$ for some q such that $p' \equiv q'$, and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle$ for some q'. By induction we then have $\langle q', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$.

- Rule 13.2. Then $p \equiv u \curvearrowright q$ for some u and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle$ for some q' such that$p' \equiv q'$ and $\xi \models u$. By induction $\langle q', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$.

- Rule 16. Then $p \equiv q_1\,;\, q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$ and $\langle q_2, \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$. Observe that $p' \equiv q_2$.

- Rule 17. Then $p \equiv q_1\,;\, q_2$ for some q_1 and q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E'\rangle$ for some q_1' such that $p' \equiv q_1'\,;\, q_2$. By induction $\langle q_1', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$. Then, by Rule 19, we have $\langle q_1'\,;\, q_2, \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$.

- Rule 20.2. Then $p \equiv b \to q$ for some b and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle$ for some q' such that $p' \equiv q'$ and $\xi \models b$. By induction $\langle q', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$.

- Rule 25.2.l. Then $p \equiv q_1 \,[\!]\, q_2$ for some q_1, q_2, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E'\rangle$ for some q_1' such that $p' \equiv q_1'$ and $\langle q_2, \sigma, E\rangle \stackrel{\xi}{\rightsquigarrow}$. By induction $\langle q_1', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \stackrel{\xi'}{\rightsquigarrow}$.

- Rule 25.2.r. The proof is similar to the case that Rule 25.2.l has been applied.

- Rule 28 and its sub-deduction rules. Rules 28.1.l and 28.1.r cannot be applied, because Rules 28.1.l and 28.1.r are defined for termination transitions. Since the proofs for the cases that other rules have been applied are similar, we only give the proofs for a (general) case that $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, and we get (from the hypothesis) $\langle q_1, \sigma, E_a\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E_a'\rangle$ and $\langle q_2, \sigma, E_b\rangle \xrightarrow{\xi,b,\xi'} \langle q_2', \sigma', E_b'\rangle$ for some q_1', q_2' such that $p' \equiv q_1' \parallel q_2'$, some (unimportant) actions $a = \mathrm{ira}(h, cs, W)$ and $b = \mathrm{isa}(h, cs)$, and some (unimportant) environments $E_a = (C, J, L, H, R)$, $E_b = (C, J \cup W, L, H, R)$, E_a' and E_b'. By induction $\langle q_1', \sigma', E_a'\rangle \overset{\xi'}{\rightsquigarrow}$ and $\langle q_2', \sigma', E_b'\rangle \overset{\xi'}{\rightsquigarrow}$. Then, by Rule 31 and by Lemma 3.5.6, we have $\langle q_1' \parallel q_2', \sigma', E_a'\rangle \overset{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 29.1.l. Then $p \equiv q_1 \parallel q_2$ for some q_1,q_2, $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$, and $\langle q_2, \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$. Observe that $p' \equiv q_2$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 29.1.r. The proofs are similar to the case that Rule 29.1.l has been applied.

- Rule 29.2.l. Then $p \equiv q_1 \parallel q_2$ for some q_1,q_2, $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E'\rangle$ for some q_1' such that $p' \equiv q_1' \parallel q_2$, and $\langle q_2, \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$. By induction $\langle q_1', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$. Then, by Rule 31, we have $\langle q_1' \parallel q_2, \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 29.2.r. The proof is similar to the case that Rule 29.2.l has been applied.

- Rule 32.2. Then $p \equiv \partial_A(q)$ for some A and q, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle$ for some q' such that $p' \equiv \partial_A(q')$, and $a \notin A$. By induction we then have $\langle q', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 34, we obtain $\langle \partial_A(q'), \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 35.2. Then $p \equiv \upsilon_H(q)$ for some H and q, and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E'\rangle$ for some q' such that $p' \equiv \upsilon_H(q')$. By induction we then have $\langle q', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 36, we obtain $\langle \upsilon_H(q'), \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 38.2. Then $p \equiv X$ for some X and $E = (C, J, L, H, R)$ and $\langle R(X), \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$ for some q' such that $p' \equiv q'$. By induction, we have $\langle q', \sigma', E'\rangle \overset{\xi}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 41.2. Then $E = (C, J, L, H, R)$ and $p \equiv \iota_{J^+}(q)$ for some J^+ and q, and $(C, J \cup J^+, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma'\rangle$ for some q' such that $p' \equiv \iota_{J^+}(q')$. By induction we have $(C, J \cup J^+, L, H, R) \Vdash \langle q', \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. By Rule 43, we have $\langle \iota_{J^+}(q'), \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

- Rule 44.2. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi_x, a, \xi_y} \langle p', \sigma', E'\rangle$ for some ξ_x, and ξ_y. Then, we have that $E = (C, J, L, H, R)$, $p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\,]\!]$ for some q, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, and $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \xrightarrow{\xi, a, \xi'} \langle q', \sigma''\rangle$ for some q',σ'' such that $p' \equiv [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'\})\ [\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid q'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}']\,]\!]$ and $\sigma' = \sigma'' \restriction \mathrm{dom}(\sigma)$, $\mathbf{d}$, $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, $\sigma_{\mathrm{d}'\mathrm{x}'}$, ξ, ξ' such that $\xi_x = \xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi_y = \xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. By induction we have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q', \sigma''\rangle \overset{\xi'}{\rightsquigarrow}$. We can also have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q'[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma' \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{\xi'}{\rightsquigarrow}$ (because variables $\mathbf{d}', \mathbf{x}', \mathbf{g}'$ are fresh and $\sigma'' = \sigma' \cup \sigma_{\mathrm{d}'\mathrm{x}'}$. Using Rule 46, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q'\,]\!], \sigma'\rangle \overset{\xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)}{\rightsquigarrow}$. We can further get $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid q'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}']\,]\!], \sigma'\rangle \overset{\xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.
- Rules 47.2, 48.2 and 51.2. The proofs are similar. We only give the proof for Rule 47.2. Then $p \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\,]\!]$ for some $\mathbf{h}$, q, $E = (C, J, L, H, R)$, $\langle q[\mathbf{h}'/\mathbf{h}], \sigma, (C, J, L, H \cup \{\mathbf{h}'\}, R)\rangle \xrightarrow{\xi, b, \xi'} \langle q', \sigma'\rangle$ for some q' such that $p' \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q'[\mathbf{h}'/\mathbf{h}]\,]\!]$, unimportant action b for this proof, $\mathbf{h}'$ and $h \in \{\mathbf{h}'\}$ for some h. By induction we then have $\langle q', \sigma', (C, J, L, H \cup \{\mathbf{h}'\}, R)\rangle \overset{\xi'}{\rightsquigarrow}$. We can also have $\langle q'[\mathbf{h}'/\mathbf{h}], \sigma', (C, J, L, H \cup \{\mathbf{h}'\}, R)\rangle \overset{\xi'}{\rightsquigarrow}$ (because channels $\mathbf{h}'$ are fresh). Using Rule 50, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q'\,]\!], \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. It is not hard to see that we can have $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q'[\mathbf{h}'/\mathbf{h}]\,]\!], \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\xi'}{\rightsquigarrow}$.

The rules that have not been considered could not have been applied last since they conclude a termination transition, a time transition or a consistency predicate.

A.5 Proof of Lemma 3.5.5

Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, $t \in T$, and ρ be a trajectory. Then,

$$\langle p, \sigma, E\rangle \overset{t, \rho}{\longmapsto} \langle p', \sigma', E'\rangle \;\;\Rightarrow\;\; \langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}.$$

PROOF. We prove this lemma by induction on the depth of the proof of $\langle p, \sigma, E\rangle \overset{t, \rho}{\longmapsto} \langle p', \sigma', E'\rangle$ using case distinction based on the deduction rule applied last. We do not explicitly separate the base cases and the inductive steps. We know that $E = E'$ (see Lemma 3.5.1), in the proofs, we may write E' as E.

The rule applied last is

- Rule 3. Then, $p \equiv u \equiv p'$ for some u, $E = (C, J, L, H, R)$, $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$. Then, by definition, $\rho(t) \models u$ and $\rho(t) \restriction \mathrm{dom}(\sigma') = \sigma'$ (see also Lemma 3.5.1). Thus $\rho(t) =$

$\sigma' \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Therefore, by Rule 4, we have $\langle u, \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 11. Then $p \equiv [q] \equiv p'$ for some q and $\rho(t) \in \Omega_{\sigma E t}$. Then, by definition, $\rho(t) \upharpoonright \mathrm{dom}(\sigma') = \sigma'$ (see also Lemma 3.5.1). Thus $\rho(t) = \sigma' \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Therefore, by Rule 12, $\langle [q], \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 14. Then $p \equiv u \curvearrowright q$ for some u and q, $\langle q, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E' \rangle$ for some q' such that $p' \equiv q'$, and $\rho(0) \models u$. By induction, $\langle q', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 18. Then $p \equiv q_1 ; q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E' \rangle$ for some q_1', and $p' \equiv q_1' ; q_2$. By induction we have $\langle q_1', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, and thus by application of Rule 19 we have $\langle q_1' ; q_2, \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 21. Then $p \equiv b \to q$ for some b and q, $\langle q, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E' \rangle$ for some q' such that $p' \equiv b \to q'$, and $\forall_{s \in [0,t]} \rho(s) \models b$. By induction we have $\langle q', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$. Since we also have $\rho(t) \models b$, we have, by Rule 23, $\langle b \to q', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 22. Then $p \equiv b \to q$ for some b and q, $\rho \in \Omega_{\sigma E t}$, $\exists_{s \in [0,t]} \rho(s) \models \neg b$, $\rho(t) \models b \Rightarrow \langle q, \rho_\sigma(t) \rangle \overset{\rho(t)}{\rightsquigarrow}$. In case $\rho(t) \models \neg b$, we also have $\sigma' \cup \xi^{\dot{C}L} \models \neg b$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Note that $\sigma' = \rho_\sigma(t)$. Then, by Rule 24, $\langle b \to q, \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$. In case $\rho(t) \models b$, we have $\langle q, \rho_\sigma(t) \rangle \overset{\rho(t)}{\rightsquigarrow}$. By Rule 23 we then have $\langle b \to q, \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$. Observe that $p' \equiv b \to q$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 26. Then $p \equiv q_1 \mathbin{[\!]} q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E' \rangle$ for some q_1', $\langle q_2, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E' \rangle$ for some q_2' such that $p' \equiv q_1' \mathbin{[\!]} q_2'$. By induction we have $\langle q_1', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\langle q_2', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, and thus by application of Rule 27 we have $\langle q_1' \mathbin{[\!]} q_2', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 30. Then $p \equiv q_1 \parallel q_2$ for some q_1 and q_2, $\langle q_1, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E' \rangle$ for some q_1', $\langle q_2, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E' \rangle$ for some q_2' such that $p' \equiv q_1' \parallel q_2'$. By induction we have $\langle q_1', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\langle q_2', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, and thus by application of Rule 31 we have $\langle q_1' \parallel q_2', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 33. Then $p \equiv \partial_A(q)$ for some A and q, $\langle q, \sigma, E \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E' \rangle$ for some q', and $p' \equiv \partial_A(q')$. By induction we then have $\langle q', \sigma', E' \rangle \overset{\rho(t)}{\rightsquigarrow}$. By Rule 34, we obtain $\langle \partial_A(q'), \sigma', E \rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 37. Then $p \equiv \upsilon_H(q)$ for some H and q, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E'\rangle$ for some q' such that $p' \equiv \upsilon_H(q')$ (some irrelevant information for the proof is omitted). By induction we then have $\langle q', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$. By Rule 36, we obtain $\langle \upsilon_H(q), \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 39. Then $p \equiv X$ for some X, $E = (C, J, L, H, R)$ and $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E'\rangle$ for some q' such that $p' \equiv q'$. As the proof for $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E'\rangle$ has smaller depth, by induction we have $\langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 42. Then $p \equiv \iota_{J^+}(q)$ for some term q and set J^+, $(C, J \cup J^+, L, H, R) \Vdash \langle q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some q' such that $p' \equiv \iota_{J^+}(q')$, $E = (C, J, L, H, R)$. By induction we then have $(C, J \cup J^+, L, H, R) \Vdash \langle q', \sigma'\rangle \overset{\rho(t)}{\rightsquigarrow}$. From Rule 43, we deduce $\langle \iota_{J^+}(q), \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$.

- Rule 45. We assume $\langle p, \sigma, E\rangle \overset{t,\rho'}{\longmapsto} \langle p', \sigma', E'\rangle$ for some ρ'. Then $p \equiv [\![_{\mathrm{V}}\ \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\ ]\!]$ for some q, $E = (C, J, L, H, R)$, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma''\rangle$ for some ρ, q' such that $p' \equiv [\![_{\mathrm{V}}\, (\sigma' \upharpoonright \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid q'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}']\ ]\!]$, $\mathbf{d}$, $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, $\sigma_{\mathrm{d}'\mathrm{x}'}$, σ'', $\sigma' = \sigma''_\sigma$, and $\rho' = \rho_{\sigma\dot{C}L} = \rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. By induction we then have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q', \sigma''\rangle \overset{\rho(t)}{\rightsquigarrow}$. We can also have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, H, R) \Vdash \langle q'[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma' \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{\xi'}{\rightsquigarrow}$ (because variables $\mathbf{d}', \mathbf{x}', \mathbf{g}'$ are fresh and $\sigma'' = \sigma' \cup \sigma_{\mathrm{d}'\mathrm{x}'}$. Using Rule 46, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q'\ ]\!], \sigma'\rangle \overset{\rho(t)\upharpoonright(\mathrm{dom}(\sigma)\cup\dot{C}\cup L)}{\rightsquigarrow}$. We can further get $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{V}}\, (\sigma' \upharpoonright \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid q'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}']\ ]\!], \sigma'\rangle \overset{\rho(t)\upharpoonright(\mathrm{dom}(\sigma)\cup\dot{C}\cup L)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\rho'(t)}{\rightsquigarrow}$.

- Rules 49. and 52. The proofs are similar. We only give the proof for Rule 49. Then $p \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q\ ]\!]$ for some $\mathbf{h}$, q, $E = (C, J, L, H, R)$, $(C,\ J,\ L, H \cup \{\mathbf{h}'\}, R) \Vdash \langle q[\mathbf{h}'/\mathbf{h}], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some $\mathbf{h}', q'$ such that $p' \equiv [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q'[\mathbf{h}'/\mathbf{h}]\ ]\!]$. By induction we then have $\langle q', \sigma',\ (C,\ J,\ L, H \cup \{\mathbf{h}'\}, R)\ \rangle \overset{\rho(t)}{\rightsquigarrow}$. We also have $\langle q'[\mathbf{h}'/\mathbf{h}], \sigma',\ (C,\ J,\ L, H \cup \{\mathbf{h}'\}, R)\ \rangle \overset{\rho(t)}{\rightsquigarrow}$ (because channels $\mathbf{h}'$ are fresh). By Rule 50, we obtain $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q'\ ]\!], \sigma'\rangle \overset{\rho(t)}{\rightsquigarrow}$. It is not hard to see that we also have $(C, J, L, H, R) \Vdash \langle [\![_{\mathrm{H}}\, \{\mathbf{h}\} \mid q'[\mathbf{h}'/\mathbf{h}]\ ]\!], \sigma'\rangle \overset{\rho(t)}{\rightsquigarrow}$, i.e. $\langle p', \sigma', E'\rangle \overset{\rho(t)}{\rightsquigarrow}$.

The rules that have not been considered could not have been applied last since they conclude a termination transition, an action transition or a consistency predicate.

A.6 Proof of Lemma 3.5.6

Let p be a closed process term, σ be a valuation, C, J, W, L be sets of various classes of χ variables such that J and $W \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, H be a set of channels, R be a recursion definition, and ξ be an extended valuation. Then

$$\langle p, \sigma, (C, J, L, H, R)\rangle \overset{\xi}{\rightsquigarrow} \;\Leftrightarrow\; \langle p, \sigma, (C, J \cup W, L, H, R)\rangle \overset{\xi}{\rightsquigarrow}.$$

PROOF. The proof is trivial. The domain of the extended valuation ξ is given by $\mathrm{dom}(\sigma) \cup \dot{C} \cup L$ for all χ consistency predicate rules. Hence, any variation in the set of jumping variables in the environment of a consistent χ process is irrelevant for the consistency predicate.

A.7 Proof of Theorem 3.5.1

Stateless bisimilarity is a congruence with respect to all χ operators.

PROOF. Besides Rules 22 and 37, it is easy to see that all deduction rules of the χ formalism satisfy the *process-tyft* format containing predicates and negative premises for stratifiable transition system specifications [MRG05] (which we call process-panth format for simplicity). It is worth mentioning that the process-panth format extends the process-tyft format with predicates and negative premises for stratifiable transition system specifications.

Actually, Rule 22 is an abbreviation of the following deduction rules:

$$\frac{C_{\rightarrow}, \rho(s) \models \neg b, \rho(0) \models b, \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma'\rangle, \rho(t) \models b, \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p, \rho_\sigma(t)\rangle} (22.\mathrm{A})$$

$$\frac{C_{\rightarrow}, \rho(0) \models \neg b, \rho(t) \models b, \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p, \rho_\sigma(t)\rangle} (22.\mathrm{B})$$

$$\frac{C_{\rightarrow}, \rho(0) \models b, \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma'\rangle, \rho(t) \models \neg b}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p, \rho_\sigma(t)\rangle} (22.\mathrm{C})$$

$$\frac{C_{\rightarrow}, \rho(0) \models \neg b, \rho(t) \models \neg b}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p, \rho_\sigma(t)\rangle} (22.\mathrm{D})$$

Here, $C_{\rightarrow}$ denotes the following hypothesis: $\rho \in \Omega_{\sigma Et}, \forall_{s\in(0,t)}\, \rho(s) \models \neg b, \exists_{s\in[0,t]}\, \rho(s) \models \neg b$. By inspection of each rule from Rule 22.A to 22.D, we know all these rules satisfy the process-panth format. Therefore, except Rule 37, all deduction rules of the χ formalism, satisfy the process-panth format, and thus stateless bisimilarity is a congruence for all χ operators except for the urgent communication operator (which is defined by Rule 37).

Rule 37 does not satisfy the process-panth format, so we need to give manual proof to show that stateless bisimilarity is a congruence for the urgent communication operator. To show this, we also need the following lemma.

Lemma A.7.1 *For arbitrary closed process terms p and q such that $p \leftrightarrows q$, valuation σ, action label a and environment E, we have*

$$\langle p, \sigma, E\rangle \overset{a}{\not\rightarrow} \;\Leftrightarrow\; \langle q, \sigma, E\rangle \overset{a}{\not\rightarrow},$$

where $\langle p, \sigma, E\rangle \overset{a}{\not\rightarrow}$ denotes $(\nexists_{\xi,\xi',p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle) \wedge (\nexists_{\xi,\xi',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle)$.

PROOF. The proof is straightforward. This can be done by proof by contradiction.

Next, we prove that stateless bisimilarity is a congruence for the urgent communication operator.

For arbitrary closed process terms p and q such that $p \leftrightarrows q$ and set of channels H, we have $\upsilon_H(p) \leftrightarrows \upsilon_H(q)$.

PROOF. Since $p \leftrightarrows q$, there exists a stateless bisimulation relation R_{pq} such that $(p, q) \in R_{pq}$. We now define $R = \{(\upsilon_H(p), \upsilon_H(q)) \mid (p, q) \in R_{pq}\} \cup R_{pq} \cup \{(i_d, i_d) \mid i_d \in P\}$, and show that all pairs of closed process terms $(p, q) \in R$ satisfy the six conditions of Definition 3.5.1 (i.e. R is a stateless bisimulation relation). This relation contains pairs of closed process terms $(p, q) \in R_{pq}$ and pairs of the form (i_d, i_d). Since the proofs are trivial for such pairs these are omitted. Furthermore, the proofs of the left implication of conditions 1 and 6 are similar to the proofs of the right implication of conditions 1 and 6. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle \upsilon_H(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 35.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. Since $p \leftrightarrows q$, we have $E \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. Using Rule 35.1, we obtain $E \Vdash \langle \upsilon_H(q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle \upsilon_H(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means Rule 35.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv \upsilon_H(k_1')$. Since $p \leftrightarrows q$, we have $E \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_2', \sigma'\rangle$ for some k_2' such that $(k_1', k_2') \in R$. Using Rule 35.2, we obtain $E \Vdash \langle \upsilon_H(q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \upsilon_H(k_2'), \sigma'\rangle$ and $k_2 \equiv \upsilon_H(k_2')$. Observe that $(k_1, k_2) \in R$.

Condition 4: We assume $E \Vdash \langle \upsilon_H(p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 37 has been applied necessarily. Then, we have $\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle$ for some p' such that $k_1 \equiv \upsilon_H(p')$, $\langle p, \sigma\rangle \overset{\mathrm{ca}(h,*)}{\twoheadrightarrow}$ for some h, $\forall_{s\in[0,t)}\,(\langle p, \sigma\rangle \overset{s,\rho\restriction[0,s]}{\longmapsto} \langle p_s, \sigma_s\rangle, \langle p_s, \sigma_s\rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle p', \sigma'\rangle, \forall_{h\in\mathcal{H}}\,\langle p_s, \sigma_s, E\rangle \overset{\mathrm{ca}(h,*)}{\twoheadrightarrow})$ for some p_s, σ_s. Since $p \leftrightarrows q$, we have $\forall_{s\in[0,t)}\;(\langle q, \sigma\rangle \overset{s,\rho\restriction[0,s]}{\longmapsto} \langle q_s, \sigma_s\rangle$, $\langle q_s, \sigma_s\rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle q', \sigma'\rangle)$ such that $(p', q') \in R$ and $(p_s, q_s) \in R$. From Lemma A.7.1 and $p \leftrightarrows q$, we obtain $\langle q, \sigma\rangle \overset{\mathrm{ca}(h,*)}{\twoheadrightarrow}$ and $\forall_{s\in[0,t)}(\forall_{h\in\mathcal{H}}\,\langle q_s, \sigma_s, E\rangle \overset{\mathrm{ca}(h,*)}{\twoheadrightarrow})$. Using Rule 37, we get $E \Vdash \langle \upsilon_H(q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \upsilon_H(q'), \sigma'\rangle$. Take $k_2 \equiv \upsilon_H(q')$ and observe that $(k_1, k_2) \in R$.

Condition 6: First, we assume $E \Vdash \langle v_H(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 36 has been applied necessarily. Then, $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Since $p \underline{\leftrightarrow} q$, we have $E \Vdash \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 36, we obtain $E \Vdash \langle v_H(q), \sigma \rangle \overset{\xi}{\rightsquigarrow}$.

Finally, we conclude that stateless bisimilarity is a congruence with respect to all χ operators.

APPENDIX

B

Proofs of properties of the Chi operators

In this appendix, the outline of the proofs for the properties in Section 3.5.4 is given. For all of these properties, the proofs follow the same lines. A relation R is defined containing at least all closed instantiations of the property to be proved. Then, it must be shown that this relation is a stateless bisimulation. For this, for each pair of closed process terms $(p, q) \in R$, it has to be shown that it satisfies the six conditions of Definition 3.5.1. Often, the relation R contains pairs of the form (i_d, i_d). Since the proofs are trivial for such pairs these are omitted. As the deduction rules of χ are such that the environment does not change in a transition, we only consider those cases in the proofs. As a consequence we use the notation $E \Vdash$ as much as possible.

B.1 Properties of any delay operator

The following lemmas prove the properties of Proposition 3.5.1.

Lemma B.1.1 *For arbitrary closed process term p we have*

$$[p] \leftrightarrows [[p]].$$

PROOF. Let $R = \{([p], [[p]]) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similarly straightforward to the proof of the right implication. The proof of condition 3 is similarly straightforward to the proof of condition 2. The proofs of conditions 4 – 6 are trivial, because process terms $[p]$ and $[[p]]$ allow arbitrary time transitions, and thereby do not change. Process terms $[p]$ and $[[p]]$ are consistent with any extended valuation with respect to σ in any environment.
Condition 1: First, we assume $E \Vdash \langle [p], \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma' \rangle$ for some E, σ, ξ, a, ξ', σ'. Using Rule 10.1, we have $E \Vdash \langle [[p]], \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma' \rangle$.
Condition 2: We assume $E \Vdash \langle [p], \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle k_1, \sigma' \rangle$ for some E, σ, ξ, a, ξ', k_1 σ'. Using Rule 10.2, we have $E \Vdash \langle [[p]], \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle k_1, \sigma' \rangle$. Take $k_2 \equiv k_1$ and observe that $(k_1, k_2) \in R$.

Lemma B.1.2 *For arbitrary predicate u we have*

$$[u] \leftrightarrows \text{true}.$$

PROOF. Let $R = \{([u], \text{true}) \mid \text{predicate } u\}$. Since there are no termination and action transition rules defined for u and true predicate, the conditions 1 – 3 hold trivially. The proofs of conditions 4 – 6 are trivial, because process term $[u]$ and predicate true allow arbitrary time transitions, and thereby do not change. Process term $[u]$ and true predicate are consistent with any extended valuation with respect to σ in any environment.

B.2 Properties of signal emission operator

The following lemmas prove the properties of Proposition 3.5.2.

Lemma B.2.1 *For arbitrary closed process term p we have*

$$\text{true} \curvearrowright p \leftrightarrows p.$$

PROOF. Let $R = \{(\text{true} \curvearrowright p, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of conditions 1 – 3 are similar to the proofs of conditions 1 – 3 of Lemma B.4.1 (except the premise $\xi \models b$ is replaced by $\xi \models u$). The proofs of conditions 4 and 5 are similar to the proofs of conditions 2 and 3 (notice that the premise $\xi \models u$ is replaced by $\rho(0) \models u$ in the proofs). The proofs of condition 6 are similar to the proofs of condition 6 of Lemma B.4.1 (except Rule 24 has not been applied, and the premise $\xi \models b$ is replaced by $\xi \models u$ in the proofs).

Lemma B.2.2 *For arbitrary closed process term p we have*

$$\text{false} \curvearrowright p \leftrightarrows \bot.$$

PROOF. The fact that there are no action transition rules, time transition rules and consistency predicate rules defined for $\curvearrowright$ in which the initialization predicate is not satisfied, also indicates that false $\curvearrowright p$ cannot perform any transition. Therefore, the conditions 1 – 6 hold trivially.

Lemma B.2.3 *For arbitrary predicate u we have*

$$u \curvearrowright u \leftrightarrows u.$$

PROOF. Let $R = \{(u \curvearrowright u, u) \mid \text{predicate } u\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The fact that there are no action transition rules defined for u, also indicates that $u \curvearrowright u$ has no action transitions. Therefore, the conditions 1 – 3 hold trivially.
Condition 4: We assume $E \Vdash \langle u \curvearrowright u, \sigma\rangle \xrightarrow{t,\rho} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means that Rule 14 has been applied necessarily. Then, $E \Vdash \langle u, \sigma\rangle \xrightarrow{t,\rho} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u$. Observe that $(k_1, k_1) \in R$.
Condition 5: We assume $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \xrightarrow{t,\rho} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ', which means that Rule 3 has been applied necessarily. Then, $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$,

$\sigma' = \rho_\sigma(t)$ and $k_1 \equiv u$. We know that $\forall_{s\in[0,t]}\, \rho(s) \models u$ (from the definition of the function Ω_{FG}). Hence, we also have $\rho(0) \models u$. Using Rule 14, we obtain $(C, J, L, H, R) \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle u, \rho_\sigma(t)\rangle$ and observe that $(u, u) \in R$.

Condition 6: First, we assume $E \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means that Rule 15 has been applied necessarily. Then, $E \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u$. Second, we assume $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models u$. According to Rule 15, we get $(C, J, L, H, R) \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Lemma B.2.4 *For arbitrary closed process term p and arbitrary predicates u, u' we have*

$$u \curvearrowright (u' \curvearrowright p) \leftrightarrows (u \wedge u') \curvearrowright p.$$

PROOF. Let $R = \{(u \curvearrowright (u' \curvearrowright p), (u \wedge u') \curvearrowright p) \mid p \in P, \text{predicates } u, u'\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

Condition 1: First, we assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 13.1 has been applied necessarily. Then, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models u$. Again, Rule 13.1 has been applied necessarily. Therefore, we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models u'$. From $\xi \models u$ and $\xi \models u'$ we get $\xi \models u \wedge u'$. Using Rule 13.1, we obtain $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 13.1 has been applied necessarily. Thus, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models u \wedge u'$. From $\xi \models u \wedge u'$ we obtain $\xi \models u$ and $\xi \models u'$. Using Rule 13.1, we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Again using Rule 13.1, we obtain $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means Rule 13.2 has been applied necessarily. Thus, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models u$. Again, Rule 13.2 has been applied necessarily. Therefore, we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models u'$. From $\xi \models u$ and $\xi \models u'$, we obtain $\xi \models u \wedge u'$. Using Rule 13.2, we get $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 3: We assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means Rule 13.2 has been applied necessarily. Therefore, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models u \wedge u'$. From $\xi \models u \wedge u'$, we also have $\xi \models u$ and $\xi \models u'$. Using Rule 13.2 we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. Again using Rule 13.2 we obtain $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 4: We assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means that Rule 14 has been applied necessarily. Then, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u$. For $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$, Rule 14 has been applied necessarily.

Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u'$. From $\rho(0) \models u$ and $\rho(0) \models u'$, we obtain $\rho(0) \models u \wedge u'$. Using Rule 14, we get $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle (u \wedge u') \curvearrowright k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 5: We assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means that Rule 14 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u \wedge u'$. From $\rho(0) \models u \wedge u'$, we can also have $\rho(0) \models u$ and $\rho(0) \models u'$. Using Rule 14, we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$. Again, using Rule 14 we get $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 6: First, we assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means that Rule 15 has been applied necessarily. Then, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u$. For $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, Rule 15 has been applied necessarily. Then $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u'$. From $\xi \models u$ and $\xi \models u'$, we can have $\xi \models u \wedge u'$. Using Rule 15, we obtain $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 15 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u \wedge u'$. From $\xi \models u \wedge u'$, we get $\xi \models u$ and $\xi \models u'$. According to Rule 15, we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 15, we get $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

B.3 Properties of alternative composition

The following lemmas prove the properties of Proposition 3.5.3.

Lemma B.3.1 *For closed term p we have*

$$p \,[\!]\, \text{true} \leftrightarrows p.$$

PROOF. Let $R = \{(p \,[\!]\, \text{true}, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similarly straightforward to the proof of the right implication of condition 1. The proofs of conditions 3 and 5 are similarly straightforward to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle p \,[\!]\, \text{true}, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means that Rule 25.1.l has been applied necessarily. Note that Rule 25.1.r cannot be applied, because true predicate cannot perform any termination transition. Then we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle p \,[\!]\, \text{true}, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 25.2.l has been applied necessarily. Note that Rule 25.2.r cannot be applied, because true predicate cannot perform any action transition. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 4: We assume $E \Vdash \langle p \,[\!]\, \text{true}, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 26 has been applied necessarily. Then, we get $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some

k_p such that $k_1 \equiv k_p \,[\!]\, \text{true}$ and $E \Vdash \langle \text{true}, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{true}, \sigma'\rangle$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.

Condition 6: First, we assume $E \Vdash \langle p \,[\!]\, \text{true}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 27 has been applied necessarily. Then, we get $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ. We also know that true predicate is always consistent. Using Rule 27, we obtain $E \Vdash \langle p \,[\!]\, \text{true}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.3.2 (Idempotency of alternative composition) *For closed term p we have*

$$p \,[\!]\, p \leftrightarrow p.$$

PROOF. Let $R = \{(p \,[\!]\, p, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

Condition 1: First, we assume $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. Since the left and right argument of the $[\!]$ are the same, we only give the proofs in which Rule 25.1.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ'. We know that $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Lemma 3.5.2). Using Rule 25.1.l, we obtain $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 25.2.l or Rule 25.2.r has been applied necessarily. Since the left and right argument of the $[\!]$ are the same, we only the proofs in which Rule 25.1.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 3: We assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ'. We also know that $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Lemma 3.5.2). Using Rule 25.2.l, we obtain $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 4: We assume $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 26 has been applied necessarily. Then, we get $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p \,[\!]\, k_p$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.

Condition 5: We assume $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ'. Using Rule 26, we obtain $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1 \,[\!]\, k_1, \sigma'\rangle$. Take $k_2 \equiv k_1 \,[\!]\, k_1$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $E \Vdash \langle p \,[\!]\, p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 27 has been applied necessarily. Then, we get $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ. Using Rule 27, we obtain $E \Vdash \langle p \,[\!]\, \text{true}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.3.3 (Commutativity of alternative composition) *For arbitrary closed process terms p and q we have*

$$p \,[\!]\, q \leftrightarrow q \,[\!]\, p.$$

PROOF. Let $R = \{(p \between q, q \between p) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since the deduction rules for $\between$ are symmetrical w.r.t. the left and right argument, obviously all conditions are met.

Lemma B.3.4 (Associativity of alternative composition) *For closed process terms p, q and r we have*

$$(p \between q) \between r \leftrightarrows p \between (q \between r).$$

PROOF. Let $R = \{((p \between q) \between r, p \between (q \between r)) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similar to the proof of the right implication. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4. The proof of the left implication of condition 6 is similar to the proof of the right implication.
Condition 1: We assume $E \Vdash \langle (p \between q) \between r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. Hence, we distinguish two cases:

1. Rule 25.1.l has been applied. Then, $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, this means that again either Rule 25.1.l or Rule 25.1.r has been applied necessarily. Hence, we can further distinguish two cases:

   (a) Rule 25.1.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \overset{\xi,a,\xi'}{\rightsquigarrow} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we obtain $E \Vdash \langle q \between r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We further get $E \Vdash \langle p \between (q \between r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ using Rule 25.1.l.

   (b) Rule 25.1.r has been applied. Then, $E \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 25.1.l, we obtain $E \Vdash \langle q \between r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Applying Rule 25.1.r, we obtain $E \Vdash \langle p \between (q \between r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

2. Rule 25.1.r has been applied. The proof is similar to the previous case.

Condition 2: We assume $E \Vdash \langle (p \between q) \between r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that either Rule 25.2.l or Rule 25.2.r has been applied necessarily. Hence, we distinguish two cases:

1. Rule 25.2.l has been applied. Then $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, this means that again either Rule 25.2.l or Rule 25.2.r has been applied necessarily. Hence, we again distinguish two cases:

   (a) Rule 25.2.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we obtain $E \Vdash \langle q \between r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We further get $E \Vdash \langle p \between (q \between r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ using Rule 25.2.l and observe that $(k_1, k_1) \in R$.

(b) Rule 25.2.r has been applied. Then $E \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We get $E \Vdash \langle q \between r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ using Rule 25.2.l. Applying Rule 25.2.r, we obtain $E \Vdash \langle p \between (q \between r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

2. Rule 25.2.r has been applied. The proof is similar to the previous case.

Condition 4: We assume $E \Vdash \langle (p \between q) \between r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 26 has been applied necessarily. Then $E \Vdash \langle p \between q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ and $E \Vdash \langle r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_r, \sigma'\rangle$ for some k_{pq} and k_r such that $k_1 \equiv k_{pq} \between k_r$. For $E \Vdash \langle p \between q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, we obtain $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some k_p, k_q such that $k_{pq} \equiv k_p \between k_q$ (using Rule 26). Applying Rule 26, we get $E \Vdash \langle q \between r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q \between k_r, \sigma'\rangle$. Again, due to Rule 26, we can have $E \Vdash \langle p \between (q \between r), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p \between (k_q \between k_r), \sigma'\rangle$. Note that $k_1 \equiv (k_p \between k_q) \between k_r$. Take $k_2 \equiv k_p \between (k_q \between k_r)$ and observe that $(k_1, k_2) \in R$.
Condition 6: We assume $E \Vdash \langle (p \between q) \between r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means Rule 27 has been applied necessarily. Then $E \Vdash \langle p \between q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle p \between q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, we obtain $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 27). Applying Rule 27, we get $E \Vdash \langle q \between r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 27, we can have $E \Vdash \langle p \between (q \between r), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.3.5 *For arbitrary closed process terms p, q we have*

$$[p \between q] \leftrightarrows [p] \between [q].$$

PROOF. Let $R = \{([p \between q], [p] \between [q]) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similarly straightforward to the proof of the right implication of condition 1. The proofs of conditions 4 – 6 are trivial, because process terms $[p \between q]$, $[p]$ and $[q]$ (also $[p] \between [q]$, see Rule 26) allow arbitrary time transitions, and process terms $[p \between q]$, $[p]$ and $[q]$ (also $[p] \between [q]$) are consistent with any extended valuation with respect to σ in any environment.
Condition 1: First, we assume $E \Vdash \langle [p \between q], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 10.1 has been applied necessarily. Then, $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. For $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, we further distinguish two cases:

- Rule 25.1.l has been applied. Then, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 10.1, we can have $E \Vdash \langle [p], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. We know that $E \Vdash \langle [q], \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 12). According to Rule 25.1.l, we have $E \Vdash \langle [p] \between [q], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$.

- Rule 25.1.r has been applied. The proof is similar to the previous case.

Condition 2: We assume $E \Vdash \langle [p \parallel q], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 10.2 has been applied necessarily. Then, $E \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. For $E \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, we further distinguish two cases:

- Rule 25.2.l has been applied. Then, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 10.2, we can have $E \Vdash \langle [p], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. We know that $E \Vdash \langle [q], \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 12). According to Rule 25.2.l, we have $E \Vdash \langle [p] \parallel [q], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. Take $k_2 \equiv k_1$, and observe that $(k_1, k_2) \in R$.
- Rule 25.2.r has been applied. The proof is similar to the previous case.

B.4 Properties of guard operator

The following lemmas prove the properties of Proposition 3.5.4.

Lemma B.4.1 *For arbitrary closed process term p we have*

$$\text{true} \rightarrow p \leftrightarrows p.$$

PROOF. Let $R = \{(\text{true} \rightarrow p, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.
Condition 1: First, we assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means that Rule 20.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$. We also know that $\xi \models \text{true}$, and obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ using Rule 20.1.
Condition 2: We assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 20.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.
Condition 3: We assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ'. We also know that $\xi \models \text{true}$. We obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ using Rule 20.2 and observe that $(k_1, k_1) \in R$.
Condition 4: We assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 21 has been applied necessarily. Notice that Rule Rule 22 cannot be applied, because the premise $\exists_{s\in[0,t]}\, \rho(s) \models \neg\text{true}$ does not hold. Then $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \text{true} \rightarrow k_p$ and $\forall_{s\in[0,t]}\, \rho(s) \models \text{true}$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.
Condition 5: We assume $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ'. We also know that $\forall_{s\in[0,t]}\, \rho(s) \models \text{true}$. We obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{true} \rightarrow k_1, \sigma'\rangle$ using Rule 21. Take $k_2 \equiv \text{true} \rightarrow k_1$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $E \Vdash \langle \text{true} \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 23 has been applied necessarily. Notice that Rule 24 cannot have been applied, because the premise $\sigma \cup \xi^{\dot{C}L} \models \neg\text{true}$ does not hold. Then $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ. We also know $\xi \models \text{true}$. We obtain $E \Vdash \langle \text{true} \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23.

Lemma B.4.2 *For arbitrary closed process term p we have*

$$\text{false} \to p \leftrightarrow \text{true}.$$

PROOF. Let $R = \{(\text{false} \to p, \text{true}) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since there no are action transition rules defined for a guard that evaluates to false in the extended valuation (i.e. $\xi \models \text{false}$), and for the process term true also no action transition rules are defined, the conditions 1 – 3 hold trivially.
Condition 4: We assume $(C, J, L, H, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ' which means Rule 22 has been applied necessarily. Notice that Rule 21 cannot be applied, because the premise $\forall_{s\in[0,t]}\, \rho(s) \models \text{false}$ does not hold. Then $k_1 \equiv \text{false} \to p$, $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}$ and $\forall_{s\in(0,t)}\, \rho(s) \models \neg\text{false}$. For $\rho \in \Omega_{\sigma E t}$, we can have $(C, J, L, H, R) \Vdash \langle \text{true}, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{true}, \rho_\sigma(t)\rangle$ (see also Rule 3). Take $k_2 \equiv \text{true}$ and observe that $(k_1, k_2) \in R$.
Condition 5: We assume $(C, J, L, H, R) \Vdash \langle \text{true}, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ', which means Rule 3 has been applied necessarily. Then $k_1 \equiv \text{true}$, $\sigma' = \rho_\sigma(t)$, and $\rho \in \Omega_{FG}(\sigma, C, L, \text{true}, t)$. We know that $\forall_{s\in(0,t)}\, \rho(s) \models \neg\text{false}$, $\rho(0) \models \text{false} \Rightarrow (C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$ for some p', σ'', and $\rho(t) \models \text{false} \Rightarrow (C, J, L, H, R) \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (since the left-hand sides of the implications are false, these two implications hold trivially). Using Rule 22, we obtain $(C, J, L, H, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{false} \to p, \sigma'\rangle$. Take $k_2 \equiv \text{false} \to p$ and observe that $(k_2, k_1) \in R$.
Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means Rule 24 has been applied necessarily. Notice that Rule 23 cannot have been applied, because $\xi \models \text{false}$ does not hold. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$. We know that $\sigma \cup \xi^{\dot{C}L} \models \text{true}$. Using Rule 4, we obtain $(C, J, L, H, R) \Vdash \langle \text{true}, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Second, we assume $(C, J, L, H, R) \Vdash \langle \text{true}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \text{true}$. We also know that $\sigma \cup \xi^{\dot{C}L} \models \neg\text{false}$. Using Rule 24 we get $(C, J, L, H, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Lemma B.4.3 *For arbitrary guard b we have*

$$b \to \bot \leftrightarrow \neg b.$$

PROOF. Let $R = \{(b \to \bot, \neg b) \mid \text{guard } b\}$.

Since there are no action transition rules defined for $\bot$, also $b \to \bot$ has no action transition rules defined, and there are no action transition rules defined for delay predicates, the conditions 1 – 3 hold trivially.

Condition 4: We assume $(C, J, L, H, R) \Vdash \langle b \to \bot, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ', which means that either Rule 21 or Rule 22 has been applied necessarily. Then we can distinguish two cases:

1. Rule 21 has been applied. Then, $(C, J, L, H, R) \Vdash \langle \bot, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some k_p. This leads to a contradiction, because $\bot$ cannot perform any time transitions. Thus, Rule 21 cannot have been applied.

2. Rule 22 has been applied. Then, $k_1 \equiv b \to \bot$ and $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma Et}$, $\forall_{s\in(0,t)}$ $\rho(s) \models \neg b, \rho(0) \models b \Rightarrow (C, J, L, H, R) \Vdash \langle \bot, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ for some z, σ'' and $\rho(t) \models b \Rightarrow (C, J, L, H, R) \Vdash \langle \bot, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From the facts $\rho(0) \models b \Rightarrow (C, J, L, H, R) \Vdash \langle \bot, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ and $\rho(t) \models b \Rightarrow (C, J, L, H, R) \Vdash \langle \bot, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(0) \models \neg b$ and $\rho(t) \models \neg b$, since the right-hand side of these implications are false (since $\bot$ cannot perform any transition). Thus, we have $\forall_{s\in[0,t]}\, \rho(s) \models \neg b$. From $\rho \in \Omega_{\sigma Et}$ and $\forall_{s\in[0,t]}\, \rho(s) \models \neg b$. It is not hard to see that $\rho \in \Omega_{FG}(\sigma, C, L, \neg b, t)$. Then, we can also obtain the following transition $(C, J, L, H, R) \Vdash \langle \neg b, \sigma\rangle \xmapsto{t,\rho} \langle \neg b, \rho_\sigma(t)\rangle$ (see also Rule 3). Take $k_2 \equiv \neg b$ and observe that $(k_1, k_2) \in R$.

Condition 5: We assume $(C, J, L, H, R) \Vdash \langle \neg b, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ', which means that Rule 3 has been applied necessarily. Then, $k_1 \equiv \neg b$, $\rho \in \Omega_{FG}(\sigma, C, L, \neg b, t)$ and $\sigma' = \rho_\sigma(t)$. From $\rho \in \Omega_{FG}(\sigma, C, L, \neg b, t)$, we know that $\forall_{s\in[0,t]}\, \rho(s) \models \neg b$ and $\bot$ also cannot perform any transition. Then we also have the following premises $\exists_{s\in[0,t]}\, \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow (C, J, L, H, R) \Vdash \langle \bot, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ for some z, σ'', and $\rho(t) \models b \Rightarrow (C, J, L, H, R) \Vdash \langle \bot, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ hold (because the left-hand side of the implications are false). Using Rule 22, we obtain $(C, J, L, H, R) \Vdash \langle b \to \bot, \sigma\rangle \xmapsto{t,\rho} \langle b \to \bot, \rho_\sigma(t)\rangle$. Take $k_2 \equiv b \to \bot$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle b \to \bot, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 24 has been applied necessarily. Notice that Rule 23 cannot be applied, because the premise $(C, J, L, H, R) \Vdash \langle \bot, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ does not hold. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Applying Rule 4, we get $(C, J, L, H, R) \Vdash \langle \neg b, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Second, we assume $(C, J, L, H, R) \Vdash \langle \neg b, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means Rule 4 has been applied necessarily. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24, we obtain $(C, J, L, H, R) \Vdash \langle b \to \bot, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Lemma B.4.4 (Distributivity of guard over alternative composition) *For arbitrary closed process terms p and q and arbitrary guard b we have*

$$b \to (p \mathbin{[\!]} q) \leftrightarrow b \to p \mathbin{[\!]} b \to q.$$

PROOF. Let $R = \{(b \to (p \,[\!]\, q), b \to p \,[\!]\, b \to q) \mid p, q \in P, \text{guard } b\} \cup \{(i_d, i_d) \mid i_d \in P\}$.
Condition 1: First, we assume $E \Vdash \langle b \to (p \,[\!]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means that Rule 20.1 has been applied necessarily. Then, $E \Vdash \langle p \,[\!]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models b$. For $E \Vdash \langle p \,[\!]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, we distinguish two cases:

1. Rule 25.1.l has been applied. Then $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 20.1, we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. We also obtain $E \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23. Applying Rule 25.1.l, we get $E \Vdash \langle b \to p \,[\!]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.
2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

Second, we assume $(C, J, L, H, R) \Vdash \langle b \to p \,[\!]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some C, J, L, H, R, σ, ξ, a, ξ', σ', which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. We distinguish two cases:

1. Rule 25.1.l has been applied. Then $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 20.1, we must have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models b$. For $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means that either Rule 23 or Rule 24 has been applied necessarily. We distinguish two cases:
   (a) Rule 23 has been applied. Then $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 25.1.l, we can have $(C, J, L, H, R) \Vdash \langle p \,[\!]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Due to Rule 20.1, we finally get $(C, J, L, H, R) \Vdash \langle b \to (p \,[\!]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.
   (b) Rule 24 has been applied. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. This leads to a contradiction. Therefore this case cannot occur.
2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

Condition 2: We assume $E \Vdash \langle b \to (p \,[\!]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 20.2 has been applied necessarily. Then, $E \Vdash \langle p \,[\!]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models b$. For $E \Vdash \langle p \,[\!]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, we distinguish two cases:

1. Rule 25.2.l has been applied. Then $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 20.1, we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. We also obtain $E \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23. Applying Rule 25.2.l, we get $E \Vdash \langle b \to p \,[\!]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.
2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

Condition 3: We assume $(C, J, L, H, R) \Vdash \langle b \to p \,[\!]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. We distinguish two cases:

1. Rule 25.2.l has been applied. Then $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 20.2, we must have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models b$. For $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means that either Rule 23 or Rule 24 has been applied necessarily. We distinguish two cases:

   (a) Rule 23 has been applied. We have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 25.2.l, we have $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. Due to Rule 20.2, we finally get $(C, J, L, H, R) \Vdash \langle b \to (p \parallel q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

   (b) Rule 24 has been applied. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. This leads to a contradiction. Therefore this case cannot occur.

2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

Condition 4: We assume $E \Vdash \langle b \to (p \parallel q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that either Rule 21 or Rule 22 has been applied necessarily. Then we can distinguish two cases:

1. Rule 21 has been applied. Then, $E \Vdash \langle p \parallel q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ for some k_{pq} such that $k_1 \equiv b \to k_{pq}$ and $\forall_{s\in[0,t]}\, \rho(s) \models b$. For $E \Vdash \langle p \parallel q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, we get $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some k_p, k_q such that $k_{pq} \equiv k_p \parallel k_q$ (using Rule 26). Applying Rule 21, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p, \sigma'\rangle$ and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_q, \sigma'\rangle$. According to Rule 26, we have $E \Vdash \langle b \to p \parallel b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p \parallel b \to k_q, \sigma'\rangle$. Note that $k_1 \equiv b \to (k_p \parallel k_q)$. Take $k_2 \equiv b \to k_p \parallel b \to k_q$ and observe that $(k_1, k_2) \in R$.

2. Rule 22 has been applied. Then, $k_1 \equiv b \to (p \parallel q)$ and $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}, \exists_{s\in[0,t]}\, \rho(s) \models \neg b, \forall_{s\in(0,t)}\, \rho(s) \models \neg b, \rho(0) \models b \Rightarrow E \Vdash \langle p \parallel q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ for some z, σ'' and $\rho(t) \models b \Rightarrow E \Vdash \langle p \parallel q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p \parallel q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$, we can also have $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p_z, \sigma''\rangle$ for some p_z, and $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma''\rangle$ for some q_z (see also Rule 26). From $\rho(t) \models b \Rightarrow E \Vdash \langle p \parallel q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we also get $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (see also Rule 27). Using Rule 22, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p, \rho_\sigma(t)\rangle$ and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to q, \rho_\sigma(t)\rangle$. According to Rule 26, we obtain $E \Vdash \langle b \to p \parallel b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p \parallel b \to q, \rho_\sigma(t)\rangle$. Take $k_2 \equiv b \to p \parallel b \to q$ and observe that $(k_1, k_2) \in R$.

Condition 5: We assume $E \Vdash \langle b \to p \parallel b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means that Rule 26 has been applied necessarily. Then $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$,

and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some k_p, k_q such that $k_1 \equiv k_p \,[\!]\, k_q$. For $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$, and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$, four cases can be distinguished:

1. Rule 21 has been applied for both. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_p, \sigma'\rangle$, $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_q, \sigma'\rangle$ for some k'_p, k'_q such that $k_p \equiv b \to k'_p$, $k_q \equiv b \to k'_q$, and $\forall_{s\in[0,t]}\, \rho(s) \models b$. Using Rule 26, we obtain $E \Vdash \langle p \,[\!]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_p \,[\!]\, k'_q, \sigma'\rangle$. Applying Rule 21, we get $E \Vdash \langle b \to (p \,[\!]\, q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to (k'_p \,[\!]\, k'_q), \sigma'\rangle$. Note that $k_1 \equiv b \to k'_p \,[\!]\, b \to k'_q$. Take $k_2 \equiv b \to (k'_p \,[\!]\, k'_q)$ and observe that $(k_2, k_1) \in R$.

2. Rule 22 has been applied for both. Then, $k_p \equiv b \to p, k_q \equiv b \to q$ and $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma Et}$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b, \exists_{s\in[0,t]}\, \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p_z, \sigma''\rangle, \rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma'''\rangle$, for some p_z, q_z, σ'', σ''', $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p_z, \sigma''\rangle$, and $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma'''\rangle$, by Lemma 3.5.1 we know that $\sigma'' = \sigma''' = \rho_\sigma(0)$, we get $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[\!]\, q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z', \sigma''\rangle$ for some z' (see also Rule 26). From $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(t) \models b \Rightarrow E \Vdash \langle p \,[\!]\, q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (see also Rule 27). Using Rule 22, we obtain $E \Vdash \langle b \to (p \,[\!]\, q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to (p \,[\!]\, q), \rho_\sigma(t)\rangle$. Notice that $k_1 \equiv b \to p \,[\!]\, b \to q$. Take $k_2 \equiv b \to (p \,[\!]\, q)$ and observe that $(k_2, k_1) \in R$.

3. Rule 21 has been applied for $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$, and Rule 22 has been applied for $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_p, \sigma'\rangle$ for some k'_p such that $k_p \equiv b \to k'_p$, $\sigma' = \rho_\sigma(t)$, $\forall_{s\in[0,t]}\, \rho(s) \models b$, $\rho \in \Omega_{\sigma Et}$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b, \exists_{s\in[0,t]}\, \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma''\rangle$, for some q_z, σ'', $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, $k_q \equiv b \to q$, and $k_1 \equiv b \to k'_p \,[\!]\, b \to q$. From $\forall_{s\in[0,t]}\, \rho(s) \models b$, and $\forall_{s\in(0,t)}\, \rho(s) \models \neg b$, this leads to a contradiction, unless $t = 0$. Hence, $t = 0$. Then we consider only the case in which $t = 0$. From $E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k'_p, \sigma'\rangle$, $\rho(0) \models b$, and $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma''\rangle$, it is not hard to see that we get $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[\!]\, q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k_z, \sigma'''\rangle$ for some k_z, σ'''. We know that $\sigma' = \sigma'' = \sigma''' = \rho_\sigma(0)$ (see also Rule 26 and Lemma 3.5.1). Since $\rho \in \Omega_{\sigma Et}$, we have $\sigma = \rho_\sigma(0)$. Also, from $E \Vdash \langle p, \rho_\sigma(0)\rangle \overset{0,\rho}{\longmapsto} \langle k'_p, \sigma'\rangle$, we have $E \Vdash \langle p, \rho_\sigma(0)\rangle \overset{\rho(0)}{\rightsquigarrow}$ (by Lemma 3.5.3). Using Rule 26, we have $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[\!]\, q, \rho_\sigma(0)\rangle \overset{\rho(0)}{\rightsquigarrow}$. Applying Rule 22, we obtain $E \Vdash \langle b \to (p \,[\!]\, q), \sigma\rangle \overset{0,\rho}{\longmapsto} \langle b \to (p \,[\!]\, q), \rho_\sigma(0)\rangle$. Take $k_2 \equiv b \to (p \,[\!]\, q)$ and observe that $(k_2, k_1) \in R$.

4. Rule 21 has been applied for $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$, and Rule 22 has been applied for $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$. The proof is similar to the previous case.

Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle b \to (p \parallel q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 23 or Rule 24 has been applied necessarily. Then, we distinguish two cases:

1. Rule 23 has been applied. Then $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. Using Rule 27, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 23, we obtain $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 27, we get $(C, J, L, H, R) \Vdash \langle b \to p \parallel b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

2. Rule 24 has been applied. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24, we can have $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Applying Rule 27, we get that $(C, J, L, H, R) \Vdash \langle b \to p \parallel b \to q, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Second, we assume $(C, J, L, H, R) \Vdash \langle b \to p \parallel b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 27 has been applied necessarily. Then, $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, four cases can be distinguished:

1. Rule 23 has been applied for both. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. According to Rule 27, we obtain $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 23, we get $(C, J, L, H, R) \Vdash \langle b \to (p \parallel q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

2. Rule 24 has been applied for both. Then $\xi \equiv \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. According to Rule 24, we can have $(C, J, L, H, R) \Vdash \langle b \to (p \parallel q), \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

3. Rule 23 has been applied for $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and Rule 24 has been applied for $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Then, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $\xi \models b$, and $\xi = \sigma \cup \xi^{\dot{C}L}$, and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. This leads to a contradiction. Therefore, Rule 23 cannot have been applied for $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ or Rule 24 cannot have been applied for $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

4. Rule 23 has been applied for $(C, J, L, H, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and Rule 24 has been applied for $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. The proof is similar to the previous case.

B.5 Properties of sequential composition

The following lemmas prove the properties of Proposition 3.5.5.

Lemma B.5.1 (Left-zero element for sequential composition) *For every closed process term p we have*

$$\delta;\ p \leftrightarrow \delta.$$

PROOF. Let $R = \{(\delta;\ p, \delta) \mid p \in P\}$. Since there are no action transition rules and time transition rules defined for δ, and therefore also not for $\delta;\ p$, the conditions 1 – 5 hold trivially.

Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle \delta;\ p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 19 has been applied necessarily. Then, $(C, J, L, H, R) \Vdash \langle \delta, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $(C, J, L, H, R) \Vdash \langle \delta, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $(C, J, L, H, R) \Vdash \langle \delta;\ p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.5.2 (Associativity of sequential composition) *For all closed process terms p, q and r we have*

$$(p;\ q);\ r \leftrightarrow p;\ (q;\ r).$$

PROOF. Let $R = \{((p;\ q);\ r, p;\ (q;\ r)) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of conditions 4 and 5 are similar to the proofs of conditions 2 and 3 (except Rule 16 has not been applied, because no χ process can transform to a terminated process by means of time transitions) since the deduction rules for non-terminating action transitions and time transitions of ; are similar.

Condition 1: Since there are no termination transitions defined for the transitions $\langle (p\ ;\ q);\ r, \sigma\rangle$ and $\langle p;\ (q;\ r), \sigma\rangle$, condition 1 holds trivially.

Condition 2: We assume $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we distinguish two cases:

1. Rule 16 has been applied. Then $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma'\rangle$. This leads to a contradiction as there is no deduction rule that allows a sequential composition to perform a termination transition. Hence, this case cannot occur.

2. Rule 17 has been applied. Then, $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv k_1';\ r$. We distinguish two cases for $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1', \sigma'\rangle$:

   (a) Rule 16 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma'\rangle$, $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1' \equiv q$. According to Rule 19, we have $E \Vdash \langle q;\ r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 16, we have $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle q;\ r, \sigma'\rangle$. Note that $k_1 \equiv q;\ r$. Take $k_2 \equiv q;\ r$ and observe that $(k_1, k_2) \in R$.

   (b) Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1' \equiv k_p;\ q$. Using Rule 17 we obtain $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_p;\ (q;\ r), \sigma'\rangle$. Note that $k_1 \equiv (k_p;\ q);\ r$. Take $k_2 \equiv k_p;\ (q;\ r)$ and observe that $(k_1, k_2) \in R$.

Condition 3: We assume $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we distinguish two cases:

1. Rule 16 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $E \Vdash \langle q;\ r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1 \equiv q;\ r$. According to Rule 19, we have $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 16, we obtain $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Using Rule 17, we obtain $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q;\ r, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

2. Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p;\ (q;\ r)$. Using Rule 17, we obtain $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$. Again, using Rule 17, we obtain $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle (k_p;\ q);\ r, \sigma'\rangle$. Take $k_2 \equiv (k_p;\ q);\ r$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means that Rule 19 has applied necessarily. Then $E \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 19, we get $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means that Rule 19 has applied necessarily. Then $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 19, we get $E \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.5.3 (Distribution of sequential over alternative composition) *For p, q and r arbitrary closed process terms we have*

$$(p \mathbin{[\!]} q);\ r \leftrightarrows p;\ r \mathbin{[\!]} q;\ r.$$

PROOF. Let $R = \{((p \mathbin{[\!]} q);\ r, p;\ r \mathbin{[\!]} q;\ r) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.
Condition 1: Since there no action transition rules defined for any closed process term k_1 and k_2 such that $E \Vdash \langle k_1;\ k_2, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, condition 1 holds trivially.
Condition 2: We assume $E \Vdash \langle (p \mathbin{[\!]} q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ'; which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we can distinguish two cases:

1. Rule 16 has been applied. Then, $E \Vdash \langle p \mathbin{[\!]} q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, and $E \Vdash \langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. For $E \Vdash \langle p \mathbin{[\!]} q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ we again distinguish two cases:

   (a) Rule 25.1.l has been applied. Then, $k_1 \equiv r$, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 16, we get $E \Vdash \langle p;\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle r, \sigma'\rangle$. Due to Rule 19, we have $E \Vdash \langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 25.2.l, we have $E \Vdash \langle p;\ r \mathbin{[\!]} q;\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle r, \sigma'\rangle$ and observe that $(r, r) \in R$.

(b) Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

2. Rule 17 has been applied. Then, $E \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv k_1'; r$. For $E \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ we can further distinguish two cases:

   (a) Rule 25.2.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 17, we get $E \Vdash \langle p; r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1'; r, \sigma'\rangle$. Using Rule 19, we get $\langle q; r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 25.2.l, we have $E \Vdash \langle p; r \parallel q; r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1'; r, \sigma'\rangle$. Take $k_2 \equiv k_1'; r$ and observe that $(k_1, k_2) \in R$.

   (b) Rule 25.2.r has been applied. The proof is similar to the proof of the previous case.

Condition 3: We assume $E \Vdash \langle p; r \parallel q; r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ'; which means that either Rule 25.2.l or Rule 25.2.r has been applied necessarily. Hence, we can distinguish two cases:

1. Rule 25.2.l has been applied. Then, $E \Vdash \langle p; r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle q; r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle q; r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, we must have $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ due to Rule 19. For $E \Vdash \langle p; r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ we again distinguish two cases:

   (a) Rule 16 has been applied. Then, $k_1 \equiv r$, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, and $\langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Applying Rule 25.1.l, we get $E \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. According to Rule 16, we have $E \Vdash \langle (p \parallel q); r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle r, \sigma'\rangle$ and observe that $(r, r) \in R$.

   (b) Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p; r$. Using Rule 25.2.l, we get $E \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$. According to Rule 17, we have $E \Vdash \langle (p \parallel q); r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p; r, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

2. Rule 25.2.r has been applied. The proof is similar to the proof of the previous case.

Condition 4: We assume $E \Vdash \langle (p \parallel q); r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ'; which means Rule 18 has been applied necessarily. Then, $E \Vdash \langle p \parallel q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ for some k_{pq} such that $k_1 \equiv k_{pq}; r$. For $E \Vdash \langle p \parallel q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, Rule 26 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some k_p, k_q such that $k_{pq} \equiv k_p \parallel k_q$. Using Rule 18, we obtain $E \Vdash \langle p; r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p; r, \sigma'\rangle$ and $E \Vdash \langle q; r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q; r, \sigma'\rangle$. According to Rule 26 we obtain $E \Vdash \langle p; r \parallel q; r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p; r \parallel k_q; r, \sigma'\rangle$. Note that $k_1 \equiv (k_p \parallel k_q); r$. Take $k_2 \equiv k_p; r \parallel k_q; r$ and observe that $(k_1, k_2) \in R$.

Condition 5: We assume $E \Vdash \langle p; r \parallel q; r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ'; which means Rule 26 has been applied necessarily. Then, $E \Vdash \langle p; r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pr}, \sigma'\rangle$ and $E \Vdash$

$\langle q;\, r, \sigma\rangle \xmapsto{t,\rho} \langle k_{qr}, \sigma'\rangle$ for some k_{pr}, k_{qr} such that $k_1 \equiv k_{pr} \parallel k_{qr}$. For $E \Vdash \langle p;\, r, \sigma\rangle \xmapsto{t,\rho} \langle k_{pr}, \sigma'\rangle$ and $E \Vdash \langle q;\, r, \sigma\rangle \xmapsto{t,\rho} \langle k_{qr}, \sigma'\rangle$, Rule 18 has been applied to both. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \xmapsto{t,\rho} \langle k_q, \sigma'\rangle$ for some k_p, k_q such that $k_{pr} \equiv k_p;\, r$ and $k_{qr} \equiv k_q;\, r$. Using Rule 26 we then obtain $E \Vdash \langle p \parallel q, \sigma\rangle \xmapsto{t,\rho} \langle k_p \parallel k_q, \sigma'\rangle$. Applying Rule 19, we get $E \Vdash \langle (p \parallel q);\, r, \sigma\rangle \xmapsto{t,\rho} \langle (k_p \parallel k_q);\, r, \sigma'\rangle$. Note that $k_1 \equiv k_p;\, r \parallel k_q;\, r$. Take $k_2 \equiv (k_p \parallel k_q);\, r$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $E \Vdash \langle (p \parallel q);\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ; which means Rule 19 has been applied necessarily. Then, $E \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we have $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 19, we get $E \Vdash \langle p;\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q;\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 27, we get $E \Vdash \langle p;\, r \parallel q;\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p;\, r \parallel q;\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ; which means Rule 27 has been applied necessarily. Then, $E \Vdash \langle p;\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q;\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. By Rule 19, we obtain $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we get $E \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 19, we have $E \Vdash \langle (p \parallel q);\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.5.4 *For arbitrary closed process terms p and q and arbitrary guard b we have*

$$b \to (p;\, q) \leftrightarrows b \to p;\, q.$$

PROOF. Let $R = \{(b \to (p;\, q), b \to p;\, q) \mid p, q \in P, \text{guard } b\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

Condition 1: We assume $E \Vdash \langle b \to (p;\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means that Rule 20.1 has been applied necessarily. Then, $E \Vdash \langle p;\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models b$. This leads to a contradiction as there is no deduction rule that allows a sequential composition to perform a termination transition. Second, we assume $E \Vdash \langle b \to p;\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ'. This also leads to a contradiction as there is no deduction rule that allows a sequential composition to perform a termination transition. Thus, condition 1 holds trivially.

Condition 2: We assume $E \Vdash \langle b \to (p;\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 20.2 has been applied necessarily. Then, we have $E \Vdash \langle p;\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, and $\xi \models b$. For $E \Vdash \langle p;\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, two cases can be distinguished:

1. Rule 16 has been applied. Then, $k_1 \equiv q$, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 20.1 we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Applying Rule 16 we have $E \Vdash \langle b \to p;\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$ and observe that $(q, q) \in R$.

2. Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_q such that $k_1 \equiv k_p;\, q$. Using Rule 20.2 we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$, and using Rule 17 we have $E \Vdash \langle b \to p;\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\, q, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 3: We assume $E \Vdash \langle b \to p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means that Rule 16 or Rule 17 has been applied necessarily. Then, we distinguish two cases:

1. Rule 16 has been applied. Then, $k_1 \equiv q$, $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. According to Rule 20.1 we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $\xi \models b$. Applying Rule 16 we have $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Using Rule 20.2 we get $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$ and observe that $(q, q) \in R$.

2. Rule 17 has been applied. Then $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p;\ q$. Using Rule 20.2, we obtain $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ and $\xi \models b$. Using Rule 17, we get $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$. Applying Rule 20.2, we have $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

Condition 4: We assume $E \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that either Rule 21 or Rule 22 has been applied necessarily. Then we can distinguish two cases:

1. Rule 21 has been applied. Then, $E \Vdash \langle p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$ and $\forall_{s\in[0,t]}\, \rho(t) \models b$ such that $k_1 \equiv b \to k_1'$. For $E \Vdash \langle p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$, we further get $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ using Rule 18 for some k_p such that $k_1' \equiv k_p;\ q$. Applying Rule 21, we get $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p, \sigma'\rangle$. According to Rule 18, we obtain $E \Vdash \langle b \to p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p;\ q, \sigma'\rangle$. Note that $k_1 \equiv b \to (k_p;\ q)$. Take $k_2 \equiv b \to k_p;\ q$ and observe that $(k_1, k_2) \in R$.

2. Rule 22 has been applied. Then, $k_1 \equiv b \to (p;\ q)$, $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\exists_{s\in[0,t]}\, \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle p;\ q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, for some p', σ'', $\rho(t) \models b \Rightarrow E \Vdash \langle p;\ q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p;\ q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, we get $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k_p', \sigma''\rangle$, for some k_p' (see also Rule 18). For $\rho(t) \models b \Rightarrow E \Vdash \langle p;\ q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (see also Rule 19). Using Rule 22, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p, \rho_\sigma(t)\rangle$. Using Rule 18, we obtain $E \Vdash \langle b \to p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p;\ q, \rho_\sigma(t)\rangle$. Take $k_2 \equiv b \to p;\ q$ and observe that $(k_1, k_2) \in R$.

Condition 5: We assume $E \Vdash \langle b \to p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that Rule 18 has been applied necessarily. Then $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv k_1';\ q$. For $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$, two cases can be distinguished:

1. Rule 21 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $\forall_{s\in[0,t]}\, \rho(t) \models b$ for some k_p such that $k_1' \equiv b \to k_p$. Using Rule 18, we get $E \Vdash \langle p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p;\ q, \sigma'\rangle$.

According to Rule 21, we have $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xmapsto{t,\rho} \langle b \to (k_p;\ q), \sigma'\rangle$. Note that $k_1 \equiv b \to k_p;\ q$. Take $k_2 \equiv b \to (k_p;\ q)$ and observe that $(k_2, k_1) \in R$.

2. Rule 22 has been applied. Then, $k_1' \equiv b \to p$, $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\exists_{s\in[0,t]}\ \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, for some p', σ'', and $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, we get $\rho(0) \models b \Rightarrow E \Vdash \langle p;\ q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k_p', \sigma''\rangle$ using Rule 18 for some k_p'. From $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(t) \models b \Rightarrow E \Vdash \langle p;\ q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ using Rule 19. Applying Rule 22, we obtain $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xmapsto{t,\rho} \langle b \to (p;\ q), \rho_\sigma(t)\rangle$. Note that $k_1 \equiv b \to p;\ q$. Take $k_2 \equiv b \to (p;\ q)$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that either Rule 23 or Rule 24 has been applied necessarily. Then we can distinguish two cases:

1. Rule 23 has been applied. Then, $(C, J, L, H, R) \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. From Rule 19, we obtain $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 23, we get $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 19, we obtain $(C, J, L, H, R) \Vdash \langle b \to p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

2. Rule 24 has been applied. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24 we get $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Applying Rule 19, we obtain $(C, J, L, H, R) \Vdash \langle b \to p;\ q, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Second, we assume $(C, J, L, H, R) \Vdash \langle b \to p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 19 has been applied necessarily. Then $(C, J, L, H, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For this, we can distinguish two cases:

1. Rule 23 has been applied. Then, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. Using Rule 19, we obtain $(C, J, L, H, R) \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 23, we get $(C, J, L, H, R) \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

2. Rule 24 has been applied. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24 we get $(C, J, L, H, R) \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Lemma B.5.5 *For arbitrary closed process term p and arbitrary predicate u we have*

$$u;\ p \leftrightarrows u.$$

PROOF. Let $R = \{u;\ p, u) \mid p \in P, \text{predicate } u\}$. Since there are no termination and action transition rules defined for u, this means that $u;\ p$ has no termination and action transitions. So, the conditions 1 – 3 hold trivially.

Condition 4: We assume $E \Vdash \langle u;\ p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that Rule 18 has been applied necessarily. Then, $E \Vdash \langle u, \sigma\rangle \xmapsto{t,\rho} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv k_1';\ p$. Observe that $(k_1, k_1') \in R$.
Condition 5: We assume $E \Vdash \langle u, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$. According to Rule 3, we obtain $k_1 \equiv u$, $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$ and $\sigma' = \rho_\sigma(t)$. Using Rule 18, we can have $E \Vdash \langle u;\ p, \sigma\rangle \xmapsto{t,\rho} \langle k_2, \sigma'\rangle$ for some k_2 such that $k_2 \equiv u;\ p$. Observe that $(k_2, k_1) \in R$.
Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle u;\ p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 19 has been applied necessarily. Then, $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$. According to Rule 19, we get $(C, J, L, H, R) \Vdash \langle u;\ p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Lemma B.5.6 *For arbitrary closed process terms p and q we have*

$$[p];\ q \leftrightarrows [p;\ q].$$

PROOF. Let $R = \{([p];\ q, [p;\ q]) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4. The proof of condition 6 is trivial, because process term $[p]$ (also $[p];\ q$, see Rule 19) and $[p;\ q]$ are consistent with respect to σ in any environment.
Condition 1: Since there are no termination transition rules defined for $\langle [p];\ q, \sigma\rangle$ and $\langle [p;\ q], \sigma\rangle$, condition 1 holds trivially.
Condition 2: We assume $E \Vdash \langle [p];\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we distinguish two cases:

1. Rule 16 has been applied. Then $E \Vdash \langle [p], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1 \equiv q$. According to Rule 10.1, we obtain $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Using Rule 16, we can have $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Applying Rule 10.2, we get $E \Vdash \langle [p;\ q], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Take $k_2 \equiv q$ and observe that $(k_1, k_2) \in R$.

2. Rule 17 has been applied. Then $E \Vdash \langle [p], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv k_1';\ q$. According to Rule 10.2, we obtain $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$. Using Rule 17, we can have $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1';\ q, \sigma'\rangle$. Applying Rule 10.2, we get $E \Vdash \langle [p;\ q], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1';\ q, \sigma'\rangle$. Take $k_2 \equiv k_1';\ q$ and observe that $(k_1, k_2) \in R$.

Condition 4: We assume $E \Vdash \langle [p];\ q, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$; which means Rule 18 has been applied necessarily. Then, $E \Vdash \langle [p], \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p;\ q$. According to Rule 11, we obtain $k_p \equiv [p]$, $\rho \in \Omega_{\sigma Et}$, and $\sigma' = \rho_\sigma(t)$. Using

Rule 11, we obtain $E \Vdash \langle [p;\ q], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle [p;\ q], \sigma'\rangle$. Take $k_2 \equiv [p;\ q]$ and observe that $(k_1, k_2) \in R$.

B.6 Properties of parallel composition

The following lemmas prove the properties of Proposition 3.5.6.

Lemma B.6.1 (Commutativity of parallel composition) *For arbitrary closed process terms p and q we have*

$$p \parallel q \leftrightarrow q \parallel p.$$

PROOF. Let $R = \{(p \parallel q, q \parallel p) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since the deduction rules for $\parallel$ are symmetrical w.r.t. the left and right argument, obviously all conditions are met.

Lemma B.6.2 (Associativity of parallel composition) *For arbitrary closed process terms p, q and r we have*

$$(p \parallel q) \parallel r \leftrightarrow p \parallel (q \parallel r).$$

PROOF. Let $R = \{((p \parallel q) \parallel r, p \parallel (q \parallel r)) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similar to the proof of the right implication of condition 1. The proof of condition 3 is similar to the proof of condition 2. The proofs of conditions 4 – 6 are the same as the proofs of conditions 4 – 6 of Lemma B.3.4 (apart from the operator that has been used), because the deduction rules defined for the time transitions and the consistency predicates for $[\![$ and $\parallel$ are the same. To increase the readability of this proof, we often apply Lemma 3.5.6 to obtain $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ from $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ or the other way around without mentioning explicitly the use of Lemma 3.5.6.
Condition 1: We assume $(C, J, L, H, R) \Vdash \langle (p \parallel q) \parallel r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma'\rangle$ for some C, J, L, H, R, σ, ξ, a, ξ', σ', which means that either Rule 28.1.l or Rule 28.1.r has been applied necessarily. Hence, we distinguish two cases:

1. Rule 28.1.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma\rangle$, $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$ for some W, h ,cs and $a = ca(h, cs)$. Since we do not have a rule for $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma'\rangle$, we obtain a contradiction and the right implication of condition 1 holds trivially.

2. Rule 28.1.r has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. Since we do not have a rule for $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$, we obtain a contradiction and the right implication of condition 1 holds trivially.

Condition 2: We assume $(C, J, L, H, R) \Vdash \langle (p \parallel q) \parallel r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, ξ, a, ξ', k_1, σ'. Based on the deduction rule that has been applied we can distinguish ten cases:

1. Rule 28.2.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_1, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. For $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_1, \sigma'\rangle$ we can distinguish four more cases:

   (a) Rule 29.1.l has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma\rangle$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv q$. Using Rule 29.1.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle q, \sigma'\rangle$. Using Rule 28.3.l we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle q, \sigma'\rangle$, and observe that $(q, q) \in R$.

   (b) Rule 29.1.r has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv p$. Using Rule 28.1.l we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$. Using Rule 29.1.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle p, \sigma'\rangle$ and observe that $(p, p) \in R$.

   (c) Rule 29.2.l has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p \parallel q$, and $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.1.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle q, \sigma'\rangle$. Using Rule 28.4.l we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_p \parallel q, \sigma'\rangle$. Take $k_2 \equiv k_p \parallel q$ and observe that $(k_1, k_2) \in R$.

   (d) Rule 29.2.r has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_q, \sigma'\rangle$ for some k_q such that $k_1 \equiv p \parallel k_q$, and $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.2.l we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_q, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle p \parallel k_q, \sigma'\rangle$. Take $k_2 \equiv p \parallel k_q$ and observe that $(k_1, k_2) \in R$.

2. Rule 28.2.r has been applied. Then, $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ and $(C, J \cup W, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_1, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. This case cannot occur since the conclusion $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ cannot be obtained from the deduction rules.

3. Rule 28.3.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_1, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. The conclusion $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$ cannot be obtained from the deduction rules. Hence, this case cannot occur.

4. Rule 28.3.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_1, \sigma'\rangle$ and $(C, J \cup W, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. For $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_1, \sigma'\rangle$ we can distinguish four more cases:

   (a) Rule 29.1.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1 \equiv q$. Using Rule 29.1.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q, \sigma'\rangle$. Using Rule 28.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle q, \sigma'\rangle$ and observe that $(q, q) \in R$.

   (b) Rule 29.1.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv p$. Using Rule 28.1.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$. Using Rule 29.1.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle p, \sigma'\rangle$ and observe that $(p, p) \in R$.

   (c) Rule 29.2.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv k_p \parallel q$, and $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.1.r we obtain $(C, J \cup W, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q, \sigma'\rangle$. Using Rule 28.4.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_p \parallel q, \sigma'\rangle$. Take $k_2 \equiv k_p \parallel q$ and observe that $(k_1, k_2) \in R$.

   (d) Rule 29.2.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_q, \sigma'\rangle$ for some k_q such that $k_1 \equiv p \parallel k_q$, and $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.3.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_q, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle p \parallel k_q, \sigma'\rangle$. Take $k_2 \equiv p \parallel k_q$ and observe that $(k_1, k_2) \in R$.

5. Rule 28.4.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_r, \sigma'\rangle$ for some W, h, cs, k_{pq}, k_r such that $k_1 \equiv k_{pq} \parallel k_r$, and $a = ca(h, cs)$. For $(C, J \cup W, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$ four cases can be distinguished:

(a) Rule 29.1.l has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv q$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.3.l, we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel k_r$. Take $k_2 \equiv q \parallel k_r$ and observe that $(k_1, k_2) \in R$.

(b) Rule 29.1.r has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv p$. Using Rule 28.3.l we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv p \parallel k_r$. Take $k_2 \equiv p \parallel k_r$ and observe that $(k_1, k_2) \in R$.

(c) Rule 29.2.l has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_{pq} \equiv k_p \parallel q$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.4.l we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_p \parallel (q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel q) \parallel k_r$. Take $k_2 \equiv k_p \parallel (q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

(d) Rule 29.2.r has been applied. Then, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_q, \sigma'\rangle$ for some k_q such that $k_{pq} \equiv p \parallel k_q$, and $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.4.l we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_q \parallel k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel (k_q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (p \parallel k_q) \parallel k_r$. Take $k_2 \equiv p \parallel (k_q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

6. Rule 28.4.r has been applied. Then, $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$, $(C, J \cup W, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_r, \sigma'\rangle$ for some W, h, cs, k_{pq}, k_r such that $k_1 \equiv k_{pq} \parallel k_r$, and $a = ca(h, cs)$. For $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$ we can distinguish four more cases:

(a) Rule 29.1.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_{pq} \equiv q$. Using Rule 29.2.r we obtain $(C, J \cup W, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel k_r$. Take $k_2 \equiv q \parallel k_r$ and observe that $(k_1, k_2) \in R$.

(b) Rule 29.1.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$,

$(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_{pq} \equiv p$. Using Rule 28.2.r we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle b \rightarrow q, \sigma\rangle \overset{\xi}{\rightsquigarrow} \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv p \parallel k_r$. Take $k_2 \equiv p \parallel k_r$ and observe that $(k_1, k_2) \in R$.

(c) Rule 29.2.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_{pq} \equiv k_p \parallel q$, and $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.r we obtain $(C, J \cup W, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.4.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_p \parallel (q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel q) \parallel k_r$. Take $k_2 \equiv k_p \parallel (q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

(d) Rule 29.2.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_q, \sigma'\rangle$ for some k_q such that $k_{pq} \equiv p \parallel k_q$, and $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.4.r we obtain $(C,J,L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_q \parallel k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel (k_q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (p \parallel k_q) \parallel k_r$. Take $k_2 \equiv p \parallel (k_q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

7. Rule 29.1.l has been applied. Then, $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $k_1 \equiv r$, and $(C, J, L, H, R) \Vdash \langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Then two cases can be considered:

   (a) Rule 28.1.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs and $a = ca(h, cs)$. Using Rule 29.1.l we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle r, \sigma'\rangle$. Using Rule 28.3.l we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle r, \sigma'\rangle$, and observe that $(r, r) \in R$.

   (b) Rule 28.1.r has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. Then, using Rule 29.1.l we obtain $(C, J \cup W, L, H, R)$ $\langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle r, \sigma'\rangle$. Using Rule 28.3.r we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle r, \sigma'\rangle$, and observe that $(r, r) \in R$.

8. Rule 29.1.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv p \parallel q$.

According to Rule 31, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.1.r, we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Then, using Rule 29.2.r, we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel q, \sigma'\rangle$. Take $k_2 \equiv p \parallel q$ and observe that $(k_1, k_2) \in R$.

9. Rule 29.2.l has been applied. Then, $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_{pq}, \sigma'\rangle$ for some term k_{pq} such that $k_1 \equiv k_{pq} \parallel r$ and $(C, J, L, H, R) \Vdash \langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. For $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_{pq}, \sigma'\rangle$, ten cases can be distinguished.

   (a) Rule 28.2.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. Then applying Rule 29.1.l followed by Rule 28.4.l gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

   (b) Rule 28.2.r has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$ for some W, h, cs, and $a = ca(h,cs)$. Then Rule 29.1.l followed by Rule 28.4.r gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

   (c) Rule 28.3.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle \checkmark, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$ for some W, h, cs, and $a = ca(h, cs)$. Then Rule 29.2.l followed by Rule 28.3.l gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

   (d) Rule 28.3.r has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ for some W, h, cs, $a = ca(h, cs)$. Then applying Rule 29.1.l followed by Rule 28.4.r gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

   (e) Rule 28.4.l has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_p, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_q, \sigma'\rangle$ for some W, h, cs, k_p, k_q such that $k_{pq} \equiv k_p \parallel k_q$, and $a = ca(h, cs)$. Then applying Rule 29.2.l followed by Rule 28.4.l gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_p \parallel (k_q \parallel r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel k_q) \parallel r$. Take $k_2 \equiv k_p \parallel (k_q \parallel r)$ and observe that $(k_1, k_2) \in R$.

(f) Rule 28.4.r has been applied. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_q, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_p, \sigma'\rangle$ for some W, h, cs, k_p, k_q such that $k_{pq} \equiv k_p \parallel k_q$, and $a = ca(h, cs)$. Then applying Rule 29.2.l followed by Rule 28.4.r gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p \parallel (k_q \parallel r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel k_q) \parallel r$. Take $k_2 \equiv k_p \parallel (k_q \parallel r)$ and observe that $(k_1, k_2) \in R$.

(g) Rule 29.1.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv q$. We have $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ (see Rule 31). Applying Rule 29.1.l gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q \parallel r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel r$. Take $k_2 \equiv q \parallel r$ and observe that $(k_1, k_2) \in R$.

(h) Rule 29.1.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_{pq} \equiv p$. Applying Rule 29.1.l and then Rule 29.2.r gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel r$. Take $k_2 \equiv p \parallel r$ and observe that $(k_1, k_2) \in R$.

(i) Rule 29.2.l has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_{pq} \equiv k_p \parallel q$, and $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. We have $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ (see Rule 31). Applying Rule 29.2.l gives $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p \parallel (q \parallel r), \sigma'\rangle$. Take $k_2 \equiv k_p \parallel (q \parallel r)$ and observe that $(k_1, k_2) \in R$.

(j) Rule 29.2.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_q, \sigma'\rangle$ for some k_q such that $k_{pq} \equiv p \parallel k_q$, and $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Applying Rule 29.2.l and then Rule 29.2.r gives (C, J, L, H, R) $\langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel (k_q \parallel r), \sigma'\rangle$. Notice $k_1 \equiv (p \parallel k_q) \parallel r$. Take $k_2 \equiv p \parallel (k_q \parallel r)$ and observe that $(k_1, k_2) \in R$.

10. Rule 29.2.r has been applied. Then, we have $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_r, \sigma'\rangle$ for some k_r such that $k_1 \equiv (p \parallel q) \parallel k_r$, and $(C, J, L, H, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. According to Rule 31, we have $(C, J, L, H, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, H, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.r, we obtain $(C, J, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 29.2.r, we obtain $(C, J, L, H, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel (q \parallel k_r), \sigma'\rangle$. Take $k_2 \equiv p \parallel (q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

Lemma B.6.3 *For arbitrary predicates u, u' we have*

$$u \parallel u' \leftrightarrow u \wedge u'.$$

PROOF. Let $R = \{(u \parallel u', u \wedge u') \mid \text{predicates } u, u'\}$. The fact that there are no action transition rules defined for u, u', also indicates that $u \parallel u'$ and $u \wedge u'$ have no action transitions. Therefore, the conditions 1 – 3 hold trivially.

Condition 4: We assume $(C, J, L, H, R) \Vdash \langle u \parallel u', \sigma\rangle \xrightarrow{t,\rho} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ', which means that Rule 30 has been applied necessarily. Then, $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \xrightarrow{t,\rho} \langle k_u, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle u', \sigma\rangle \xrightarrow{t,\rho} \langle k'_u, \sigma'\rangle$ for some k_u, k'_u such that $k_1 \equiv k_u \parallel k'_u$. For $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \xrightarrow{t,\rho} \langle k_u, \sigma'\rangle$ and $(C, J, L, H, R) \Vdash \langle u', \sigma\rangle \xrightarrow{t,\rho} \langle k'_u, \sigma'\rangle$, Rule 3 has been applied necessarily. Then, $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$, $\sigma' = \rho_\sigma(t)$, $k_u \equiv u$, and $\rho \in \Omega_{FG}(\sigma, C, L, u', t)$, $\sigma' = \rho_\sigma(t)$ and $k'_u \equiv u'$. From the definition of the function Ω_{FG}, we know that $\forall s \in [0, t] : \rho(s) \models u$ and $\forall s \in [0, t] : \rho(s) \models u'$. We can further get $\forall s \in [0, t] : \rho(s) \models u \wedge u'$. Hence, it is not hard to see that $\rho \in \Omega_{FG}(\sigma, C, L, u \wedge u', t)$. Using Rule 3, we obtain $(C, J, L, H, R) \Vdash \langle u \wedge u', \sigma\rangle \xrightarrow{t,\rho} \langle u \wedge u', \rho_\sigma(t)\rangle$, and observe that $(u \parallel u', u \wedge u') \in R$.

Condition 5: We assume $(C, J, L, H, R) \Vdash \langle u \wedge u', \sigma\rangle \xrightarrow{t,\rho} \langle k_1, \sigma'\rangle$ for some C, J, L, H, R, σ, t, ρ, k_1, σ', which means that Rule 3 has been applied necessarily. Then, $\rho \in \Omega_{FG}(\sigma, C, L, u \wedge u', t)$, $\sigma' = \rho_\sigma(t)$ and $k_1 \equiv u \wedge u'$. From the definition of the function Ω_{FG}, we know that $\forall s \in [0, t] : \rho(s) \models u \wedge u'$. We can further get $\forall s \in [0, t] : \rho(s) \models u$ and $\forall s \in [0, t] : \rho(s) \models u'$. Hence, it is not hard to see that $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$ and $\rho \in \Omega_{FG}(\sigma, C, L, u', t)$. Using Rule 3, we obtain $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \xrightarrow{t,\rho} \langle u, \rho_\sigma(t)\rangle$ and $(C, J, L, H, R) \Vdash \langle u', \sigma\rangle \xrightarrow{t,\rho} \langle u', \rho_\sigma(t)\rangle$. Applying Rule 30, we obtain $(C, J, L, H, R) \Vdash \langle u \parallel u', \sigma\rangle \xrightarrow{t,\rho} \langle u \parallel u', \rho_\sigma(t)\rangle$, and observe that $(u \wedge u', u \parallel u') \in R$.

Condition 6: First, we assume $(C, J, L, H, R) \Vdash \langle u \parallel u', \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means that Rule 31 has been applied necessarily. Then, $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle u', \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle u', \sigma\rangle \overset{\xi}{\rightsquigarrow}$, Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$, $\sigma \cup \xi^{\dot{C}L} \models u$, and $\sigma \cup \xi^{\dot{C}L} \models u'$. We can further obtain $\sigma \cup \xi^{\dot{C}L} \models u \wedge u'$. Using Rule 4, we can have $(C, J, L, H, R) \Vdash \langle u \wedge u', \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $(C, J, L, H, R) \Vdash \langle u \wedge u', \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some C, J, L, H, R, σ, ξ, which means Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models u \wedge u'$. We can further get $\sigma \cup \xi^{\dot{C}L} \models u$ and $\sigma \cup \xi^{\dot{C}L} \models u'$. Using Rule 4, we can have $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$ and $(C, J, L, H, R) \Vdash \langle u', \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. According to Rule 31, we get $(C, J, L, H, R) \Vdash \langle u \parallel u', \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

B.7 Properties of action encapsulation operator

The following lemmas prove the properties of Proposition 3.5.7.

Lemma B.7.1 *For arbitrary set of actions A we have*

$$\partial_A(\delta) \leftrightarrow \delta.$$

PROOF. Let $R = \{(\partial_A(\delta), \delta)\}$. Process term δ cannot perform any transitions. However, it is consistent for arbitrary extended valuations. Also, action encapsulation has no effect on consistency. Hence, conditions 1–6 hold trivially.

Lemma B.7.2 *For arbitrary closed process term p we have*

$$\partial_\emptyset(p) \leftrightarrow p.$$

PROOF. Let $R = \{(\partial_\emptyset(p), p) \mid p \in P\}$.
Condition 1: First, we assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$. We know that $a \notin \emptyset$. Using Rule 32.1, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.
Condition 2: We assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_\emptyset(k_p)$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.
Condition 3: We assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$. We know that $a \notin \emptyset$. Using Rule 32.2, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_\emptyset(k_1), \sigma'\rangle$. Take $k_2 \equiv \partial_\emptyset(k_1)$ and observe that $(k_2, k_1) \in R$.
Condition 4: We assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_\emptyset(k_p)$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.
Condition 5: We assume $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$. Using Rule 33, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xmapsto{t,\rho} \langle \partial_\emptyset(k_1), \sigma'\rangle$. Take $k_2 \equiv \partial_\emptyset(k_1)$ and observe that $(k_2, k_1) \in R$.
Condition 6: First, we assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ. Using Rule 34, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.7.3 *For arbitrary closed process term p and sets of actions A and A' we have*

$$\partial_A(\partial_{A'}(p)) \leftrightarrow \partial_{A\cup A'}(p).$$

PROOF. Let $R = \{(\partial_A(\partial_{A'}(p)), \partial_{A\cup A'}(p)) \mid p \in P, \text{sets of actions } A, A'\}$.
Condition 1: First, we assume $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and

$a \notin A$. Again, due to Rule 32.1, we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $a \notin A'$. From $a \notin A$ and $a \notin A'$, we know that $a \notin A \cup A'$. Using Rule 32.1, we obtain $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $a \notin A \cup A'$. From $a \notin A \cup A'$, we know that $a \notin A$ and $a \notin A'$. Using Rule 32.1, we get $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Again, using Rule 32.1, we obtain $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_A(k_p)$ and $a \notin A$. Using Rule 32.2, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k'_p, \sigma'\rangle$ for some k'_p such that $k_p \equiv \partial_{A'}(k'_p)$ and $a \notin A'$. From $a \notin A$ and $a \notin A'$, we know that $a \notin A \cup A'$. Using Rule 32.2, we get $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_{A\cup A'}(k'_p), \sigma'\rangle$. Note that $k_1 \equiv \partial_A(\partial_{A'}(k'_p))$. Take $k_2 \equiv \partial_{A\cup A'}(k'_p)$ and observe that $(k_1, k_2) \in R$.

Condition 3: We assume $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1, σ', which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_{A\cup A'}(k_p)$ and $a \notin A \cup A'$. From $a \notin A \cup A'$, we know that $a \notin A$ and $a \notin A'$. Using Rule 32.2, we get $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_{A'}(k_p), \sigma'\rangle$. Again, due to Rule 32.2, we have $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(\partial_{A'}(k_p)), \sigma'\rangle$. Take $k_2 \equiv \partial_A(\partial_{A'}(k_p))$ and observe that $(k_2, k_1) \in R$.

Condition 4: We assume $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_A(k_p)$. Again, due to Rule 33, we get $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k'_p, \sigma'\rangle$ for some k'_p such that $k_p \equiv \partial_{A'}(k'_p)$. Using Rule 33, we obtain $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \xmapsto{t,\rho} \langle \partial_{A\cup A'}(k'_p), \sigma'\rangle$. Note that $k_1 \equiv \partial_A(\partial_{A'}(k'_p))$. Take $k_2 \equiv \partial_{A\cup A'}(k'_p)$ and observe that $(k_1, k_2) \in R$.

Condition 5: We assume $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_{A\cup A'}(k_p)$. Using Rule 33, we get $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \xmapsto{t,\rho} \langle \partial_{A'}(k_p), \sigma'\rangle$. Due to Rule 33, we obtain $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \xmapsto{t,\rho} \langle \partial_A(\partial_{A'}(k_p)), \sigma'\rangle$. Take $k_2 \equiv \partial_A(\partial_{A'}(k_p))$ and observe that $(k_2, k_1) \in R$.

Condition 6: First, we assume $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle \partial_{A'}(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 34, we have $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Then using Rule 34, we obtain $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle \partial_{A\cup A'}(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34, we obtain $\langle E \Vdash \partial_{A'}(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34 again, we obtain $E \Vdash \langle \partial_A(\partial_{A'}(p)), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.7.4 *For arbitrary closed process terms p, q and set of actions A we have*

$$\partial_A(p \between q) \leftrightarrow \partial_A(p) \between \partial_A(q).$$

PROOF. Let $R = \{(\partial_A(p \between q), \partial_A(p) \between \partial_A(q)) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of the left implication of conditions 1 and 6 are similar to the proofs of the right implication of conditions 1 and 6. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle \partial_A(p \between q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, and we know that $a \notin A$. For $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, we further distinguish two cases:

- Rule 25.1.l has been applied. Then, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 32.1, we can have $E \Vdash \langle \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Also, we obtain $E \Vdash \langle \partial_A(q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 34. According to Rule 25.1.l, we have $E \Vdash \langle \partial_A(p) \between \partial_A(q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.
- Rule 25.1.r has been applied. The proof is similar to the previous case.

Condition 2: We assume $E \Vdash \langle \partial_A(p \between q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_A(k_p)$, and $a \notin A$. For $E \Vdash \langle p \between q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$, we further distinguish two cases:

- Rule 25.2.l has been applied. Then, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$, and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 32.2, we can have $E \Vdash \langle \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k_p), \sigma'\rangle$. Also, we obtain $E \Vdash \langle \partial_A(q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 34. According to Rule 25.2.l, we have $E \Vdash \langle \partial_A(p) \between \partial_A(q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k_p), \sigma'\rangle$. Take $k_2 \equiv \partial_A(k_p)$, and observe that $(\partial_A(k_p), \partial_A(k_p)) \in R$.
- Rule 25.2.r has been applied. The proof is similar to the previous case.

Condition 4: We assume $E \Vdash \langle \partial_A(p \between q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle p \between q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ for some k_{pq} such that $k_1 \equiv \partial_A(k_{pq})$. For $E \Vdash \langle p \between q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, we further obtain (see also Rule 26) $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$, and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some k_p, k_q such that $k_{pq} \equiv k_p \between k_q$. Using Rule 33, we have $E \Vdash \langle \partial_A(p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \partial_A(k_p), \sigma'\rangle$, and $E \Vdash \langle \partial_A(q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \partial_A(k_q), \sigma'\rangle$. Applying Rule 26, we get $E \Vdash \langle \partial_A(p) \between \partial_A(q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \partial_A(k_p) \between \partial_A(k_q), \sigma'\rangle$. Note that $k_1 \equiv \partial_A(k_p \between k_q)$, take $k_2 \equiv \partial_A(k_p) \between \partial_A(k_q)$, and observe that $(k_1, k_2) \in R$.

Condition 6: First, we assume $E \Vdash \langle \partial_A(p \between q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle p \between q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 27, we obtain $E \Vdash$

$\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}$, and $E \Vdash \langle q,\sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34, we get $E \Vdash \langle \partial_A(p),\sigma\rangle \overset{\xi}{\rightsquigarrow}$, and $E \Vdash \langle \partial_A(q),\sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 27, we obtain $E \Vdash \langle \partial_A(p) \between \partial_A(q),\sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.7.5 *For arbitrary closed process terms p, q, set of action A, we have*

$$\partial_A(p;\ q) \leftrightarrows \partial_A(p);\ \partial_A(q).$$

PROOF. Let $R = \{(\partial_A(p;\ q), \partial_A(p);\ \partial_A(q)) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of the left implication of conditions 1 and 6 are similar to the proofs of the right implication of conditions 1 and 6. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle \partial_A(p;\ q),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd,\sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd,\sigma'\rangle$, and we know that $a \notin A$. However, the termination transition $E \Vdash \langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd,\sigma'\rangle$ is not possible in our semantics. Hence, the left implication of condition 1 holds trivially.

Condition 2: We assume $E \Vdash \langle \partial_A(p;\ q),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1,\sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_{pq},\sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_A(k_{pq})$, and $a \notin A$. For $E \Vdash \langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_{pq},\sigma'\rangle$, we further distinguish two cases:

- Rule 16 has been applied. Then, we get $E \Vdash \langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd,\sigma'\rangle$, and $E \Vdash \langle q,\sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv q$. Applying Rule 32.1, we can have $E \Vdash \langle \partial_A(p),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd,\sigma'\rangle$. Also, we obtain $E \Vdash \langle \partial_A(q),\sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ using Rule 34. According to Rule 16, we have $E \Vdash \langle \partial_A(p);\ \partial_A(q),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(q),\sigma'\rangle$. Note that $k_1 \equiv \partial_A(q)$, and take $k_2 \equiv \partial_A(q)$, and observe that $(k_1, k_2) \in R$.

- Rule 17 has been applied. Then, we get $E \Vdash \langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k',\sigma'\rangle$, and $E \Vdash \langle q,\sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, for some k' such that $k_{pq} \equiv k';\ q$. Applying Rule 32.2, we can have $E \Vdash \langle \partial_A(p),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k'),\sigma'\rangle$. According to Rule 17, we have $E \Vdash \langle \partial_A(p);\ \partial_A(q),\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\ \partial_A(k')\ ;\ \partial_A(q),\sigma'\ \rangle$. Note that $k_1 \equiv \partial_A(k';\ q)$, $k_2 \equiv \partial_A(k');\ \partial_A(q)$, and observe that $(k_1, k_2) \in R$.

Condition 4: The proof of this condition is similar to the proof of condition 2 in which Rule 17 has been applied in the case distinction.

Condition 6: First, we assume $E \Vdash \langle \partial_A(p;\ q),\sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle p;\ q,\sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 19, we obtain $E \Vdash \langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34, we get $E \Vdash \langle \partial_A(p),\sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 19, we obtain $E \Vdash \langle \partial_A(p);\ \partial_A(q),\sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Lemma B.7.6 *For arbitrary predicate u and set of actions A, we have*

$$\partial_A(u) \leftrightarrows u.$$

PROOF. Let $R = \{(\partial_A(u), u) \mid \text{predicate } u\}$. Since there are no termination and action transition rules defined for u, also indicates there are no termination and action transition rules defined for $\partial_A(u)$, the conditions 1 – 3 hold trivially. Moreover, action encapsulation has no effect in time transitions and consistency. Hence, the conditions 4 – 6 also hold trivially.

Lemma B.7.7 *For arbitrary closed process term p, set of actions A we have*

$$\partial_A([p]) \leftrightarrow [\partial_A(p)].$$

PROOF. Let $R = \{(\partial_A([p]), [\partial_A(p)]) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similar to the proof of the right implication of condition 1. The proof of condition 3 is similar to the proof of condition 2. We know that action encapsulation has no effect in time transitions and consistency. So, $\partial_A([p])$ also allows arbitrary time transitions and thereby do not change (because $[p]$ allows arbitrary time transitions, see Rule 12). $[p]$ (also $\partial_A([p])$, see Rule 34) and $[\partial_A(p)]$ are consistent with any extended valuation with respect to σ in any environment. It is not hard to see that the conditions 4 – 6 hold trivially

Condition 1: First, we assume $E \Vdash \langle \partial_A([p]), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle [p], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, and we know that $a \notin A$. Followed by Rule 10.1, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. Using Rule 32.1, we can obtain $E \Vdash \langle \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. According to Rule 10.1, we get $E \Vdash \langle [\partial_A(p)], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle \partial_A([p]), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle [p], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some k_p such that $k_1 \equiv \partial_A(k_p)$, and $a \notin A$. Followed by Rule 10.2, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$. Using Rule 32.2, we can obtain $E \Vdash \langle \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k_p), \sigma'\rangle$. According to Rule 10.2, we get $E \Vdash \langle [\partial_A(p)], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k_p), \sigma'\rangle$. Take $k_2 \equiv \partial_A(k_p)$, and observe that $(k_1, k_2) \in R$.

Lemma B.7.8 *For arbitrary closed process term p, guard b, and set of actions A we have*

$$\partial_A(b \to p) \leftrightarrow b \to \partial_A(p).$$

PROOF. Let $R = \{(\partial_A(b \to p), b \to \partial_A(p)) \mid p \in P\}$. The proofs of the left implication of conditions 1 and 6 are similarly straightforward to the proofs of the right implication of conditions 1 and 6. The proofs of conditions 3 and 5 are similarly straightforward to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle \partial_A(b \to p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some E, σ, ξ, a, ξ',σ', which means Rule 32.1 has been applied necessarily. Then, we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $a \notin A$. By Rule 20.1, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $\xi \models b$. Using

Rule 32.1, we obtain $E \Vdash \langle \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Followed by Rule 20.1, we have $E \Vdash \langle b \to \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

Condition 2: We assume $E \Vdash \langle \partial_A(b \to p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', k_1, which means Rule 32.2 has been applied necessarily. Then, we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ such that $k_1 \equiv \partial_A(k_1')$ and $a \notin A$. Using Rule 32, we obtain $E \Vdash \langle \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k_1'), \sigma'\rangle$. Followed by Rule 20.2, we have $E \Vdash \langle b \to \partial_A(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_A(k_1'), \sigma'\rangle$. Take $k_2 \equiv \partial_A(k_1')$ and observe that $(k_1, k_2) \in R$.

Condition 4: We assume $E \Vdash \langle \partial_A(b \to p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$ for some k_1' such that $k_1 \equiv \partial_A(k_1')$. For $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$, we can distinguish two cases:

- Rule 21 has been applied. Then $\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1'', \sigma'\rangle$ such that $k_1' \equiv b \to k_1''$, and $\forall_{s\in[0,t]}\, \rho(s) \models b$. Using Rule 33, we have $E \Vdash \langle \partial_A(p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \partial_A(k_1''), \sigma'\rangle$. Followed by Rule 21, we get $E \Vdash \langle b \to \partial_A(p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to \partial_A(k_1''), \sigma'\rangle$. Take $k_2 \equiv b \to \partial_A(k_1'')$ and observe that $(k_1, k_2) \in R$.
- Rule 22 has been applied. Then $\rho \in \Omega_{\sigma E t}$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b$, $\exists_{s\in[0,t]}\, \rho(s) \models \neg b$ $\rho(0) \models b \Rightarrow \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma'\rangle$ for some p', $\rho(t) \models b \Rightarrow \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, $k_1' \equiv b \to p$, and $\sigma' = \rho_\sigma(t)$. It is not hard to see that we have $\rho(0) \models b \Rightarrow \langle \partial_A(p), \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle \partial_A(p'), \sigma'\rangle$ and $\rho(t) \models b \Rightarrow \langle \partial_A(p), \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ using Rules 33 and 34. Applying Rule 22, we get $E \Vdash \langle b \to \partial_A(p), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to \partial_A(p), \sigma'\rangle$. Take $k_2 \equiv b \to \partial_A(p)$ and observe that $(k_1, k_2) \in R$.

Condition 6: First, we assume $E \Vdash \langle \partial_A(b \to p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 23 has been applied necessarily. Then, $E \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, we can distinguish two cases:

- Rule 23 has been applied. Then, we get $\xi \models b$ and $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 34, we get $E \Vdash \langle \partial_A(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Followed by Rule 23, we conclude that $E \Vdash \langle b \to \partial_A(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.
- Rule 24 has been applied. Then, we get $\sigma \cup \xi^{\dot{C}L} \models \neg b$ for some $\xi^{\dot{C}L}$. Using Rule 24, we obtain $E \Vdash \langle b \to \partial_A(p), \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

B.8 Inconsistent process

The following lemmas prove the properties of Proposition 3.5.8.

Lemma B.8.1 *For arbitrary predicate u we have,*

$$u \curvearrowright \bot \underline{\leftrightarrow} \bot.$$

PROOF. Using the previous properties we have $u \curvearrowright \bot \leftrightarrow u \curvearrowright (\text{false} \curvearrowright p) \leftrightarrow (u \wedge \text{false}) \curvearrowright p \leftrightarrow \text{false} \curvearrowright p \leftrightarrow \bot$.

Lemma B.8.2 *For arbitrary closed term p we have*

$$p \,[\!]\, \bot \leftrightarrow \bot.$$

PROOF. Since there are no transition rules defined for $\bot$, also note that $p \,[\!]\, \bot$ has no transitions, the conditions 1 – 6 hold trivially.

Lemma B.8.3 *For arbitrary closed process term p we have*

$$p \parallel \bot \leftrightarrow \bot.$$

PROOF. Since there are no transition rules defined for $\bot$, also note that $p \parallel \bot$ has no transitions, the conditions 1 – 6 hold trivially.

Lemma B.8.4 *For arbitrary set of actions A we have*

$$\partial_A(\bot) \leftrightarrow \bot.$$

PROOF. Since there are no transition rules defined for $\bot$, also note that $\partial_A(\bot)$ has no transitions, the conditions 1 – 6 hold trivially.

Lemma B.8.5 *For arbitrary closed process term p we have*

$$\bot;\, p \leftrightarrow \bot.$$

PROOF. Since there are no transition rules defined for $\bot$, also note that $\bot;\, p$ has no transitions, the conditions 1 – 6 hold trivially.

Lemma B.8.6 *We have*

$$\text{skip};\, \bot \leftrightarrow \delta.$$

PROOF. We know that skip $\equiv \emptyset : \text{true} \gg \tau$. Let $R = \{(\emptyset : \text{true} \gg \tau;\, \bot, \delta)\}$. Since there are no action transitions and time transitions defined for δ and $\bot$, also $\emptyset : \text{true} \gg \tau;\, \bot$ cannot perform any action transitions (because $\bot$ is not consistent) and time transitions (because no time transitions are defined for $\emptyset : \text{true} \gg \tau$), the conditions 1 – 5 hold trivially. *Condition 6*: First, we assume $(C, J, L, H, R) \Vdash \langle \emptyset : \text{true} \gg \tau;\, \bot, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some C,J, L, H, R, σ, ξ, which means that Rule 19 has been applied necessarily. Then, $(C, J, L, H, R) \Vdash \langle \emptyset : \text{true} \gg \tau, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ such that $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ (see also Rule 2). Then, we get $(C, J, L, H, R) \Vdash \langle \delta, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}$ using Rule 9. Second, we assume $(C, J, L, H, R) \Vdash \langle \delta, \sigma \rangle \overset{\xi}{\rightsquigarrow}$, which means that Rule 9 has been applied necessarily. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$.

Using Rule 2, we get $(C, J, L, H, R) \Vdash \langle \emptyset : \text{true} \gg \tau, \sigma \rangle \overset{\sigma \cup \xi \dot{C} L}{\rightsquigarrow}$. According to Rule 19, we obtain $(C, J, L, H, R) \Vdash \langle \emptyset : \text{true} \gg \tau;\ \bot, \sigma \rangle \overset{\sigma \cup \xi \dot{C} L}{\rightsquigarrow}$.

Lemma B.8.7 *We have,*

$$\bot \underline{\leftrightarrow} \text{ false}.$$

PROOF. Using the previous properties we have $\bot \underline{\leftrightarrow} \text{ false} \curvearrowright \text{false} \underline{\leftrightarrow} \text{ false}$.

APPENDIX

C

Proofs of the translation from Chi to Hybrid Automata

C.1 The semantics of the repetition operator

In Section 2.5.2, the semantics of repetition is given as an expression of the repetition operator in terms of other language elements. For the proofs in this thesis, this is rather inconvenient as several language constructs are used in this expression that play no role in this thesis. Therefore, in this appendix, we provide deduction rules for the repetition operator in the restricted setting of this thesis. It can be shown that these coincide with the formal semantics, though we omit the proof here.

$$\frac{\langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E\rangle}{\langle *p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix}*p\\ p';\ *p\end{smallmatrix}, \sigma', E\rangle}(\mathrm{A}) \qquad \frac{\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle}{\langle *p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p';\ *p, \sigma', E\rangle}(\mathrm{B})$$

$$\frac{\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}}{\langle *p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}}(\mathrm{C})$$

C.2 Proof of Theorem 5.3.1

Let p be a closed process term, v_0 be the initial location of $\mathcal{T}_J(p)$, α and σ be valuations such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$, ξ be an extended valuation such that $\sigma = \xi \restriction \mathrm{dom}(\sigma)$ and $E = (C, J, \emptyset, H, \emptyset)$ be an environment. Then

$$(v_0, \alpha) \textit{ is an admissible state of } [\![\, \mathcal{HA}(\langle p, \sigma, E\rangle) \,]\!] \quad \Leftrightarrow \quad \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} .$$

PROOF. We prove this theorem by induction on the structure of p. Firstly, we give the proof for (v_0, α) is an admissible state of $[\![\, \mathcal{HA}(\langle p, \sigma, E\rangle) \,]\!] \quad \Rightarrow \quad \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Guarded action predicate $p \equiv b \rightarrow W : r \gg l_{\mathrm{a}}$ for some b, W, r, l_{a}. We assume that (v_0, α) is an admissible state of $[\![\mathcal{HA}(\langle b \rightarrow W : r \gg l_{\mathrm{a}}, \sigma, E\rangle)]\!]$. According to translation defined for the guarded action predicate, $\mathcal{T}_J(b \rightarrow W : r \gg l_{\mathrm{a}})$ has 2 locations source v_0 and target v_0'. They are connected by an urgent edge e, which is guarded by b with label l_{a} and jump condition $(W \cup J \cup X_{\mathrm{aux}}, \zeta_{W\cup J}(r))$. Then, we can distinguish two cases:

  - $\mathrm{source}(e) = v_0, \mathrm{target}(e) = v_0', \mathrm{event}(e) = l_{\mathrm{a}}, \alpha \models b, (\alpha, \alpha') \models \mathrm{jump}(e)$ for some α'. According to the semantics of a hybrid automaton, we know that there exists $(v_0, \alpha) \xrightarrow{l_{\mathrm{a}}} (v_0', \alpha')$. By Theorem 5.3.5, Lemma 3.5.2, $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma = \xi \upharpoonright \mathrm{dom}(\sigma)$, it is not hard to see that $\langle W : r \gg l_{\mathrm{a}}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. Using Rule 23, we have $\langle b \rightarrow W : r \gg l_{\mathrm{a}}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.
  - $\alpha \models \neg b$. From $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma = \xi \upharpoonright \mathrm{dom}(\sigma)$, it is not hard to see that we have $\sigma \cup \xi^{\dot{C}L} \models \neg b$ for some $\xi^{\dot{C}L}$. Using Rule 24, we get $(C, J, L, H, R) \Vdash \langle b \rightarrow W : r \gg l_{\mathrm{a}}, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

- Guarded send $p \equiv b \rightarrow h\,!!\,\mathbf{e}_n$ for some $b, h, \mathbf{e}_n$. The proof is similar to the proof of the case that $p \equiv b \rightarrow W : r \gg l_{\mathrm{a}}$.

- Guarded receive $p \equiv b \rightarrow h\,??\,\mathbf{x}_n$ for some $b, h, \mathbf{x}_n$. The proof is similar to the proof of the case that $p \equiv b \rightarrow W : r \gg l_{\mathrm{a}}$.

- Delay predicate $p \equiv u$ for some u. We assume that (v_0, α) is an admissible state of $[\![\mathcal{HA}(\langle u, \sigma, E\rangle)]\!]$. According to the translation defined for the delay predicate and the semantics of a hybrid automaton, we know that $\alpha \models \mathrm{inv}(v_0)$ and $\mathrm{inv}(v_0) = u[DC/\dot{C}]$. Since $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $DC = \{dc \mid c \in C\}$, it is not hard to see that $\sigma \cup \xi^{\dot{C}L} \models u$ for some $\xi^{\dot{C}L}$. Using Rule 4, we get $(C, J, \emptyset, H, \emptyset) \Vdash \langle u, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

- Any delay operator $p \equiv [q]$ for some q. The proof is trivial, because process term $[q]$ (for any $q \in P$) is consistent with any extended valuation with respect to σ in any environment.

- Sequential composition operator $p \equiv q_1;\ q_2$ for some q_1 and q_2. We have (v_0, α) is an admissible state of $\mathcal{HA}(\langle q_1; q_2, \sigma, E\rangle)$. According to translation defined for the sequential composition operator, (v_0, α) is also an admissible state of $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $\langle q_1;\ q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Alternative composition operator $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2. We assume (v_0, α) is an admissible state of $[\![\mathcal{HA}(\langle q_1 \,[\!]\, q_2, \sigma, E\rangle)]\!]$. According to translation defined for the alternative composition operator and the semantics of a hybrid automaton, we know that $\alpha \models \mathrm{inv}(v_0)$, $v_0 = (v_{q_1}, v_{q_2})$ and $\mathrm{inv}(v_0) = \mathrm{inv}(q_1) \wedge \mathrm{inv}(q_2)$, where v_{q_1} and v_{q_2} are the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$, respectively. It is not hard to see that

we also obtain $\alpha \models \mathrm{inv}(q_1)$ and $\alpha \models \mathrm{inv}(q_2)$. So, (v_{q_1}, α) and (v_{q_2}, α) are admissible states of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$, respectively. By induction, we then have $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we obtain $\langle q_1 ;\ q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. The proof is similar to the proof of the case that $p \equiv q_1 \mathbin{[\![]\!]} q_2$.

- Repetition operator $p \equiv *q$ for some q. We assume that (v_0, α) is an admissible state of $[\![\mathcal{HA}(\langle *q, \sigma, E\rangle)]\!]$. According to translation defined for the alternative composition operator, we know that v_0 is also the initial location of $\mathcal{T}_J(q)$. Since $\alpha \models \mathrm{inv}(v_0)$, (v_q, α) is an admissible states of $\mathcal{T}_J(q)$. By induction, we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule C for the repetition operator, we get $\langle *q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. The proof is trivial, becuase $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J \cup J^+}(q)$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. The proof is similar to the proof of the case that $p \equiv *q$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $H \subseteq \mathcal{H}$ and q. The proof is similar to the proof of the case that $p \equiv *q$.

Secondly, we give the proof for $\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \ \Rightarrow \ (v_0, \alpha)$ is an admissible state of $[\![\mathcal{HA}(\langle p, \sigma, E\rangle)]\!]$.

- Guarded action predicate $p \equiv b \to W : r \gg l_{\mathrm{a}}$ for some b, W, r, l_{a}. We assume $\langle b \to W : r \gg l_{\mathrm{a}}, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. According to the translation defined for the guarded action predicate, $\mathcal{T}_J(b \to W : r \gg l_{\mathrm{a}})$ has the initial location v_0 with invariant true. Since $\alpha \models \mathrm{true}$, according to the semantics of a hybrid automaton, (v_0, α) is an admissible state of $[\![\mathcal{HA}(\langle b \to W : r \gg l_{\mathrm{a}}, \sigma, E\rangle)]\!]$.

- Guarded send $p \equiv b \to h \,!!\, \mathbf{e}_n$ for some $b, h, \mathbf{e}_n$. The proof is similar to the proof of the case that $p \equiv b \to W : r \gg l_{\mathrm{a}}$.

- Guarded receive $p \equiv b \to h \,??\, \mathbf{x}_n$ for some $b, h, \mathbf{x}_n$. The proof is similar to the proof of the case that $p \equiv b \to W : r \gg l_{\mathrm{a}}$.

- Delay predicate $p \equiv u$ for some u. We assume $(C, J, \emptyset, H, \emptyset) \Vdash \langle u, \sigma\rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}$ for some $\xi^{\dot{C}L}$. By Rule 4, we get $\sigma \cup \xi^{\dot{C}L} \models u$. According to the translation defined for the delay predicate, the invariant of the initial location v_0 of $\mathcal{T}_J(u)$ has invariant $u[DC/\dot{C}]$. Since $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$ and $DC = \{dc \mid c \in C\}$, it is not hard to see that $\alpha \models u[DC/\dot{C}]$. So, (v_0, α) is an admissible state of $[\![\mathcal{HA}(\langle u, \sigma, E\rangle)]\!]$.

- Any delay operator $p \equiv [q]$ for some q. We assume $\langle [q], \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. According to the translation defined for the any delay operator, the initial location v_0 of $\mathcal{T}_J([q])$ has invariant true. We also know that $\alpha \models \text{true}$. According to the semantics of a hybrid automaton, (v_0, α) is an admissible state of $[\![\, \mathcal{HA}(\langle [q], \sigma, E\rangle) \,]\!]$.

- Sequential composition operator $p \equiv q_1;\ q_2$ for some q_1 and q_2. We assume that $\langle q_1;\ q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. From Rule 19, we know that $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction, we then have that (v_0, α) is an admissible state of $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!]$ for some v_0, where v_0 is the initial location of $\mathcal{T}_J(q_1)$. From the semantics of a hybrid automaton, we know $\alpha \models \text{inv}(v_0)$. According to the translation defined for the sequential composition operator, we know that v_0 is also the initial location of $\mathcal{T}_J(q_1;\ q_2)$. So, (v_0, α) is also an admissible state of $[\![\, \mathcal{HA}(\langle q_1;\ q_2, \sigma, E\rangle) \,]\!]$

- Alternative composition operator $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2. We assume that $\langle q_1 \,[\!]\, q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By Rule 27, we further obtain $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction, we then have that (v_{q_1}, α) is an admissible state of $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!]$ and (v_{q_2}, α) is an admissible state of $[\![\, \mathcal{HA}(\langle q_2, \sigma, E\rangle) \,]\!]$ for some v_{q_1} and v_{q_2}, where v_{q_1} and v_{q_2} are the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$, respectively. From the semantics of a hybrid automaton, we have $\alpha \models \text{inv}(v_{q_1})$ and $\alpha \models \text{inv}(v_{q_2})$. According to the translation defined for the alternative composition operator, we know that $\text{inv}(v_0) = \text{inv}(v_{q_1}) \wedge \text{inv}(v_{q_2})$, where v_0 is the initial location of $\mathcal{T}_J(q_1 \,[\!]\, q_2)$. It is not hard to see that $\alpha \models \text{inv}(v_0)$. So, (v_{q_0}, α) is an admissible state of $[\![\, \mathcal{HA}(\langle q_1 \,[\!]\, q_2, \sigma, E\rangle) \,]\!]$.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. The proof is similar to the proof of the case that $p \equiv q_1 \,[\!]\, q_2$.

- Repetition operator $p \equiv *q$ for some q. We assume that $\langle *q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By Rule C of the repetition operator, we obtain $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction, we then have that (v_0, α) is an admissible state of $[\![\, \mathcal{HA}(\langle q, \sigma, E\rangle) \,]\!]$ for some v_0, where v_0 is the initial location of $\mathcal{T}_J(q)$. From the semantics of a hybrid automaton, we know that $\alpha \models \text{inv}(v_0)$. According to the translation defined for the repetition operator, v_0 is also the initial location of $\mathcal{T}_J(*q)$. So, (v_0, α) is an admissible state of $[\![\, \mathcal{HA}(\langle *q, \sigma, E\rangle) \,]\!]$.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. The proof is trivial, becuase $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J \cup J^+}(q)$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. The proof is similar to the proof of the case that $p \equiv *q$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $H \subseteq \mathcal{H}$ and q. The proof is similar to the proof of the case that $p \equiv *q$.

C.3 Proof of Theorem 5.3.4

C.3.1 Theorem 5.3.4.1 – part 1

The proof is by induction on the structure of closed process term p. Since there are no termination transition rules (with a as specified) defined for delay predicates, consistent deadlock δ, guarded inconsistent process term $b \rightarrow \bot$, sequential composition, parallel composition and the repetition operator, the theorem holds trivially for these cases.

For the guarded send and receive process terms there cannot be a termination transition not involving a 'communication' label, therefore, the theorem holds trivially for these cases.

- Guarded action predicate $p \equiv b \rightarrow W : r \gg l_{\mathrm{a}}$ for some b, W, r, l_{a}. We have $\langle b \rightarrow W : r \gg l_{\mathrm{a}}, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle \checkmark, \sigma', E\rangle$, which means Rule 20.1 has been applied necessarily. Then, $\langle W : r \gg l_{\mathrm{a}}, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$ and $\xi \models b$. In turn, Rule 1 has been applied necessarily, so $a = l_{\mathrm{a}}$, $\xi'_\sigma = \sigma'$, $\xi = \sigma \cup \xi^{\dot{C}L}$ such that $\xi^{\dot{C}L} \in \dot{C} \rightarrow \Lambda$, $\xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C},\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\, \xi(x) = \sigma(x)\}$, and $\xi^- \cup \xi' \models r$. According to the translation defined for the guarded action predicate, $\mathcal{T}_J(b \rightarrow W : r \gg l_{\mathrm{a}})$ has the initial location v_0 and the terminating location v'_0, both with invariant true, that are connected by an urgent edge e, guarded by predicate b, with jump condition $(W \cup J \cup X_{\mathrm{aux}}, \zeta_{W \cup J}(r))$ and labelled with event l_{a}. Since $\xi \models b$ and $\xi \upharpoonright \mathrm{dom}(\sigma) = \sigma$ and the variables outside $\mathrm{dom}(\sigma)$ are not allowed to occur in b, we also have $\alpha \models \mathrm{guard}(e)$. From $\xi^- \cup \xi' \models r$, we have $(\alpha, \alpha') \models \mathrm{jump}(e)$ for arbitrary α and α' such that $\alpha \upharpoonright \mathrm{dom}(\sigma) = \sigma$ and $\alpha' \upharpoonright \mathrm{dom}(\sigma') = \sigma'$ since the variables from $\mathrm{dom}(\alpha) \setminus \mathrm{dom}(\sigma)$ do not occur in predicate $\mathrm{jump}(e)$. Therefore, $(v_0, \alpha) \xrightarrow{a} (v'_0, \alpha')$ is in $[\![\, \mathcal{HA}(\langle b \rightarrow W : r \gg l_{\mathrm{a}}, \sigma, E\rangle) \,]\!]$ for arbitrary α and α' such that $\alpha' \upharpoonright \mathrm{dom}(\sigma') = \sigma'$. Note that the states (v_0, α) and (v'_0, α') are admissible since the locations v_0 and v'_0 have invariant true.

- Any delay operator $p \equiv [q]$ for some q. We have $\langle [q], \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle \checkmark, \sigma', E\rangle$, which means Rule 10.1 has been applied necessarily. We have $\langle q, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle \checkmark, \sigma', E\rangle$. By induction we then have $[\![\, \mathcal{HA}(\langle q, \sigma, E\rangle) \,]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q and v'_q such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location and a terminating location of $\mathcal{T}_J(q)$. According to the translation defined for the any delay operator, v_q and v'_q are a location and a terminating location of $\mathcal{T}_J([q])$ respectively. Moreover, all possible termination transitions in $[\![\, \mathcal{HA}(\langle q, \sigma, E\rangle) \,]\!]$ are preserved in $[\![\, \mathcal{HA}(\langle [q], \sigma, E\rangle) \,]\!]$, because all edges in $\mathcal{T}_J(q)$ are still present in $\mathcal{T}_J([q])$. Furthermore, an additional initial location (v''_q) of $\mathcal{T}_J([q])$ is introduced into $\mathcal{T}_J(q)$ to obtain $\mathcal{T}_J([q])$. The invariant and flow condition of v''_q are true. Also, all outgoing edges of the initial location of $\mathcal{T}_J(q)$ are copied to the initial location of $\mathcal{T}_J([q])$ (i.e. v''_q) with original targets. Obviously, the action transition $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is in $[\![\, \mathcal{HA}(\langle [q], \sigma, E\rangle) \,]\!]$.

- Alternative composition operator $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2. We have $\langle q_1 \,[\!]\, q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$, which means Rule 25.1.l or 25.1.r has been applied necessarily. Since

the proofs for both cases are similar, we only give the proofs for the case that Rule 25.1.l has been applied. Then, we get $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$. By induction we then have $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ for some v_{q_1} and v'_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v'_{q_1} denote the initial location and a terminating location of $\mathcal{T}_J(q_1)$. According to the translation defined for the alternative composition operator, all outgoing edges for v_{q_1} of $\mathcal{T}_J(q_1)$ are copied to new initial location $v_0 = (v_{q_1}, v_{q_2})$ of $\mathcal{T}_J(q_1 \,[\!]\, q_2)$ with the original targets, where v_{q_2} denotes the initial location of $\mathcal{T}_J(q_2)$. Hence, $(v_0, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ is in $[\![\mathcal{HA}(\langle q_1 \,[\!]\, q_2, \sigma, E\rangle)]\!]$. Observe that v'_{q_1} is also a terminating location in $\mathcal{T}_J(q_1 \,[\!]\, q_2)$.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. We have $\langle \iota_{J^+}(q), \sigma,$ $(C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', (C, J, \emptyset, H, \emptyset)\rangle$, which means Rule 41.1 has been applied necessarily. Then, we get $\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', (C, J \cup J^+, \emptyset, H, \emptyset)\rangle$. By induction, we then have $[\![\mathcal{HA}(\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q, v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location and a terminating location of $\mathcal{T}_{J\cup J^+}(q)$. According to the translation defined for the jump enabling operator, $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J\cup J^+}(q)$, $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \iota_{J^+}(q),$ $\sigma, (C, J, \emptyset, H, \emptyset)\rangle)]\!]$, and v_q and v'_q are the initial and a terminating location of $\mathcal{T}_{J\cup J^+}(q)$ respectively.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. We have $\langle \partial_A(q),\ \sigma, E\rangle$ $\xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$, which means Rule 32.1 has been applied necessarily. Then, $\langle q, \sigma, E\rangle$ $\xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$ and $a \notin A$. By induction, we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha)$ $\xrightarrow{a}$ (v'_q, α') for some v_q and v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location and a terminating location of $\mathcal{T}_J(q)$. According to the translation defined for action encapsulation operator, v_q and v'_q are also the initial location and a terminating location of $\mathcal{T}_J(\partial_A(q))$. We obtain $\mathcal{T}_J(\partial_A(q))$ by replacing the jump conditions of edges labelled with events from A of $\mathcal{T}_J(q)$ with predicates false with an empty set of variables that are allowed to change. Since $a \notin A$, $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \partial_A(q), \sigma, E\rangle)]\!]$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $H \subseteq \mathcal{H}$ and q. We have $\langle \upsilon_H(q),$ $\sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ $\langle \checkmark, \sigma', E\rangle$, which means Rule 35.1 has been applied necessarily. Then, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$. By induction we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha)$ $\xrightarrow{a}$ (v'_q, α') for some v_q, v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location and a terminating location of $\mathcal{T}_J(q)$. According to the translation defined for urgent communication operator, v_q is also the initial location of $\mathcal{T}_J(\upsilon_H(q))$. Moreover, all possible termination transitions in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$ are preserved in $[\![\mathcal{HA}(\langle \upsilon_H(q), \sigma, E\rangle)]\!]$, because $\mathcal{T}_J(q)$ and $\mathcal{T}_J(\upsilon_H(q))$ have the same edges. The fact that some of these edges have become urgent ones, only has effect on the possibility of time transitions. Hence, $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is obviously in $[\![\mathcal{HA}(\langle \upsilon_H(q), \sigma, E\rangle)]\!]$.

C.3.2 Theorem 5.3.4.1 – part 2

The proof is by induction on the structure of closed process term p. Since there are no action transition rules (with a as specified) defined for guarded action predicates, delay predicates, consistent deadlock δ, inconsistent process term $b \rightarrow \bot$, guarded send process terms and guarded receive process terms, the theorem holds trivially for these cases.

- Any delay operator $p \equiv [q]$ for some q. We have $\langle [q], \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle p', \sigma', E\rangle$, which means Rule 10.2 has been applied necessarily. We have $\langle q, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle p', \sigma', E\rangle$. By induction we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q and v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial locations of $\mathcal{T}_J(q)$ and $\mathcal{T}_J(p')$. According to the translation defined for the any delay operator, all edges in $\mathcal{T}_J(q)$ are still present in $\mathcal{T}_J([q])$. Moreover, an additional initial location (v''_q) of $\mathcal{T}_J([q])$ is introduced into $\mathcal{T}_J(q)$ to obtain $\mathcal{T}_J([q])$. The invariant and flow condition of v''_q are true. Also, all outgoing edges of the initial location of $\mathcal{T}_J(q)$ are copied to the initial location of $\mathcal{T}_J([q])$ (i.e. v''_q) with original targets. Therefore, all possible action transitions in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$ are preserved in $[\![\mathcal{HA}(\langle [q], \sigma, E\rangle)]\!]$. Obviously, the action transition $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is in $[\![\mathcal{HA}(\langle [q], \sigma, E\rangle)]\!]$.

- Sequential composition operator $p \equiv q_1;\ q_2$ for some q_1 and q_2. We have $\langle q_1;\ q_2, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle p', \sigma', E\rangle$, which means Rule 16 or Rule 17 has been applied necessarily.

  - Rule 16 has been applied. Then $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle \checkmark, \sigma', E\rangle$ and $p' \equiv q_2$. By Theorem 5.3.4.1 we then have $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!] \models (v_{q_1}, \sigma) \xrightarrow{a} (v'_{q_1}, \sigma')$ for some v_{q_1}, v'_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v'_{q_1} denote the initial location and a terminating location of $\mathcal{T}_J(q_1)$. The edge that causes this transition is still also present in $\mathcal{T}_J(q_1;\ q_2)$, but now ends in the initial node of $\mathcal{T}_J(q_2)$. Therefore, $[\![\mathcal{HA}(\langle q_1;\ q_2, \sigma, E\rangle)]\!] \models (v_{q_1}, \sigma) \xrightarrow{a} (v_{q_2}, \sigma')$ for some v_{q_1}, v_{q_2} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v_{q_2} denote the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$ respectively.

  - Rule 17 has been applied. Then $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle q'_1, \sigma', E\rangle$ and $p' \equiv q'_1;\ q_2$. By induction, we then have $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!] \models (v_{q_1}, \sigma) \xrightarrow{a} (v'_{q_1}, \sigma')$ for some v_{q_1}, v'_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v'_{q_1} denote the initial location of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q'_1)$. Since v'_{q_1} is not a terminating location, referring to the translation defined for the sequential composition operator, we have that the edge between v_{q_1} and v'_{q_1} is still present in $\mathcal{HA}(\langle q_1;\ q_2, \sigma, E\rangle)$. Hence, the action transition $(v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ is preserved in $[\![\mathcal{HA}(\langle q_1; q_2, \sigma, E\rangle)]\!]$. Also v_{q_1} is the initial location of $\mathcal{T}_J(q_1;\ q_2)$ and v'_{q_1} is the initial location of $\mathcal{T}_J(q'_1;\ q_2)$.

- Alternative composition operator $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2. We have $\langle q_1 \,[\!]\, q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E\rangle$, which means Rule 25.2.l or 25.2.r has been applied necessarily. Since the proofs for both cases are similar, we only give the proofs for the case that Rule

25.2.l has been applied. Then, we get $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E\rangle$ for some q_1' and $p' \equiv q_1'$. By induction we then have $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ for some v_{q_1} and v'_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v'_{q_1} denote the initial location of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_1')$. According to the translation defined for the alternative composition operator, all outgoing edges for v_{q_1} of $\mathcal{T}_J(q_1)$ are copied to new initial location $v_0 = (v_{q_1}, v_{q_2})$ of $\mathcal{T}_J(q_1 \,[\!]\, q_2)$ with the original targets, where v_{q_2} denotes the initial location of $\mathcal{T}_J(q_2)$. Hence, $(v_0, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ is in $[\![\, \mathcal{HA}(\langle q_1 \,[\!]\, q_2, \sigma, E\rangle) \,]\!]$.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $\langle q_1 \parallel q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E\rangle$, which means Rule 29.1.l, Rule 29.1.r, Rule 29.2.l or Rule 29.2.r has been applied necessarily. Since the proofs for the first and second case and the proofs for the third and fourth case are similar, we only give the proofs for the first and third case.

  - Rule 29.1.l has been applied. Then, we have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark,\ \sigma', E\rangle$, $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$ and $p' \equiv q_2$. By part 1 of Theorem 5.3.4.1, we have $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ for some v_{q_1} and v'_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v'_{q_1} denote the initial location and a terminating location of $\mathcal{T}_J(q_1)$ respectively. This means that there exists an edge, say e_{q_1} with source location v_{q_1}, target location v'_{q_1}, event a, and a guard and a jump condition that hold for α and α'. From the translation, it then follows that (v_{q_1}, v_{q_2}), where v_{q_2} is the initial location of $\mathcal{T}_J(q_2)$, is the initial location of $\mathcal{T}_J(q_1 \parallel q_2)$. Also, there is an edge (e_{q_1}, v_{q_2}) in $\mathcal{T}_J(q_1 \parallel q_2)$, with source location (v_{q_1}, v_{q_2}), target location v_{q_2}, and the same event, guard and jump condition as edge e_{q_1} in $\mathcal{T}_J(q_1)$. Therefore, $[\![\, \mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle) \,]\!] \models ((v_{q_1}, v_{q_2}), \alpha) \xrightarrow{a} (v_{q_2}, \alpha')$ with v_{q_2} the initial location of $\mathcal{T}_J(q_2)$, hence of $\mathcal{T}_J(p')$.

  - Rule 29.2.l has been applied. Then, we have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E\rangle$ for some q_1', and $p' = q_1' \parallel q_2$. By induction we then have $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ for some v_{q_1} and v'_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} and v'_{q_1} denote the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_1')$, respectively. We also know that there exists an edge, say e_{q_1} in $\mathcal{HA}(\langle q_1, \sigma, E\rangle)$ with source location v_{q_1}, target location v'_{q_1}, and event a. Since location v'_{q_1} is a non-terminating location, according to the translation defined for the parallel composition operator, there is an edge (e_{q_1}, v_{q_2}) in $\mathcal{T}_J(q_1 \parallel q_2)$, with v_{q_2} the initial location of $\mathcal{T}_J(q_2)$, with source location (v_{q_1}, v_{q_2}), target location (v'_{q_1}, v_{q_2}), event a, the guard and jump conditions as edge e_{q_1} in $\mathcal{T}_J(q_1)$ and with the same urgency status as edge e_{q_1} in $\mathcal{T}_J(q_1)$. Hence, $((v_{q_1}, v_{q_2}), \alpha) \xrightarrow{a} ((v'_{q_1}, v_{q_2}), \alpha')$ is in $[\![\, \mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle) \,]\!]$. Note that (v'_{q_1}, v_{q_2}) is the initial location of $\mathcal{T}_J(p' \parallel q_2)$.

- Repetition operator $p \equiv *q$ for some q. We have $\langle *q, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle p', \sigma', E\rangle$, which means Rule A.1 or A.2 from Appendix C.1 has been applied necessarily.

– Rule A.1 has been applied. Then, we have $\langle q, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle \checkmark, \sigma', E\rangle$ and $p' \equiv *q$. By part 1 of Theorem 5.3.4.1, we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q, v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location and a terminating location of $\mathcal{T}_J(q)$, respectively. We also know that there exists an edge e with source location v_q, target location v'_q, event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for the repetition operator, v_q is also the initial location of $\mathcal{T}_J(*q)$. Since v'_q is a terminating location, the target location of e is relocated to v_q. Neverthelese, the edge e from $\mathcal{T}_J(q)$ is still in $\mathcal{T}_J(*q)$, so the action transition $(v_q, \alpha) \xrightarrow{a} (v_q, \alpha')$ is in $[\![\mathcal{HA}(\langle *q, \sigma, E\rangle)]\!]$.

– Rule A.2 has been applied. Then, we have $\langle q, \sigma, E\rangle \xrightarrow{\xi,\ a,\ \xi'} \langle q', \sigma', E\rangle$ for some q', and $p' \equiv q'; *q$. By induction we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q, v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location of $\mathcal{T}_J(q)$ and $\mathcal{T}_J(q')$. We also know that there exists an edge e with source location v_q, target location v'_q, event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for the repetition operator, v_q and v'_q are also the initial location and a location of $\mathcal{T}_J(*q)$. Since v'_q is not a terminating location and the edge e from $\mathcal{T}_J(q)$ is also in $\mathcal{T}_J(*q)$, the action transition $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is obviously in $[\![\mathcal{HA}(\langle *q, \sigma, E\rangle)]\!]$.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. We have $\langle \iota_{J^+}(q), \sigma, (C,\ J, \emptyset,\ H, \emptyset)\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', (C, J, \emptyset, H, \emptyset)\rangle$, which means Rule 41.2 has been applied necessarily. Then, we get $\langle q,\ \sigma,\ (C,\ J \cup J^+, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi,a,\xi'} \langle q',\ \sigma',\ (C,\ J \cup J^+, \emptyset, H, \emptyset)\rangle$ for some q' and $p' \equiv \iota_{J^+}(q')$. By induction we then have $[\![\ \mathcal{HA}(\ \langle\ q,\ \sigma,\ (C,\ J \cup J^+,\ \emptyset,\ H,\ \emptyset)\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q, v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial location of $\mathcal{T}_{J\cup J^+}(q)$ and $\mathcal{T}_{J\cup J^+}(q')$. According to the translation defined for the jump enabling operator, we have $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J\cup J^+}(q)$, v_q and v'_q are also the initial locations of $\mathcal{T}_J(\iota_{J^+}(q))$ and $\mathcal{T}_J(\iota_{J^+}(q'))$. Obviously, $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \iota_{J^+}(q), \sigma, (C, J, \emptyset, H, \emptyset)\rangle)]\!]$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. We have $\langle \partial_A(q),\ \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E\rangle$, which means Rule 32.2 has been applied necessarily. Then, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E\rangle$ for some q' such that $p' \equiv \partial_A(q')$, and $a \notin A$. By induction, we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q and v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial locations of $\mathcal{T}_J(q)$ and $\mathcal{T}_J(q')$, respectively. According to the translation defined for action encapsulation operator, v_q and v'_q are also the initial locations of $\mathcal{T}_J(\partial_A(q))$ and $\mathcal{T}_J(\partial_A(q'))$. We obtain $\mathcal{T}_J(\partial_A(q))$ from $\mathcal{T}_J(q)$ by replacing the jump conditions of edges labelled with events from A of $\mathcal{T}_J(q)$ with predicates false with an empty set of variables that are allowed to change. Since $a \notin A$, $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \partial_A(q), \sigma, E\rangle)]\!]$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $H \subseteq \mathcal{H}$ and q. We have $\langle \upsilon_H(q),\sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E\rangle$, which means Rule 35.2 has been applied necessarily. Then, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E\rangle$ for some q' such that $p' \equiv \partial_{\mathcal{A}}(q')$. By induction we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models (v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ for some v_q, v'_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q and v'_q denote the initial locations of $\mathcal{T}_J(q)$ and $\mathcal{T}_J(q')$, respectively. According to the translation defined for urgent communication operator, v_q and v'_q are also the initial locations of $\mathcal{T}_J(\upsilon_H(q))$ and $\mathcal{T}_J(\upsilon_H(q'))$, respectively. Moreover, all possible action transitions in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$ are preserved in $[\![\mathcal{HA}(\langle \upsilon_H(q), \sigma, E\rangle)]\!]$, because $\mathcal{T}_J(q)$ and $\mathcal{T}_J(\upsilon_H(q))$ have the same edges. The fact that some of these edges have become urgent ones, only has effect on the possibility of time transitions. Hence, $(v_q, \alpha) \xrightarrow{a} (v'_q, \alpha')$ is obviously in $[\![\mathcal{HA}(\langle \upsilon_H(q), \sigma, E\rangle)]\!]$.

C.3.3 Theorem 5.3.4.2 – part 1

The proof is by induction on the structure of closed process term p. Since there are no termination transition rules (with $\mathrm{isa}(h, cs)$ as a label) defined for delay predicate, consistent deadlock δ, inconsistent process term $b \to \bot$, guarded action predicate, guarded receive process term, sequential composition, parallel composition and the repetition operator, the theorem holds trivially for these cases.

The proofs for the any delay operator, the alternative composition operator, the jump enabling operator, the action encapsulation operator, and the urgent communication operator are similar to the proofs for these operators in the proof of the first part of Theorem 5.3.4.1 in Appendix C.3.1 since these operators treat send actions similarly as normal actions.

- Guarded send $p \equiv b \to h\,!!\,\mathbf{e}_n$ for some $b, h, \mathbf{e}_n$. We have $\langle b \to h\,!!\,\mathbf{e}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{isa}(h,cs),\ \xi'} \langle \checkmark, \sigma', E\rangle$ for some $cs = [\xi(\mathbf{e}_n)]$, which means Rule 20.1 has been applied necessarily. Then, $\langle h\,!!\,\mathbf{e}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{isa}(h,cs),\ \xi'} \langle \checkmark, \sigma', E\rangle$, and $\xi \models b$. In turn, Rule 5 has been applied necessarily, which means that $\xi'_\sigma = \sigma'$, $\xi = \sigma \cup \xi^{\dot{C}L}$ such that $\xi^{\dot{C}L} \in \dot{C} \to \Lambda, \xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C},\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\ \xi(x) = \sigma(x)\}$. According to the translation defined for the guarded send process term, $\mathcal{T}_J(b \to h\,!!\,\mathbf{e}_n)$ has the initial location v_0 and one terminating location v'_0 that are connected by an urgent edge e, guarded by predicate b, with jump condition $(X_{\mathrm{aux}} \cup J, \bigwedge_{i=1}^{n} e_i = h'_i)$ and labelled with $\mathrm{isa}(h)$. Since $\xi \models b$ and $\xi \upharpoonright \mathrm{dom}(\sigma) = \sigma$ and the variables outside $\mathrm{dom}(\sigma)$ are not allowed to occur in b, we also have $\alpha \models \mathrm{guard}(e)$. From $\xi^{\dot{C}L} \in \dot{C} \to \Lambda, \xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C},\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\ \xi(x) = \sigma(x)\}$, we know that $(\alpha, \alpha') \models \mathrm{jump}(e)$. Therefore, $(v_0, \alpha) \xrightarrow{\mathrm{isa}(h)} (v'_0, \alpha')$ is in $[\![\mathcal{HA}(\langle b \to h\,!!\,\mathbf{e}_n, \sigma, E\rangle)]\!]$. Due to Lemma 3.5.1, we have $\xi'_{\sigma'} = \sigma'$, and $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. It is not hard to see that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$ and $\alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$.

C.3.4 Theorem 5.3.4.2 – part 2

The proof is by induction on the structure of closed process term p. Since there are no action transition rules (with $\mathrm{isa}(h, cs)$ as a label) defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate, guarded send process term and guarded receive process term, the theorem holds trivially for these cases.

The proofs for the any delay operator, the sequential composition operator, the alternative composition operator, the parallel composition operator, the repetition operator, the jump enabling operator, the action encapsulation operator, and the urgent communication operator are similar to the proofs for these operators in the second part of Theorem 5.3.4.1 in Appendix C.3.1 since these operators treat send actions similarly as normal actions.

C.3.5 Theorem 5.3.4.3 – part 1

The proof is by induction on the structure of closed process term p. Since there are no termination transition rules (with $\mathrm{ira}(h, cs, Y)$ as a label) defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate, guarded send process term, sequential composition, parallel composition and the repetition operator, the theorem holds trivially for these cases.

The proofs for the any delay operator, the alternative composition operator, the jump enabling operator, the action encapsulation operator, and the urgent communication operator are similar to the proofs for these operators in the first part of Theorem 5.3.4.1 in Appendix C.3.1 since these operators treat receive actions similarly as normal actions.

- Guarded receive $p \equiv b \to h \,??\, \mathbf{x}_n$ for some $b, h, \mathbf{x}_n$. We have $\langle b \to h \,??\, \mathbf{x}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{ira}(h,cs,Y),\ \xi'} \langle \checkmark, \sigma', E\rangle$ for some $Y = \{\mathbf{x}_n\}$, which means Rule 20.1 has been applied necessarily. Then, $\langle h \,??\, \mathbf{x}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{ira}(h,cs,Y),\ \xi'} \langle \checkmark, \sigma', E\rangle$, and $\xi \models b$. In turn, Rule 6 has been applied necessarily. Then $\xi'_\sigma = \sigma'$ and $\xi = \sigma \cup \xi^{\dot{C}L}$ such that $\xi^{\dot{C}L} \in \dot{C} \to \Lambda, \xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C}, \forall_{y \in \mathrm{dom}(\sigma) \setminus (J \cup \{\mathbf{x}_n\})}\, \xi(y) = \sigma(y)\}$, and $\xi'(\mathbf{x}_n) = cs$. According to the translation defined for the guarded receive process term, $\mathcal{T}_J(b \to h \,??\, \mathbf{x}_n)$ has the initial location v_0 and the terminating location v'_0 that are connected by an urgent edge e, guarded by predicate b, with jump condition $(\{\mathbf{x}_n\} \cup J \cup X_{\mathrm{aux}}, \bigwedge_{i=1}^{n} h'_i = x'_i)$ and labelled with $\mathrm{ira}(h, \{\mathbf{x}_n\})$. Since $\xi \models b$ and $\xi \restriction \mathrm{dom}(\sigma) = \sigma$ and the variables outside $\mathrm{dom}(\sigma)$ are not allowed to occur in b, we also have $\alpha \models \mathrm{guard}(e)$. From $\xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C}, \forall_{y \in \mathrm{dom}(\sigma) \setminus (J \cup \{\mathbf{x}_n\})}\, \xi(y) = \sigma(y)\}$, and $\xi'(\mathbf{x}_n) = cs$, we know that $(\alpha, \alpha') \models \mathrm{jump}(e)$. Therefore, $(v_0, \alpha) \xrightarrow{\mathrm{ira}(h)} (v'_0, \alpha')$ is in $[\![\, \mathcal{HA}(\langle b \to h \,??\, \mathbf{x}_n, \sigma, E\rangle) \,]\!]$.

C.3.6 Theorem 5.3.4.3 – part 2

The proof is by induction on the structure of closed process term p. Since there are no action transition rules (with $\mathrm{ira}(h, cs, Y)$ as a label) defined for delay predicate, consistent

deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate, guarded send process term and guarded receive process term, the theorem holds trivially for these cases.

The proofs for the any delay operator, the sequential composition operator, the alternative composition operator, the parallel composition operator, the repetition operator, the jump enabling operator, the action encapsulation operator, and the urgent communication operator are similar to the proofs for these operators in second part of Theorem 5.3.4.2 in Appendix C.3.1 since these operators treat receive actions similarly as normal actions.

C.3.7 Theorem 5.3.4.4 – part 1

The proof is by induction on the structure of closed process term p. Since there are no termination transition rules (with $\mathrm{ca}(h, cs)$ as specified) defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate, guarded send process term, guarded receive process term, sequential composition and the repetition operator, the theorem holds trivially for these cases.

The proofs for the any delay operator, the alternative composition operator, the jump enabling operator, the action encapsulation operator, and the urgent communication operator are similar to the proofs for these operators in the first part of Theorem 5.3.4.1 in Appendix C.3.1 since these operators treat communication actions similarly as normal actions.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $\langle q_1 \parallel q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \langle \surd, \sigma', (C, J, \emptyset, H, \emptyset)\rangle$, which means Rule 28.1.l or 28.1.r has been applied necessarily. Since the proofs for both cases are similar, we only give the proofs for the case that Rule 28.1.l has been applied. Then, $\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \surd, \sigma', (C, J \cup W, \emptyset, H, \emptyset)\rangle$ for some W, and $\langle q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \surd, \sigma', (C, J, \emptyset, H, \emptyset)\rangle$. By part 1 of Theorem 5.3.4.2 we then have $[\![\mathcal{HA}(\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{\mathrm{isa}(h)} (v'_{q_1}, \alpha')$ for some v_{q_1}, v'_{q_1} such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$ and $\alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$, where v_{q_1} and v'_{q_1} denote the initial location and a terminating location of $\mathcal{T}_J(q_1)$, respectively. We also know that there is an edge, say e_{q_1} in $\mathcal{T}_J(q_1)$ with source location v_{q_1}, target location v'_{q_1}, event $\mathrm{isa}(h)$, and a guard and a jump condition that hold for α and α'. Also, by part 1 of Theorem 5.3.4.3, we have $[\![\mathcal{HA}(\langle q_2, \sigma, E\rangle)]\!] \models (v_{q_2}, \alpha) \xrightarrow{\mathrm{ira}(h,W)} (v'_{q_2}, \alpha')$ for some v_{q_2}, v'_{q_2} such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$, where v_{q_2} and v'_{q_2} denote the initial location and a terminating location of $\mathcal{T}_J(q_2)$. Therefore, we also know that there exists an edge, say e_{q_2}, with source location v_{q_2}, target location v'_{q_2}, event $\mathrm{ira}(h, W)$, and a guard and a jump condition that hold for α and α'. According to the translation defined for the parallel composition operator, there is an edge (e_{q_1}, e_{q_2}) in $\mathcal{T}_J(q_1 \parallel q_2)$ with source location (v_{q_1}, v_{q_2}), target location v_{done}, event $\mathrm{ca}(h)$, a guard and a jump condition that are the logical and of the guard and jump conditions of e_{q_1} and e_{q_2} and, therefore hold for α and α', and an urgency status that is the logical

or of the urgency status of edges e_{q_1} and e_{q_2}. Hence, $((v_{q_1}, v_{q_2}), \alpha) \xrightarrow{\mathrm{ca}(h)} (v_{\mathrm{done}}, \alpha')$ is in $[\![\mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle)]\!]$. Note that (v_{q_1}, v_{q_2}) and v_{done} are the initial location and a terminating location of $\mathcal{T}_J(q_1 \parallel q_2)$.

C.3.8 Theorem 5.3.4.4 – part 2

The proof is by induction on the structure of closed process term p. Since there are no action transition rules (with $\mathrm{ca}(h, cs)$ as specified) defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate, guarded send process term and guarded receive process term, the theorem holds trivially for these cases.

The proofs for the any delay operator, the sequential composition operator, the alternative composition operator, the repetition operator, the jump enabling operator, the action encapsulation operator, and the urgent communication operator are similar to the proofs for these operators in the second part of Theorem 5.3.4.2 in Appendix C.3.4 since these operators treat communication actions similarly as normal actions.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $\langle q_1 \parallel q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \langle p', \sigma', (C, J, \emptyset, H, \emptyset)\rangle$, which means Rule 28.2.l, 28.2.r, 28.3.l, 28.3.r, 28.4.l or 28.4.r has been applied necessarily. Since the proofs for most cases are similar, we only give the proofs for the cases that Rules 28.2.l and 28.4.l have been applied.

  - Rule 28.2.l has been applied. Then, $\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle p', \sigma', (C, J \cup W, \emptyset, H, \emptyset)\rangle$ and $\langle q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma', (C, J, \emptyset, H, \emptyset)\rangle$ for some W. By part 2 of Theorem 5.3.4.2 we then have $[\![\mathcal{HA}(\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{\mathrm{isa}(h)} (v'_{q_1}, \alpha')$ for some v_{q_1}, v'_{q_1} such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$ and $\alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$, where v_{q_1} and v'_{q_1} denote the initial location of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(p')$. We also know that there exists an edge, say e_{q_1}, with source location v_{q_1}, target location v'_{q_1}, event $\mathrm{isa}(h)$ and a guard and jump condition that hold for α and α'. By part 1 of Theorem 5.3.4.3 we have $[\![\mathcal{HA}(\langle q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_2}, \alpha) \xrightarrow{\mathrm{ira}(h, \{\mathbf{x}_n\})} (v'_{q_2}, \alpha')$ for some v_{q_2}, v'_{q_2} such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$, where v_{q_2} and v'_{q_2} denote the initial location and a terminating location of $\mathcal{T}_J(q_2)$, respectively. We also know that there is an edge, say e_{q_2}, in $\mathcal{T}_J(q_2)$ with source location v_{q_2}, target location v'_{q_2}, event $\mathrm{ira}(h, W)$, and a guard and a jump condition that hold for α and α'. According to the translation defined for the parallel composition operator, (e_{q_1}, e_{q_2}) is an edge in $\mathcal{T}_J(q_1 \parallel q_2)$ with source location such that (v_{q_1}, v_{q_2}), target location v'_{q_1}, event $\mathrm{ca}(h)$, an urgency status that is the logical or of the urgency status of e_{q_1} and e_{q_2}, and a guard and a jump condition that are the logical and of the guard and the jump conditions of e_{q_1} and e_{q_2}. Note that this guard

and this jump condition therefore hold for α and α'. Hence, $((v_{q_1}, v_{q_2}), \alpha) \xrightarrow{\text{ca}(h)} (v'_{q_1}, \alpha')$ is in $[\![\mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle)]\!]$. Note that (v_{q_1}, v_{q_2}) is the initial location of $\mathcal{T}_J(q_1 \parallel q_2)$.

– Rule 28.4.l has been applied. Then, $\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle q'_1, \sigma', (C, J \cup W, \emptyset, H, \emptyset)\rangle$ and $\langle q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle q'_2, \sigma', (C, J, \emptyset, H, \emptyset)\rangle$ for some q'_1, q'_2 and W such that $p' \equiv q'_1 \parallel q'_2$. By part 2 of Theorem 5.3.4.2 we then have $[\![\mathcal{HA}(\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{\text{isa}(h)} (v'_{q_1}, \alpha')$ for some v_{q_1}, v'_{q_1} such that $\sigma = \alpha \restriction \text{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \text{dom}(\sigma')$ and $\alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$, where v_{q_1} and v'_{q_1} denote the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q'_1)$, respectively. We also know that there exists an edge, say e_{q_1} in $\mathcal{T}_J(q_1)$ with source location v_{q_1}, target location v'_{q_1}, event isa(h), and a guard and a jump condition that hold for α and α'. By part 2 of Theorem 5.3.4.3 we have $[\![\mathcal{HA}(\langle q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_2}, \alpha) \xrightarrow{\text{ira}(h,W)} (v'_{q_2}, \alpha')$ for some v_{q_2}, v'_{q_2} such that $\sigma = \alpha \restriction \text{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \text{dom}(\sigma')$, where v_{q_2} and v'_{q_2} denote the initial location of $\mathcal{T}_J(q_2)$ and $\mathcal{T}_J(q'_2)$. We also know that there exists an edge, say e_{q_2}, with source location v_{q_2}, target location v'_{q_2}, event ira(h, W), and a guard and a jump condition that hold for α and α'. According to the translation defined for the parallel composition operator there exists an edge (e_{q_1}, e_{q_2}) in $\mathcal{T}_J(q_1 \parallel q_2)$ with source location (v_{q_1}, v_{q_2}), target location (v'_{q_1}, v'_{q_2}), event ca(h), a guard and a jump condition that hold for α and α', and an urgency status that is the logical or of the urgency status of edges e_{q_1} and e_{q_2}. Hence, $((v_{q_1}, v_{q_2}), \alpha) \xrightarrow{\text{ca}(h)} ((v'_{q_1}, v'_{q_2}), \alpha')$ is in $[\![\mathcal{HA}(\langle q_1 \parallel q_2, \sigma, \rangle)]\!]$. Note that (v_{q_1}, v_{q_2}) and (v'_{q_1}, v'_{q_2}) are the initial locations of $\mathcal{T}_J(q_1 \parallel q_2)$ and $\mathcal{T}_J(q'_1 \parallel q'_2)$ respectively.

C.3.9 Theorem 5.3.4.5

The proof is by induction on the structure of closed process term p. Since there are no time transition rules defined for consistent deadlock δ, the theorem holds trivially for this case.

- Delay predicate $p \equiv u$ for some u. We have $\langle u, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E\rangle$ and $\sigma' = \rho_\sigma(t)$, which means Rule 3 has been applied necessarily. Then $p' \equiv u$, $\rho \in \Omega_{FG}(\sigma, C, \emptyset, u, t)$. According to the translation defined for the delay predicate, $\mathcal{T}_J(u)$ has only one location v_0 (it is also the initial location) with flow predicate u and invariant $u[DC/\dot{C}]$ and has no outgoing edges. So, let ρ' be the trajectory such that $\rho = \rho' \downarrow (\text{dom}(\sigma) \cup \dot{C})$ and $\rho'(r)(\dot{c}) = \rho'(r)(dc)$ for $r \in [0, t]$ and $c \in C$. By Theorem 5.3.3, ρ' is a solution for the flow predicate u. From the fact that ρ' is a solution for u and $\rho'(r)(\dot{c}) = \rho'(r)(dc)$ for $r \in [0, t]$ and $c \in C$ it follows that ρ' is a solution for the invariant $u[DC/\dot{C}]$. Since there are no outgoing edges for location v_0, we can conclude that $(v_0, \alpha) \xmapsto{t} (v_0, \alpha')$ is in $[\![\mathcal{HA}(\langle u, \sigma, E\rangle)]\!]$.

- Guarded inconsistent process term $b \to \bot$ for some b. We know that the guarded inconsistent process term $b \to \bot$ is equivalent to $\neg b$ (see also Proposition 3.5.4). It is not hard to see that the proof for this case is similar to the proof for the case of the delay predicate.

- Guarded action predicate $p \equiv b \to W : r \gg l_{\mathrm{a}}$ for some b, W, r, l_{a}. We have $\langle b \to W : r \gg l_{\mathrm{a}}, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$, which means Rule 22 has been applied necessarily (Rule 21 cannot be applied, because no time transitions defined for action predicate). Then, we have $\rho \in \Omega_{\sigma E t}, \forall_{s\in(0,t)}\, \rho(s) \models \neg b$, $\exists_{s\in[0,t]}\, \rho(s) \models \neg b$ and $p' \equiv b \to W : r \gg l_{\mathrm{a}}$. Let ρ' be the trajectory such that $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$. According to the translation defined for the guarded action predicate, the initial location v_0 of $\mathcal{T}_J(b \to W : r \gg l_{\mathrm{a}})$ has both invariant and flow condition true with only one urgent outgoing edge, which is guarded by b. It is allowed to perform an arbitrary time transition in the location v_0 as long as the guard b is false (for ρ'). Hence, $(v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$ is in $[\![\, \mathcal{HA}(\langle b \to W : r \gg l_{\mathrm{a}}, \sigma, E\rangle) \,]\!]$.

- Guarded send. The proof for guarded send is similar to the proof for guarded action predicate.

- Guarded receive. The proof for guarded receive is similar to the proof for guarded action predicate.

- Any delay operator $p \equiv [q]$ for some q. We have $\langle [q], \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$, which means Rule 11 has been applied necessarily. Then, we have $\rho \in \Omega_{\sigma E t}$ (i.e., an arbitrary trajectory), $p' \equiv [q]$ and $\sigma' = \rho_\sigma(t)$. Let ρ' be the trajectory such that $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$. From the function Ω_{FG} and Lemma 3.5.1, it is not hard to see that $\rho'(0) = \alpha$ and $\rho'(r) = \alpha'$. According to the translation defined for the any delay operator, an additional initial location v is introduced into $\mathcal{T}_J(q)$ to obtain $\mathcal{T}_J([q])$. The invariant and flow condition of v are true. Also, all outgoing edges of the initial location of $\mathcal{T}_J(q)$ are copied to the initial location v of $\mathcal{T}_J([q])$ with original targets, but the urgency status of those edges is set to false (i.e. non-urgent). Obviously, the time transition $(v, \alpha) \overset{t}{\longmapsto} (v, \alpha')$ is in $[\![\, \mathcal{HA}(\langle [q], \sigma, E\rangle) \,]\!]$.

- Sequential composition operator $p \equiv q_1;\, q_2$ for some q_1 and q_2. We have $\langle q_1;\, q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$, which means Rule 18 has been applied necessarily. Then we get $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E\rangle$ for some q_1' and $p' \equiv q_1';\, q_2$. By induction we then have $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!] \models \rho' : (v_{q_1}, \alpha) \overset{t}{\longmapsto} (v_{q_1}, \alpha')$ for some v_{q_1} such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_{q_1} denotes the initial location of $\mathcal{T}_J(q_1)$. We also know that there exists ρ' such that $\rho'(0) = \alpha, \rho'(t) = \alpha'$ and $\forall_{r\in[0,t]}\, \rho'(r) \models \mathrm{inv}(v_{q_1}) \wedge \mathrm{flow}(v_{q_1})$. Also, in $\mathcal{T}_J(q_1)$, the guard of any urgent edge with source location v_{q_1} does not hold for $[0, t)$. According to the translation defined for the sequential composition operator, v_{q_1} is also the initial location of $\mathcal{T}_J(q_1;\, q_2)$ and no new edges from v_{q_1} are added into $\mathcal{T}_J(q_1;\, q_2)$. Therefore, also $(v_{q_1}, \alpha) \overset{t}{\longmapsto} (v_{q_1}, \alpha')$ is in $[\![\, \mathcal{HA}(\langle q_1;\, q_2, \sigma, E\rangle) \,]\!]$.

- Alternative composition operator $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2. We have $\langle q_1 \,[\!]\, q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$, which means Rule 26 has been applied necessarily. Then we get $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E\rangle$ for some q_1' and q_2', and $p' \equiv q_1' \,[\!]\, q_2'$. By induction we then have $[\![\, \mathcal{HA}(\langle q_1, \sigma, E\rangle) \,]\!] \models \rho' : (v_{q_1}, \alpha) \overset{t}{\longmapsto} (v_{q_1}, \alpha')$ for some v_{q_1} such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$, and $[\![\, \mathcal{HA}(\langle q_2, \sigma, E\rangle) \,]\!] \models \rho' : (v_{q_2}, \alpha) \overset{t}{\longmapsto} (v_{q_2}, \alpha')$ for some v_{q_1} such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$, where v_{q_1} and v_{q_2} denote the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$ and ρ' is the witness function obtained from ρ. For this witness function we have $\rho'(0) = \alpha, \rho'(t) = \alpha'$, and $\forall_{r\in[0,t]}\ \rho'(r) \models \mathrm{inv}(v_{q_1}) \wedge \mathrm{flow}(v_{q_1})$, and $\forall_{r\in[0,t]}\ \rho'(r) \models \mathrm{inv}(v_{q_2}) \wedge \mathrm{flow}(v_{q_2})$. Furthermore, for both these locations we have that the guard of any outgoing urgent edge does not hold in $[0, t)$. According to the translation defined for the alternative composition operator, the invariant and flow condition of the initial location $v_0 = (v_{q_1}, v_{q_2})$ of $\mathcal{T}_J(q_1 \,[\!]\, q_2)$ is the conjunction of the invariants and the flow conditions of v_{q_1} and v_{q_2}. These satisfy the conditions for a time transition of duration t. Note that the outgoing edges of location (v_{q_1}, v_{q_2}) in $\mathcal{T}_J(q_1 \,[\!]\, q_2)$ are precisely (i.e., with the same events, the same guard and jump conditions, the same urgency status) the outgoing edges of v_{q_1} in $\mathcal{T}_J(q_1)$ and v_{q_2} in $\mathcal{T}_J(q_2)$. Therefore, also $(v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$ is in $[\![\, \mathcal{HA}(\langle q_1 \,[\!]\, q_2, \sigma, E\rangle) \,]\!]$.

- Parallel composition operator. The proofs are similar to the proofs of the alternative composition operator. An important difference is that in parallel composition new communication transitions from the initial state of the parallel composition might occur. Note that these have an urgency status that is the disjunction of the urgency status of the contributing send and receive transitions. Hence, since these were not preventing the time transition, so is not the communication transition.

- Repetition operator $p \equiv *q$ for some q. We have $\langle *q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$. Then, by Rule B in Appendix C.1, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some q' and $p' \equiv q'; *q$. By induction, we then have $[\![\, \mathcal{HA}(\langle q, \sigma, E\rangle) \,]\!] \models \rho' : (v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ for some v_q such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$, where v_q denotes the initial location of $\mathcal{T}_J(q)$. We also know that there exists a ρ' such that $\rho'(0) = \alpha, \rho'(t) = \alpha'$, and $\forall_{r\in[0,t]}\ \rho'(r) \models \mathrm{inv}(v_q) \wedge \mathrm{flow}(v_q)$. According to the translation defined for the repetition operator, v_q is also the initial location of $\mathcal{T}_J(*q)$, and no new outgoing edges from v_q are introduced into $\mathcal{T}_J(*q)$. Hence, also $(v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ is in $[\![\, \mathcal{HA}(\langle *q, \sigma, E\rangle) \,]\!]$.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. We have $\langle \iota_{J^+}(q),\ \sigma,\ (C,\ J, \emptyset, H, \emptyset)\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', (C, J, \emptyset, H, \emptyset)\rangle$ for some q', which means Rule 42 has been applied necessarily. Then, we have $\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', (C, J \cup J^+, \emptyset, H, \emptyset)\rangle$ for some q' and $p' \equiv \iota_{J^+}(q')$. By induction we then have $[\![\, \mathcal{HA}(\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle) \,]\!] \models \rho' : (v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ for some v_q such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$, where v_q denotes the initial location of $\mathcal{T}_{J\cup J^+}(q)$. We also know that there exists a ρ' such that $\rho'(0) = \alpha, \rho'(t) = \alpha'$, and $\forall_{r\in[0,t]}\ \rho'(r) \models \mathrm{inv}(v_q) \wedge \mathrm{flow}(v_q)$. According

to the translation defined for the jump enabling operator, we have $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J\cup J^+}(q)$, v_q is also the initial location of $\mathcal{T}_J(\iota_{J^+}(q))$. Obviously, $(v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \iota_{J^+}(q), \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle)]\!]$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. We have $\langle \partial_A(q),\ \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$, which means Rule 33 has been applied necessarily. Then, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ and $p' \equiv \partial_A(q')$ for some q'. By induction, we have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models \rho' : (v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ for some v_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q denotes the initial location of $\mathcal{T}_J(q)$. We also know that there exists ρ' such that $\rho'(0) = \alpha, \rho'(t) = \alpha'$, and $\forall_{r\in[0,t]}\ \rho'(r) \models \mathrm{inv}(v_q) \wedge \mathrm{flow}(v_q)$. According to the translation defined for action encapsulation operator, v_q is also the initial location of $\mathcal{T}_J(\partial_A(q))$, and no new outgoing edges from v_q are introduced into $\mathcal{T}_J(\partial_A(q))$. Hence, also $(v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \partial_A(q), \sigma, E\rangle)]\!]$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $\mathcal{H}$ and q. We have $\langle \upsilon_H(q), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$, which means Rule 37 has been applied necessarily. Then, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some q', $\langle q, \sigma\rangle \overset{\mathrm{ca}(h,*)}{\nrightarrow}$ and $\forall_{s\in[0,t)}\ (\langle q, \sigma, E\rangle \overset{s,\rho\upharpoonright[0,s]}{\longmapsto} \langle q_s, \sigma_s, E\rangle\ ,\ \langle q_s, \sigma_s, E\rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle q', \sigma', E\rangle\ ,\ \forall_{h\in\mathcal{H}}\ \langle q_s, \sigma_s, E\rangle \overset{\mathrm{ca}(h,*)}{\nrightarrow})$ and $p' \equiv \upsilon_H(q')$. By induction we then have $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!] \models \rho' : (v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ for some v_q such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$, where v_q denotes the initial location of $\mathcal{T}_J(q)$ and some $\rho' : [0, t] \to (X \mapsto \Lambda)$ such that $\rho'(0) = \alpha, \rho'(t) = \alpha'$, and $\forall_{r\in[0,t]}\ \rho'(r) \models \mathrm{inv}(v_q) \wedge \mathrm{flow}(v_q)$ and no outgoing edge (with any label) is enabled in $[0, t)$. According to the translation defined for urgent communication operator, v_q is also the initial location of $\mathcal{T}_J(\upsilon_H(q))$, no new outgoing edges are added to v_q of $\mathcal{T}_J(q)$ to obtain $\mathcal{T}_J(\upsilon_H(q))$, and the urgency status of all edges labelled with $\mathrm{ca}(h)$ of $\mathcal{T}_J(\upsilon_H(q))$ is set to true, which means that if there is an action transition via an outgoing edge of v_q with label $\mathrm{ca}(h)$ is enabled, then time transition is not allowed in v_q. Hence, this (above-mentioned time transition) $(v_q, \alpha) \overset{t}{\longmapsto} (v_q, \alpha')$ is in $[\![\mathcal{HA}(\langle \upsilon_H(q), \sigma, E\rangle)]\!]$.

C.4 Proof of Theorem 5.3.5

C.4.1 Theorem 5.3.5.1 - part 1

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ and guarded inconsistent process term $b \to \bot$, guarded send process term and guarded receive process term these constants cannot perform any transition (with specified label a). Hence, the theorem holds trivially for these cases.

- Guarded action predicate $p \equiv b \to W : r \gg l_{\mathrm{a}}$ for some b, W, r, l_{a}. We have $[\![\ \mathcal{HA}(\langle b \to W : r \gg l_{\mathrm{a}}, \sigma, E\rangle)\ ]\!] \models (v_0, \alpha) \overset{a}{\to} (v_0', \alpha')$, which means that there exists an edge e

in $\mathcal{T}_J(b \to W : r \gg l_\mathrm{a})$ with source location v_0, target location v_0', event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for the guarded action predicate $a = l_\mathrm{a}$, the guard is b, and the jump condition is $(W \cup J \cup X_\mathrm{aux}, \zeta_{W \cup J}(r))$. From the fact that $(\alpha, \alpha') \models \mathrm{jump}(e)$, we have $\xi^- \cup \xi' \models r$ for some ξ^-, ξ', where $\xi = \sigma \cup \xi^{\dot{C}L}$, $\xi^{\dot{C}L} \in \dot{C} \to \Lambda$ and $\xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C}, \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\, \xi(x) = \sigma(x)\}$. Using Rule 1, we conclude that $\langle W : r \gg l_\mathrm{a}, \sigma, E\rangle \xrightarrow{\xi,\ l_\mathrm{a},\ \xi'} \langle \checkmark, \xi'_\sigma, E\rangle$, where $\xi'_\sigma = \sigma'$. From $\alpha \models b$, we know that $\xi \models b$. Using Rule 20.1, we obtain $\langle b \to W : r \gg l_\mathrm{a}, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$.

- Any delay operator $p \equiv [q]$ for some q. We have $[\![\mathcal{HA}(\langle [q], \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$, which means that there exists an edge e in $\mathcal{T}_J([q])$ with source location v_0, target location v_0', event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for the any delay operator, the outgoing edges of v_0 are the copies (in the sense that they have the same target locations and jump conditions) of the outgoing edges of the initial location v_0'' of $\mathcal{T}_J(q)$, but the outgoing edges of v_0 and v_0'' may have different urgency status. Also v_0' is also a terminating location of $\mathcal{T}_J(q)$, and $(v_0'', \alpha) \xrightarrow{a} (v_0', \alpha')$ is in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$ for some E, ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 10.1, we obtain $\langle [q], \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$.

- Sequential composition operator $p \equiv q_1;\ q_2$ for some q_1 and q_2. We have $\mathcal{HA}(\langle q_1; q_2, \sigma, E\ \rangle)\ ]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$, which means v'_{q_1} has to be a terminating location of $\mathcal{T}_J(q_1;\ q_2)$. However, it is not possible in our translation. Hence, the theorem holds trivially.

- Alternative composition operator $p \equiv q_1 \,[\!]\, q_2$ for some q_1 and q_2. We have $[\![\mathcal{HA}(\langle q_1 \,[\!]\, q_2, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$, which means that there exists an edge e in $\mathcal{T}_J(q_1 \,[\!]\, q_2)$ with source location v_0, target location v_0', event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for the alternative composition operator, $v_0 = (v_{q_1}, v_{q_2})$ and v_0' is a terminating location of either $\mathcal{T}_J(q_1)$ or $\mathcal{T}_J(q_2)$, where v_{q_1} and v_{q_2} are the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$, respectively. We distinguish two cases:

  - v_0' is a terminating location of $\mathcal{T}_J(q_1)$. Again, referring to the translation defined for the alternative composition operator, we know that $(v_{q_1}, \alpha) \xrightarrow{a} (v_0', \alpha')$ is in $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$ for some ξ, ξ', σ' such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 25.1.l, we obtain $\langle q_1 \,[\!]\, q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$.
  - v_0' is a terminating location of $\mathcal{T}_J(q_2)$. The proofs of this case are similar to the previous case.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $[\![\mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$, which means that there is an edge, say e, with source location v_0, target location v_0', event a and a guard and a jump condition that hold for α and α'. According to the translation defined for the parallel composition operator, $a = \mathrm{ca}(h)$ necessarily. This leads to a contradiction. Hence, the theorem holds trivially.

- Repetition operator $p \equiv *q$ for some q. The translation of the repetition operator always results in a hybrid automaton fragment without terminating locations. Therefore, this case cannot occur.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. We have $[\![\mathcal{HA}(\langle \iota_{J^+}(q), \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$. According to the translation defined for jump enabling operator, $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J\cup J^+}(q)$. We also know that $(v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$ is in $[\![\mathcal{HA}(\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle)]\!]$. By induction, we then have $\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', (C, J \cup J^+, \emptyset, H, \emptyset)\rangle$ for some ξ, ξ', σ' such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 41.1, we obtain $\langle \iota_{J^+}(q), \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E\rangle$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. We have $[\![\mathcal{HA}(\langle \partial_A(q), \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$. According to the translation defined for action encapsulation operator, we know that the jump conditions of edges labelled with events from A are predicates false with an empty set of variables that are allowed to change, and $(v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$ is also in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$, which also implies $a \notin A$ necessarily. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E\rangle$ for some ξ, ξ', σ' such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 32.1, we get $\langle \partial_A(q), \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E\rangle$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $H \subseteq \mathcal{H}$ and q. We have $[\![\mathcal{HA}(\langle \upsilon_H(q), \sigma, E \rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$. According to the translation defined for urgent communication operator, $(v_0, \alpha) \xrightarrow{a} (v_0', \alpha')$ is also in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E\rangle$ for some ξ, ξ', σ' such that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 35.1, we get $\langle \upsilon_H(q), \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E\rangle$.

C.4.2 Theorem 5.3.5.1 - part 2

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate, guarded send process term and guarded receive process term these constants cannot perform any action transition (with specified label a). Hence, the theorem holds trivially for these cases.

- Any delay operator $p \equiv [q]$ for some q. We have $[\![\mathcal{HA}(\langle [q], \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there exists an non-urgent edge e with source location v_0, target location v_0'', event a and a guard and a jump condition that hold for α

and α'. According to the translation defined for the any delay operator, the action transition $(v_0^*, \alpha) \xrightarrow{a} (v_0'', \alpha')$ is in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$, where v_0^* is the initial location of $\mathcal{T}_J(q)$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E\rangle$ for some q', ξ, ξ', σ' such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$. Using Rule 10.2, we obtain $\langle [q], \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E\rangle$.

- Sequential composition operator $p \equiv q_1 ; q_2$ for some q_1, q_2. We have $\mathcal{HA}(\langle\ q_1; q_2, \sigma, E\rangle)\ ]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v_q', \alpha')$, which means that there exists an edge e in $\mathcal{T}_J(q_1 ;\ q_2)$ with source location v_{q_1}, target location v_q', event a, and a guard and a jump condition that hold for α and α'. We can distinguish two cases:

  - v_q' is the initial location of $\mathcal{T}_J(q_2)$, which implies $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{a} (v_{q_1}', \alpha')$ for some terminating location v_{q_1}' of $\mathcal{T}_J(q_1)$. According to part 1 of Theorem 5.3.5.1, we obtain $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$ for some ξ, ξ', σ' such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$. Using Rule 16, we obtain $\langle q_1 ;\ q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_2, \sigma', E\rangle$.
  - v_q' is a location of $\mathcal{T}_J(q_1)$. By induction we then have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E\rangle$ for some q_1', ξ, ξ' such that $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$ and v_q' is also the initial location of $\mathcal{T}_J(q_1')$. Using Rule 17, we obtain $\langle q_1 ;\ q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1' ;\ q_2, \sigma', E\rangle$. Note that v_q' is the initial location of $\mathcal{T}_J(q_1' ;\ q_2)$.

- Alternative composition operator $p \equiv q_1 \ [\!]\ q_2$ for some q_1 and q_2. We have $[\![\mathcal{HA}(\langle q_1 \ [\!]\ q_2, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0'', event a and a guard and a jump condition that hold for α and α'. According to the translation defined for the alternative composition operator, $v_0 = (v_{q_1}, v_{q_2})$ and v_0'' is a location of either $\mathcal{T}_J(q_1)$ or $\mathcal{T}_J(q_2)$, where v_{q_1} and v_{q_2} are the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$. We distinguish two cases:

  - v_0'' is a location of $\mathcal{T}_J(q_1)$. Again, referring to the translation defined for the alternative composition operator, we know that $(v_{q_1}, \alpha) \xrightarrow{a} (v_0'', \alpha')$ is in $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1', \sigma', E\rangle$ for some ξ, ξ', σ' such that $\sigma = \alpha \restriction \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \mathrm{dom}(\sigma')$ and v_0'' is the initial location of $\mathcal{T}_J(q_1)$. Using Rule 25.2.l, we obtain $\langle q_1 \ [\!]\ q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_1' ;\ q_2, \sigma', E\rangle$. Note that v_0'' is also the initial location of $\mathcal{T}_J(q_1' ;\ q_2)$.
  - v_0'' is a location of $\mathcal{T}_J(q_2)$. The proofs of this case are similar to the previous case.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $[\![\mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0'', event a and a guard and a jump condition that hold for α and α'. According to the translation defined for the parallel composition operator,

$v_0 = (v_{q_1}, v_{q_2})$, where and v_{q_1} and v_{q_2} denote the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$. For v_0'' we can distinguish four cases:

- $v_0'' = (v'_{q_1}, v_{q_2})$ for some location v'_{q_1} of $\mathcal{T}_J(q_1)$. Then, $(v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ is in $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E\rangle$ for some q', ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$ and v'_{q_1} is the initial location of $\mathcal{T}_J(q')$. Using Rule 29.2.l, we obtain $\langle q_1 \parallel q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q' \parallel q_2, \sigma', E\rangle$. Note that (v'_{q_1}, v_{q_2}) is the initial location of $\mathcal{T}_J(q' \parallel q_2)$.
- $v_0'' = (v_{q_1}, v'_{q_2})$ for some location v'_{q_2} of $\mathcal{T}_J(q_2)$. The proofs of this case are similar to the previous case.
- $v_0'' = v_{q_2}$. Then, $(v_{q_1}, \alpha) \xrightarrow{a} (v'_{q_1}, \alpha')$ is in $[\![\mathcal{HA}(\langle q_1, \sigma, E\rangle)]\!]$ and v'_{q_1} is a terminating location of $\mathcal{T}_J(q_1)$. By part 1 of Theorem 5.3.5.1, we then have $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$ for some ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 29.2.l, we obtain $\langle q_1 \parallel q_2, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q_2, \sigma', E\rangle$. By definition, v_{q_2} is the initial location of $\mathcal{T}_J(q' \parallel q_2)$.
- $v_0'' = v_{q_1}$. The proofs of this case are similar to the previous case.

- Repetition operator $p \equiv *q$ for some q. We have $[\![\mathcal{HA}(\langle *q, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there is an edge, say e, with source location v_0, target location v_0'', event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for the repetition operator, $(v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$ is in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$ (because e is also an edge in $\mathcal{T}_J(q)$ and v_0'' is not a terminating location). By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', E\rangle$ for some q', ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Then, we obtain $\langle *q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q'; *q, \sigma', E\rangle$ using Rule A.2 from Appendix C.1.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. We have $[\![\mathcal{HA}(\langle\, \iota_{J^+}(q), \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0'', event a, and a guard and a jump condition that hold for α and α'. According to the translation defined for jump enabling operator, we have $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J \cup J^+}(q)$, we also know that $(v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$ is in $[\![\mathcal{HA}(\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle)]\!]$. Note that we then have $\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi,a,\xi'} \langle q', \sigma', (C, J \cup J^+, \emptyset, H, \emptyset)\rangle$ for some q', ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$ and v_0'' is the initial location of $\mathcal{T}_J(q')$. Using Rule 41.2, we obtain $\langle \iota_{J^+}(q), \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \iota_{J^+}(q'), \sigma', E\rangle$. By definition, v_0'' is also the initial location of $\mathcal{T}_J(\iota_{J^+}(q'))$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. We have $[\![\ \mathcal{HA}(\langle\, \partial_A(q), \sigma, E\rangle)\ ]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0'', event a, and a guard and jump condition that hold for α and α'. According to the translation defined for action encapsulation operator, we

know that no edges labelled with events from A are enabled (which implies $a \notin A$), and $(v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$ is also in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle q', \sigma', E\rangle$ for some q', ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$ and v_0'' is the initial location of $\mathcal{T}_J(q')$. Using Rule 32.1, we get $\langle \partial_A(q), \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \partial_A(q'), \sigma', E\rangle$. Note that v_0'' is also the initial location of $\mathcal{T}_J(\partial_A(q'))$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some $H \subseteq \mathcal{H}$ and q. We have $[\![\mathcal{HA}(\langle \upsilon_H(q),\ \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0'', event a, and a guard and jump condition that hold for α and α'. According to the translation defined for urgent communication operator, $(v_0, \alpha) \xrightarrow{a} (v_0'', \alpha')$ is also in $[\![\mathcal{HA}(\langle q, \sigma, E\rangle)]\!]$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle q', \sigma', E\rangle$ for some q', ξ, ξ', σ' such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$ and v_0'' is the initial location of $\mathcal{T}_J(q')$. Using Rule 35.2, we get $\langle \upsilon_H(q), \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \upsilon_H(q'), \sigma', E\rangle$. Note that v_0'' is also the initial location of $\mathcal{T}_J(\upsilon_H(q'))$.

C.4.3 Theorem 5.3.5.2 - part 1

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \to \bot$, guarded action predicate and guarded receive process term these constants cannot perform any action transition (with specified label $\mathrm{isa}(h)$). Hence, the theorem holds trivially for these cases.

For the any delay operator, sequential composition, alternative composition, parallel composition, repetition, jump enabling, action encapsulation and urgent communication the proofs are similar to the proofs of these cases in Appendix C.4.1.

- Guarded send $p \equiv b \to h \,!!\, \mathbf{e}_n$ for some $h, \mathbf{e}_n$. We have $[\![\mathcal{HA}(\langle b \to h \,!!\, \mathbf{e}_n, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{isa}(h)} (v_0', \alpha')$, which means that there exists an edge e with source location v_q, target location v_q', event $\mathrm{isa}(h)$, and a guard and jump condition that hold for α and α'. According to the translation defined for the guarded send, $\mathrm{guard}(e) = b$ and $\mathrm{jump}(e) = (X_{\mathrm{aux}} \cup J, \bigwedge_{i=1}^{n} e_i = h_i')$. From $\alpha \models \mathrm{guard}(e)$ and $(\alpha, \alpha') \models \mathrm{jump}(e)$, we know that there exist ξ, ξ', σ' such that $\xi = \sigma \cup \xi^{\dot{C}L}$, where $\xi^{\dot{C}L} \in \dot{C} \to \Lambda$, and $\xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C},\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\, \xi(x) = \sigma(x)\}$. Using Rule 5, we obtain $\langle h \,!!\, \mathbf{e}_n,\ \sigma,\ E\rangle \xrightarrow{\xi,\ \mathrm{isa}(h, [\xi(\mathbf{e}_n)]),\ \xi'} \langle \checkmark, \xi_\sigma', E\rangle$, where $\xi_\sigma' = \sigma'$. Due to Lemma 3.5.1 and $(\alpha, \alpha') \models \mathrm{jump}(e)$, it is not hard to see that $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$ and $(\alpha'(h_1) = \xi(e_1) \wedge \cdots \wedge \alpha'(h_n) = \xi(e_n))$. Using Rule 20.1, we know that $\xi \models b$ (from $\alpha \models \mathrm{guard}(e)$), and we obtain $\langle b \to h \,!!\, \mathbf{e}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{isa}(h, [\xi(\mathbf{e}_n)]),\ \xi'} \langle \checkmark, \xi_\sigma', E\rangle$.

C.4.4 Theorem 5.3.5.2 - part 2

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \rightarrow \bot$, guarded action predicate, guarded send process term and guarded receive process term these constants cannot perform any action transition (with specified label $\mathrm{isa}(h)$). Hence, the theorem holds trivially for these cases.

For the operators the proofs are similar to the proofs for these cases in Appendix C.4.2. The only difference is that additionally the condition $\alpha'(h_1) = cs_1 \wedge \cdots \wedge \alpha'(h_n) = cs_n$ has to be proven. In all cases it follows from induction hypothesis.

C.4.5 Theorem 5.3.5.3 - part 1

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \rightarrow \bot$, guarded action predicate and guarded send process term these constants cannot perform any transition (with specified label $\mathrm{ira}(h, Y)$). Hence, the theorem holds trivially for these cases.

For the any delay operator, sequential composition, alternative composition, parallel composition, repetition, jump enabling, action encapsulation and urgent communication the proofs are similar to the proofs of these cases in Appendix C.4.1.

- Guarded receive $p \equiv b \rightarrow h \,??\, \mathbf{x}_n$ for some $b, h, \{\mathbf{x}_n\} = Y$. We have $[\![\mathcal{HA}(\langle\, b \rightarrow h \,??\, \mathbf{x}_n, \sigma, E\rangle)]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{ira}(h,Y)} (v_0', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0', event $\mathrm{ira}(h, Y)$, and a guard and a jump condition that hold for α and α'. According to the translation defined for the guarded receive, $\mathrm{guard}(e) = b$ and $\mathrm{jump}(e) = (\{\mathbf{x}_n\} \cup J \cup X_{\mathrm{aux}}, \bigwedge_{i=1}^{n} h_i' = x_i')$. Since $\alpha \models \mathrm{guard}(e)$ and $(\alpha, \alpha') \models \mathrm{jump}(e)$ both are true, we know that there exist ξ, ξ', σ' such that $\xi = \sigma \cup \xi^{\dot{C}L}$, where $\xi^{\dot{C}L} \in \dot{C} \rightarrow \Lambda$, $\xi' \in \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C}, \forall_{y \in \mathrm{dom}(\sigma) \setminus (J \cup \{\mathbf{x}_n\})}\, \xi(y) = \sigma(y)\}$, and $\xi'(\mathbf{x}_n) = \mathbf{h}_n$. Renaming $\mathbf{h}_n$ as cs, $\{\mathbf{x}_n\}$ as Y, and using Rule 6, we obtain $\langle h \,??\, \mathbf{x}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{ira}(h,cs,Y),\ \xi'} \langle \checkmark, \xi'_\sigma, E\rangle$, where $\xi'_{\sigma'} = \sigma'$. Using Rule 20.1, we know that $\xi \models b$ (from $\alpha \models \mathrm{guard}(e)$), and we obtain $\langle b \rightarrow h \,??\, \mathbf{x}_n, \sigma, E\rangle \xrightarrow{\xi,\ \mathrm{ira}(h,cs,Y),\ \xi'} \langle \checkmark, \xi'_\sigma, E\rangle$.

C.4.6 Theorem 5.3.5.3 - part 2

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \rightarrow \bot$, guarded action predicate, guarded send process term and guarded receive process term these constants cannot perform any action transition (with specified label $\mathrm{ira}(h, Y)$). Hence, the theorem holds trivially for these cases.

For the operators the proofs are similar to the proofs for these cases in Appendix C.4.2.

C.4.7 Theorem 5.3.5.4 - part 1

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \rightarrow \bot$, guarded action predicate, guarded send process term and guarded receive process term, these constants cannot perform any termination transition (with a communication label). Hence, the theorem holds trivially for these cases.

For the any delay operator, sequential composition, alternative composition, repetition, jump enabling, action encapsulation and urgent communication the proofs are similar to the proofs of these cases in Appendix C.4.1.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $[\![\, \mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{ca}(h)} (v_0', \alpha')$, which means that there exists an edge e with source location v_0, target location v_0', event $\mathrm{ca}(h)$, and a guard and a jump condition that hold for α and α'. According to the translation defined for the parallel composition operator, $v_0 = (v_{q_1}, v_{q_2})$, $e = (e_{q_1}, e_{q_2})$, where v_{q_1}, v_{q_2} are the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$, $e_{q_1} \in E_{q_1}$, $e_{q_2} \in E_{q_2}$; $\mathrm{target}_{q_1}(e_{q_1})$ and $\mathrm{target}_{q_2}(e_{q_2})$ are terminating locations. Then we distinguish two cases:
  - $(\mathrm{event}_{q_1}(e_{q_1}) = \mathrm{isa}(h)) \wedge (\mathrm{event}_{q_2}(e_{q_2}) = \mathrm{ira}(h, Y))$ such that $Y = \{\mathbf{x}_n\}$. Then, we also know that $[\![\, \mathcal{HA}(\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle)\,]\!] \models (v_{q_1}, \alpha) \xrightarrow{\mathrm{isa}(h)} (v_{q_1}', \alpha')$ for some v_{q_1}', W, and $[\![\, \mathcal{HA}(\langle q_2, \sigma, E\rangle)\,]\!] \models (v_{q_2}, \alpha) \xrightarrow{\mathrm{ira}(h, \{\mathbf{x}_n\})} (v_{q_2}', \alpha')$ for some v_{q_2}', where v_{q_1}' and v_{q_2}' are terminating locations. By part 1 of Theorem 5.3.5.2 we then have $\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma', (C, J \cup W, \emptyset, H, \emptyset)\rangle$ for some ξ, ξ', cs, and by part 1 of Theorem 5.3.5.3 we have $\langle q_2, \sigma, E\rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma', E\rangle$ such that $\sigma = \alpha \upharpoonright \mathrm{dom}(\sigma)$ and $\sigma' = \alpha' \upharpoonright \mathrm{dom}(\sigma')$. Using Rule 28.1.l, we obtain $\langle q_1 \parallel q_2, \sigma, E\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle \checkmark, \sigma', E\rangle$.
  - $(\mathrm{event}_{q_1}(e_{q_1}) = \mathrm{ira}(h, Y)) \wedge (\mathrm{event}_{q_2}(e_{q_2}) = \mathrm{isa}(h))$. The proofs of this case are similar to the previous case.

C.4.8 Theorem 5.3.5.4 - part 2

The proof is by induction on the structure of closed process term p. According to the translations defined for delay predicate, consistent deadlock δ, guarded inconsistent process term $b \rightarrow \bot$, guarded action predicate, guarded send process term and guarded receive process term these constants cannot perform any action transition (with specified label $ca(h, cs)$). Hence, the theorem holds trivially for these cases.

For all operators, except for the parallel composition, the proofs are similar to the proofs for these cases in Appendix C.4.2.

- Parallel composition operator $p \equiv q_1 \parallel q_2$ for some q_1 and q_2. We have $[\![\, \mathcal{HA}(\langle q_1 \parallel q_2, \sigma, E\rangle)\,]\!] \models (v_0, \alpha) \xrightarrow{\mathrm{ca}(h)} (v_0'', \alpha')$ is for some v_0, v_0'', and v_0'' has to be a non-terminating

location in $\mathcal{T}_J(q_1 \parallel q_2)$. According to the translation defined for the parallel composition operator, there is an edge (e_{q_1}, e_{q_2}) in $\mathcal{T}_J(q_1 \parallel q_2)$ with source location (v_{q_1}, v_{q_2}); $e_{q_1} \in E_{q_1}$, $e_{q_2} \in E_{q_2}$, $e_{q_1} \in E_{q_1}$, $e_{q_2} \in E_{q_2}$; $\text{target}_{q_1}(e_{q_1})$ and $\text{target}_{q_2}(e_{q_2})$ are non-terminating locations, event $\text{ca}(h)$, a guard and a jump condition that are the logical and of the guard and jump conditions of e_{q_1} and e_{q_2} and, therefore hold for α and α', and an urgency status that is the logical or of the urgency status of edges e_{q_1} and e_{q_2}. We distinguish four cases:

- $v_0'' = (v'_{q_1}, v'_{q_2})$ for some non-terminating locations v'_{q_1} from $\mathcal{T}_J(q_1)$ and v'_{q_2} from $\mathcal{T}_J(q_2)$. Then we distinguish two cases:
  - $(\text{event}_{q_1}(e_{q_1}) = \text{isa}(h)) \wedge (\text{event}_{q_2}(e_{q_2}) = \text{ira}(h, Y))$ such that $Y = \{\mathbf{x}_n\}$. Then, we also know that $[\![\mathcal{HA}(\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{\text{isa}(h)} (v'_{q_1}, \alpha')$ for some v'_{q_1}, W, and $[\![\mathcal{HA}(\langle q_2, \sigma, E\rangle)]\!] \models (v_{q_2}, \alpha) \xrightarrow{\text{ira}(h,\{\mathbf{x}_n\})} (v'_{q_2}, \alpha')$ for some v'_{q_2}, where v'_{q_1} and v'_{q_2} are the initial locations of $\mathcal{T}_J(q'_1)$ and $\mathcal{T}_J(q'_2)$ respectively. By part 2 of Theorem 5.3.5.2 we then have $\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle q'_1, \sigma', (C, J \cup W, \emptyset, H, \emptyset)\rangle$ for some ξ, ξ', cs, and by part 2 of Theorem 5.3.5.3 we have $\langle q_2, \sigma, (C, J, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle q'_2, \sigma', (C, J, \emptyset, H, \emptyset)\rangle$ such that $\sigma = \alpha \restriction \text{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \text{dom}(\sigma')$. Using Rule 28.4.l, we obtain $\langle q_1 \parallel q_2, \sigma, E\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle q'_1 \parallel q'_2, \sigma', E\rangle$.
  - $(\text{event}_{q_1}(e_{q_1}) = \text{ira}(h, \{\mathbf{x}_n\})) \wedge (\text{event}_{q_2}(e_{q_2}) = \text{isa}(h))$. The proofs of this case are similar to the previous case.
- $v_0'' = v'_{q_2}$ for some non-terminating location v'_{q_2} from $\mathcal{T}_J(q_2)$. Then we distinguish two cases:
  - $(\text{event}_{q_1}(e_{q_1}) = \text{isa}(h)) \wedge (\text{event}_{q_2}(e_{q_2}) = \text{ira}(h, Y))$ such that $Y = \{\mathbf{x}_n\}$. Then, we also know that $[\![\mathcal{HA}(\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle)]\!] \models (v_{q_1}, \alpha) \xrightarrow{\text{isa}(h)} (v'_{q_1}, \alpha')$ for some v'_{q_1}, W, and $[\![\mathcal{HA}(\langle q_2, \sigma, E\rangle)]\!] \models (v_{q_2}, \alpha) \xrightarrow{\text{ira}(h,\{\mathbf{x}_n\})} (v'_{q_2}, \alpha')$ for some v'_{q_2}, where v'_{q_1} is a terminating location of $\mathcal{T}_J(q_1)$ and v'_{q_2} is the initial location $\mathcal{T}_J(q'_2)$.
    By part 1 of Theorem 5.3.5.2 we then have $\langle q_1, \sigma, (C, J \cup W, \emptyset, H, \emptyset)\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle \checkmark, \sigma', (C, J \cup W, \emptyset, H, \emptyset)\rangle$, and by part 2 of Theorem 5.3.5.3 we have $\langle q_2, \sigma, E\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle q'_2, \sigma', E\rangle$ such that $\sigma = \alpha \restriction \text{dom}(\sigma)$ and $\sigma' = \alpha' \restriction \text{dom}(\sigma')$. Using Rule 28.3.l, we obtain $\langle q_1 \parallel q_2, \sigma, E\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle q'_2, \sigma', E\rangle$.
  - $(\text{event}_{q_1}(e_{q_1}) = \text{ira}(h, \{\mathbf{x}_n\})) \wedge (\text{event}_{q_2}(e_{q_2}) = \text{isa}(h))$ such that $Y = \{\mathbf{x}_n\}$. The proofs of this case are similar to the previous case.
- $v_0'' = v'_{q_1}$ for some non-terminating location v'_{q_1} from $\mathcal{T}_J(q_1)$. Similar to the previous case.

– $v_0'' = (v'_{q_1}, v'_{q_2})$ for some terminating locations v'_{q_1} from $\mathcal{T}_J(q_1)$ and v'_{q_2} from $\mathcal{T}_J(q_2)$. According to the translation defined for the parallel composition operator, this case has to perform a termination transition labelled with $\mathrm{ca}(h)$, and not an action transition labelled with $\mathrm{ca}(h)$. This case leads to a contradiction. Hence, the theorem holds trivially.

C.4.9 Theorem 5.3.5.5

The proof is by induction on the structure of closed process term p. According to the transitions defined for consistent deadlock δ, it cannot perform any time transition, hence the theorem holds trivially.

- Delay predicate $p \equiv u$ for some u. We have $[\![\, \mathcal{HA}(\langle u, \sigma, E\rangle)\,]\!] \models \rho' : (v_0, \alpha) \xmapsto{t} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha'$. According to the translation defined for the delay predicate, the flow condition for location v_0 (the only one location) is u. From Theorem 5.3.3, we have that $\rho \in \Omega_{FG}(\sigma, C, \emptyset, u, t)$. Using Rule 3, we obtain $\langle u, \sigma, E\rangle \xmapsto{t,\rho} \langle u, \rho_\sigma(t), E\rangle$, where $\rho_\sigma(t) = \sigma'$.

- Guarded inconsistent process term $b \to \bot$ for some b. We know that the guarded inconsistent process term $b \to \bot$ is equivalent to $\neg b$ (see also Proposition 3.5.4). It is not hard to see that the proof for this case is similar to the proof for the case of delay predicate.

- Guarded action predicate $p \equiv b \to W : r \gg l_\mathrm{a}$ for some b, W, r, l_a. We have $[\![\, \mathcal{HA}(\langle b \to W : r \gg l_\mathrm{a}, \sigma, E\rangle)\,]\!] \models \rho' : (v_0, \alpha) \xmapsto{t} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha'$, $\forall_{r\in[0,t]}\, \rho'(r) \models \mathrm{inv}(v_0) \wedge \mathrm{flow}(v_0)$, $\forall_{e \in E_{b \to W : r \gg l_\mathrm{a}}}\, (\mathrm{source}(e) = v_0 \wedge \mathrm{urgent}(e)) \implies \forall_{t\in[0,r)}\, \rho'(t) \models \neg\mathrm{guard}(e)$. According to the translation defined for the guarded action predicate, $\mathcal{T}_J(b \to W : r \gg l_\mathrm{a})$ has only one edge, let say e. We know that $\mathrm{flow}(v_0) = \mathrm{true}$, $\mathrm{inv}(v_0) = \mathrm{true}$, $\mathrm{urgent}(e) = \mathrm{true}$ and $\mathrm{guard}(e) = b$. Note that this time transition $(v_0, \alpha) \xmapsto{t} (v_0, \alpha')$ implies that $\forall_{s\in[0,t)}\, \rho'(s) \models \neg b$. Using Rule 22, we obtain $\langle b \to W : r \gg l_\mathrm{a}, \sigma, E\rangle \xmapsto{t,\rho} \langle b \to W : r \gg l_\mathrm{a}, \rho_\sigma(t), E\rangle$ for some ρ such that $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$, where $\rho_\sigma(t) = \sigma'$.

- Guarded send and guarded receive. The proofs of guarded send and guarded receive are similar to the proofs of guarded action predicate.

- Any delay operator $p \equiv [q]$ for some q. We have $[\![\, \mathcal{HA}(\langle [q], \sigma, E\rangle)\,]\!] \models \rho' : (v_0, \alpha) \xmapsto{t} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha'$, $\forall_{r\in[0,t]}\, \rho'(r) \models \mathrm{inv}(v_0) \wedge \mathrm{flow}(v_0)$, $\forall_{e\in E_{[q]}}\, (\mathrm{source}(e) = v_0 \wedge \mathrm{urgent}(e)) \implies \forall_{t\in[0,r)}\, \rho'(t) \models \neg\mathrm{guard}(e)$. According to the translation defined for the any delay operator, $\mathrm{flow}(v_0) = \mathrm{inv}(v_0) = \mathrm{true}$. So, we know that it exists a $\rho \in \Omega_{FG}(\sigma, C, \emptyset, \mathrm{true}, t)$ such that $\rho = \rho' \downarrow (\mathrm{dom}(\sigma) \cup \dot{C})$ (see also Theorem 5.3.3). Using Rule 11, we obtain $\langle [q], \sigma, E\rangle \xmapsto{t,\rho} \langle [q], \rho_\sigma(t), E\rangle$, where $\rho_\sigma(t) = \sigma'$.

- Sequential composition operator $p \equiv q_1;\ q_2$ for some q_1 and q_2. We have $[\![\ \mathcal{HA}(\ \langle\ q_1;\ q_2, \sigma, E\ \rangle)\ ]\!] \models \rho' : (v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha',\ \forall_{r\in[0,t]}\ \rho'(r) \models \text{inv}(v_0) \wedge \text{flow}(v_0),\ \forall_{e\in E_{q_1;q_2}}\ (\text{source}(e) = v_0 \wedge \text{urgent}(e)) \implies \forall_{t\in[0,r)}\ \rho(t) \models \neg\text{guard}(e)$. According to the translation defined for the sequential composition operator, v_0 is also the initial location of $\mathcal{T}_J(q_1)$ and no new edges are added into $\mathcal{T}_J(q_1;\ q_2)$ (obtained from $\mathcal{T}_J(q_1)$). Therefore, also $(v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$ is in $[\![\ \mathcal{HA}(\langle q_1, \sigma, E\rangle)\ ]\!]$. By induction we then have $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some ρ, q' . Using Rule 18, we obtain $\langle q_1;\ q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q';\ q_2, \sigma', E\rangle$.

- Alternative composition operator $p \equiv q_1 \ [\!]\ q_2$ for some q_1 and q_2. We have $[\![\ \mathcal{HA}(\langle q_1 \ [\!]\ q_2, \sigma, E\rangle)\ ]\!] \models \rho' : (v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha',\ \forall_{r\in[0,t]}\ \rho'(r) \models \text{inv}(v_0) \wedge \text{flow}(v_0),\ \forall_{e\in E_{q_1 [\!] q_2}}\ (\text{source}(e) = v_0 \wedge \text{urgent}(e)) \implies \forall_{t\in[0,r)}\ \rho'(t) \models \neg\text{guard}(e)$. According to the translation defined for the alternative composition operator, $v_0 = (v_{q_1}, v_{q_2})$ for some v_{q_1}, v_{q_2}, where v_{q_1} and v_{q_2} denote the initial locations of $\mathcal{T}_J(q_1)$ and $\mathcal{T}_J(q_2)$, respectively. Also, $(v_{q_1}, \alpha) \overset{t}{\longmapsto} (v_{q_1}, \alpha')$ is in $[\![\ \mathcal{HA}(\langle q_1, \sigma, E\rangle)\ ]\!]$ and $(v_{q_2}, \alpha) \overset{t}{\longmapsto} (v_{q_2}, \alpha')$ is in $[\![\ \mathcal{HA}(\langle q_2, \sigma, E\rangle)\ ]\!]$. By induction we then have $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q'', \sigma', E\rangle$ for some ρ, q', q'' . Using Rule 26, we obtain $\langle q_1 \ [\!]\ q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q' \ [\!]\ q'', \sigma', E\rangle$.

- Parallel composition operator. The proofs are similar to the proofs of the alternative composition operator.

- Repetition operator $p \equiv *q$ for some q. We have $[\![\ \mathcal{HA}(\langle *q, \sigma, E\rangle)\ ]\!] \models \rho' : (v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha',\ \forall_{r\in[0,t]}\ \rho'(r) \models \text{inv}(v_0) \wedge \text{flow}(v_0),\ \forall_{e\in E_{*q}}\ (\text{source}(e) = v_0 \wedge \text{urgent}(e)) \implies \forall_{t\in[0,r)}\ \rho'(t) \models \neg\text{guard}(e)$. According to the translation defined for the repetition operator, v_0 is also the initial location of $\mathcal{T}_J(q)$. Also, no new outgoing edges are added to v_0 of $\mathcal{T}_J(q)$ to obtain $\mathcal{T}_J(*q)$. Hence, $(v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$ is in $[\![\ \mathcal{HA}(\langle q, \sigma, E\rangle)\ ]\!]$. By induction we then have $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some ρ, q', σ' . Then, we obtain $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q';\ *q, \sigma', E\rangle$ using Rule B in Appendix C.1.

- Jump enabling operator $p \equiv \iota_{J^+}(q)$ for some q and J^+. We have $[\![\ \mathcal{HA}(\langle\ \iota_{J^+}(q), \sigma, E\rangle)\ ]\!] \models \rho'\ (v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha',\ \forall_{r\in[0,t]}\ \rho'(r) \models \text{inv}(v_0) \wedge \text{flow}\ (v_0),\ \forall_{e\in E_{\iota_{J^+}(q)}}\ (\text{source}(e) = v_0 \wedge \text{urgent}(e)) \implies \forall_{t\in[0,r)}\ \rho'(t) \models \neg\text{guard}(e)$. According to the translation defined for the jump enabling operator, v_0 is also the initial location of $\mathcal{T}_J(q)$. Also, $\mathcal{T}_J(\iota_{J^+}(q)) = \mathcal{T}_{J\cup J^+}(q)$. Obviously, $(v_0, \alpha) \overset{t}{\longmapsto} (v_0, \alpha')$ is in $[\![\ \mathcal{HA}(\langle q, \sigma, (C, J \cup J^+, \emptyset, H, \emptyset)\rangle)\ ]\!]$. By induction we then have $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some ρ, q', σ' . Using Rule 42, we obtain $\langle \iota_{J^+}(q), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle \iota_{J^+}(q'), \sigma', E\rangle$.

- Action encapsulation operator $p \equiv \partial_A(q)$ for some A and q. We have $[\![\ \mathcal{HA}(\langle$

$\partial_A(q), \sigma, E\rangle)\ ]\!]\ \models \rho' : (v_0, \alpha) \xrightarrow{t} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha'$, $\forall_{r\in[0,t]}\ \rho'(r) \models \text{inv}(v_0) \wedge \text{flow}(v_0)$, $\forall_{e\in E_{\partial_A(q)}}\ (\text{source}(e) = v_0 \wedge \text{urgent}(e)) \implies \forall_{t\in[0,r)}\ \rho'(t) \models \neg\text{guard}(e)$. According to the translation defined for the action encapsulation operator, $(v_0, \alpha) \xrightarrow{t} (v_0, \alpha')$ is also preserved in $[\![\, \mathcal{HA}(\langle q, \sigma, E\rangle)\ ]\!]$, because $\mathcal{T}_J(\partial_A(q))$ is obtained by replacing the jump conditions of some edges labelled with some $a \in A$ to predicates false with an empty set of variables that are allowed to change from $\mathcal{T}_J(q)$ (this has effect on action transitions). We know that v_0 is also the initial location of $\mathcal{T}_J(q)$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{t,\rho} \langle q', \sigma', E\rangle$ for some ρ, q' . Using Rule 33, we obtain $\langle \partial_A(q), \sigma, E\rangle \xrightarrow{t,\rho} \langle \partial_A(q'), \sigma', E\rangle$.

- Urgent communication operator $p \equiv \upsilon_H(q)$ for some H and q. We have $[\![\, \mathcal{HA}(\langle \upsilon_H(q), \sigma, E\rangle)\ ]\!] \models (v_0, \alpha) \xrightarrow{t} (v_0, \alpha')$, which means that there is a ρ' such that $\rho'_\alpha(0) = \alpha, \rho'_\alpha(t) = \alpha'$, $\forall_{r\in[0,t]}\ \rho'(r) \models \text{inv}(v_0) \wedge \text{flow}(v_0)$, $\forall_{e\in E_{\upsilon_H(q)}}\ (\text{source}(e) = v_0 \wedge \text{urgent}(e)) \implies \forall_{t\in[0,r)}\ \rho'(t) \models \neg\text{guard}(e)$. According to the translation defined for the urgent communication operator, the urgency status of all edges labelled with $\text{ca}(h)$ of $\mathcal{T}_J(q)$ is set to *true* to obtain $\mathcal{T}_J(\upsilon_H(q))$. Then, this time transition $(v_0, \alpha) \xrightarrow{t} (v_0, \alpha')$ implies that there is no action transition via any outgoing edge of v_0 with label $\text{ca}(h)$ is enabled, and $(v_0, \alpha) \xrightarrow{t} (v_0, \alpha')$ is also in $[\![\, \mathcal{HA}(\langle q, \sigma, E\rangle)\ ]\!]$. By induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{t,\rho} \langle q', \sigma', E\rangle$ for some ρ, q', σ'. Using Rule 37, we obtain $\langle \upsilon_H(q), \sigma, E\rangle \xrightarrow{t,\rho} \langle \upsilon_H(q'), \sigma', E\rangle$.

APPENDIX

D

Proofs of the tool support

D.1 Preliminary definition

The proofs of Theorem 6.1.1, Theorem 6.1.2 and Conjecture 6.1.1 require the notion of *norm* of a closed process term. The norm of a closed process term $p \in P_{\mathrm{T}}$ is intended to be a measure of the complexity of p.

Definition D.1.1 (Norm) *For $p, q \in P_{\mathrm{T}}$, $l_p, l_q \in \{0,1\}$ and $r_p, r_q \in \mathbb{N}$, we define the mapping $|\_| \in P_{\mathrm{T}} \to \{0,1\} \times \mathbb{N}$ inductively as follows:*

- $| p | = (0,1)$ *if* $p \in \{W : r \gg l_{\mathrm{a}}, h \,!!\, \mathbf{e}_n, h \,!!\, \mathbf{e}_n, u, \delta\}$,
- $| [p] | = | u \curvearrowright p | = | b \to p | = | \partial_A(p) | = | \upsilon_H(p) | = | \iota_{J^+}(p) | = (l_p, r_p + 1)$ *for* $| p | = (l_p, r_p)$,
- $| p;\, q | = (l_p, r_p + r_q + 1)$ *for* $| p | = (l_p, r_p)$ *and* $| q | = (l_q, r_q)$,
- $| p \,[\!]\, q | = | p \parallel q | = (\max(l_p, l_q), r_p + r_q + 1)$ *for* $| p | = (l_p, r_p)$ *and* $| q | = (l_q, r_q)$,
- $| p | = (1,0)$ *if* $p = X$.

We define the lexicographical ordering $<$ as follows: $(l_p, r_p) < (l_q, r_q)$ iff $l_p < l_q \vee (l_p = l_q \wedge r_p < r_q)$.

Furthermore, the following conjecture is needed for the proof of Conjecture 6.1.1

Conjecture D.1.1 *Let p, $p' \in P_{\mathrm{T}}$, σ, σ' be valuations, ρ be a trajectory, E be an environment and $(c_p^{[0]}, c_p^{(0,t)}, c_p^{[t]}, c_p^{[0,t]}, c_p, p') \in \mathcal{S}_{\mathrm{d}}(\langle p, E \rangle)$. Then,*

$$\rho(0) \models c_p^{[0]} \Rightarrow \langle p, \sigma, E \rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma', E \rangle.$$

D.2 Proof of Theorem 6.1.1

Let $p \in P_{\mathrm{T}}$, σ be a valuation, ξ be an extended valuation such that $\xi \restriction \mathrm{dom}(\sigma)$, and E be an environment. Then

$$\xi \models \mathcal{C}_{\mathrm{c}}(p, E) \Rightarrow \langle p, \sigma, E \rangle \overset{\xi}{\rightsquigarrow} .$$

PROOF. We prove this theorem by induction on the norm of p. The proofs for the action predicate, send process term, receive process term, consistent deadlock and the any delay operator are trivial, because these atomic process terms and the any delay operator are consistent with any extended valuation with respect to σ in any environment (see also Rules 2, 7, 8, 9 and 12). We assume $\xi \models \mathcal{C}_c(p, E)$ and $E = (C, J, L, H, R)$ for some C, J, L, H, R. Then,

- $p \equiv u$ for some u. According to the definition of the function $\mathcal{C}_c$, we know that $\mathcal{C}_c(u, E) = u$. Since $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$, using Rule 4, we get $(C, J, L, H, R) \Vdash \langle u, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}$.

- $p \equiv u \curvearrowright q$ for some u and q. According to the definition of the function $\mathcal{C}_c$, we know that $\mathcal{C}_c(u \curvearrowright q, E) = u \wedge \mathcal{C}_c(q, E)$. So, $\xi \models u$, and $\xi \models \mathcal{C}_c(q, E)$, by induction, we then have $\langle q, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 15, we have $\langle u \curvearrowright q, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$.

- $p \equiv q;\ r$ for some q and r. According to the definition of the function $\mathcal{C}_c$, we know that $\mathcal{C}_c(q;\ r, E) = \mathcal{C}_c(q, E)$. So, $\xi \models \mathcal{C}_c(q, E)$, by induction, we then have $\langle q, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we have $\langle q;\ r, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$.

- $p \equiv b \to q$ for some guard b and q. According to the definition of the function $\mathcal{C}_c$, we get $\mathcal{C}_c(b \to q, E) = (b \wedge \mathcal{C}_c(q, E)) \vee \neg b$. Then, we distinguish two cases:

  - either $b \wedge \mathcal{C}_c(q, E)$. So, $\xi \models b$ and $\xi \models \mathcal{C}_c(q, E)$, by induction, we then have $\langle q, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 23, we have $\langle b \to q, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$.
  - or $\neg b$. So, $\xi \models \neg b$, and then also $\sigma \cup \xi^{\dot{C}L} \models \neg b$ for some $\xi^{\dot{C}L}$. Using Rule 24, we obtain $(C, J, L, H, R) \Vdash \langle b \to q, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}$.

- $p \equiv q \between r$ for some q and r. According to the definition of the function $\mathcal{C}_c$, we know that $\mathcal{C}_c(q \between r, E) = \mathcal{C}_c(q, E) \wedge \mathcal{C}_c(r, E)$. So, $\xi \models \mathcal{C}_c(q, E)$ and $\xi \models \mathcal{C}_c(r, E)$, by induction, we then have $\langle q, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$ and $\langle r, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we obtain $\langle q \between r, \sigma, E \rangle \overset{\xi}{\rightsquigarrow}$.

- $p \equiv q \parallel r$. The proof is similar to the proof of the case that $p \equiv q \between r$.

- $p \equiv \partial_A(q)$ for some A and q. The proof is similar to the proof of the case that $p \equiv q;\ r$.

- $p \equiv \upsilon_H(q)$ for some H and q. The proof is similar to the proof of the case that $p \equiv q;\ r$.

- $p \equiv X$ for some X. The proof is similar to the proof of the case that $p \equiv q;\ r$.

- $p \equiv \iota_{J^+}(q)$ for some q and set J^+. The proof is trivial and is similar to the proof of the case that $p \equiv q;\ r$.

D.3 Proof of Theorem 6.1.2

Let p, $p' \in P_{\mathrm{T}}$, σ, σ' be valuations, ξ, ξ' be extended valuations, E be an environment, and a be an action label. Then

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma', E\rangle \in \mathrm{Tr_a}(p,\sigma,E) \Rightarrow \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma', E\rangle.$$

PROOF. We prove this theorem by induction on the norm of p.

Firstly, we give the proofs for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle \in \mathrm{Tr_a}(p,\sigma,E) \Rightarrow \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$. Since there are no termination transition rules defined for delay predicate, consistent deadlock and sequential composition, the theorem holds trivially for these cases. To increase the readability of the proofs, some irrelevant information for the proofs is omitted. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle \in \mathrm{Tr_a}(p,\sigma,E)$ and $E = (C, J, L, H, R)$ for some C, J, L, H, R. Then,

- $p \equiv W : r \gg l_{\mathrm{a}}$ for some W, r, l_{a} and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}\gg}, \xi')$ for some $l_{\mathrm{a}\gg}$. According to the definition of the function $\mathrm{Tr_a}$, we know that there exist $c_{\gg}, W_{\gg}, r_{\gg}, C^b_{\gg}, C^a_{\gg}, p_{\gg}$ such that $(c_{\gg}, W_{\gg}, r_{\gg}, l_{\mathrm{a}\gg}, C^b_{\gg}, C^a_{\gg}, p_{\gg}) \in \mathcal{S}_{\mathrm{a}}(\langle W : r \gg l_{\mathrm{a}}, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_{\gg}, \xi' \in \Xi(\sigma, C, J \cup W_{\gg}, L), \xi^- \cup \xi' \models r_{\gg}, \xi \models C^b_{\gg}$, and $\xi' \models C^a_{\gg}$. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle W : r \gg l_{\mathrm{a}}, E\rangle)$, we know that $\mathcal{S}_{\mathrm{a}}(\langle W : r \gg l_{\mathrm{a}}, E\rangle) = \{(\mathrm{true}, W, r, l_{\mathrm{a}}, \mathrm{true}, \mathrm{true}, \checkmark)\}$ such that $c_{\gg} = \mathrm{true}, W_{\gg} = W, r_{\gg} = r, l_{\mathrm{a}\gg} = l_{\mathrm{a}}, C^b_{\gg} = \mathrm{true}, C^a_{\gg} = \mathrm{true}$, and $p_{\gg} = \checkmark$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models \mathrm{true}, \xi' \in \Xi(\sigma, C, J \cup W, L)$, and $\xi^- \cup \xi' \models r$. Using Rule 1, we have $(C, J, L, H, R) \Vdash \langle W : r \gg l_{\mathrm{a}}, \sigma\rangle \xrightarrow{\xi,\ l_{\mathrm{a}},\ \xi'} \langle \checkmark, \xi'_\sigma\rangle$ and $\sigma' = \xi'_\sigma$.

- $p \equiv h \,!!\, \mathbf{e}_n$ for some h, $\mathbf{e}_n$, and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}!!}, \xi')$ for some $l_{\mathrm{a}!!}$. According to the definition of the function $\mathrm{Tr_a}$, we know that there exist $c_{!!}, W_{!!}, r_{!!}, C^b_{!!}, C^a_{!!}, p_{!!}$ such that $(c_{!!}, W_{!!}, r_{!!}, l_{\mathrm{a}!!}, C^b_{!!}, C^a_{!!}, p_{!!}) \in \mathcal{S}_{\mathrm{a}}(\langle h \,!!\, \mathbf{e}_n, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_{!!}, \xi' \in \Xi(\sigma, C, J \cup W_{!!}, L), \xi^- \cup \xi' \models r_{!!}, \xi \models C^b_{!!}$, and $\xi' \models C^a_{!!}$. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle h \,!!\, \mathbf{e}_n, E\rangle)$, we know that $\mathcal{S}_{\mathrm{a}}(\langle h \,!!\, \mathbf{e}_n, E\rangle) = \{(\mathrm{true}, \emptyset, \mathrm{true}, \mathrm{isa}(h, [\mathbf{e}_n]), \mathrm{true}, \mathrm{true}, \checkmark)\}$ such that $c_{!!} = \mathrm{true}, W_{!!} = \emptyset, r_{!!} = \mathrm{true}$, $l_{\mathrm{a}!!} = \mathrm{isa}(h, [\mathbf{e}_n]), C^b_{!!} = \mathrm{true}$, $C^a_{!!} = \mathrm{true}$, and $p_{!!} = \checkmark$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi' \in \Xi(\sigma, C, J, L)$, and $\xi^- \cup \xi' \models \mathrm{true}$. Using Rule 5, we have $(C, J, L, H, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma\rangle \xrightarrow{\xi,\ \mathrm{isa}(h,[\xi(\mathbf{e}_n)]),\ \xi'} \langle \checkmark, \xi'_\sigma\rangle$ and $\sigma' = \xi'_\sigma$.

- $p \equiv h \,??\, \mathbf{x}_n$ for some h, $\mathbf{x}_n$ and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}??}, \xi')$ for some $l_{\mathrm{a}??}$. According to the definition of the function $\mathrm{Tr_a}$, we know that there exist $c_{??}, W_{??}, r_{??}, l_{\mathrm{a}??}, C^b_{??}, C^a_{??}, p_{??}$ such that $(c_{??}, W_{??}, r_{??}, l_{\mathrm{a}??}, C^b_{??}, C^a_{??}, p_{??}) \in \mathcal{S}_{\mathrm{a}}(\langle h \,??\, \mathbf{x}_n, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_{??}, \xi' \in \Xi(\sigma, C, J \cup W_{??}, L), \xi^- \cup \xi' \models r_{??}, \xi \models C^b_{??}$, and $\xi' \models C^a_{??}$. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle h \,??\, \mathbf{x}_n, E\rangle)$, we know that $\mathcal{S}_{\mathrm{a}}(\langle h \,??\, \mathbf{x}_n, E\rangle) = \{(\mathrm{true}, \{\mathbf{x}_n\},$

true, $\mathrm{ira}(h, \{\mathbf{x}_n\}), \mathrm{true}, \mathrm{true}, \checkmark)\}$ such that $c_{??} = \mathrm{true}, W_{??} = \{\mathbf{x}_n\}$, $r_{??} = \mathrm{true}$, $l_{\mathrm{a}??} = \mathrm{ira}(h, \{\mathbf{x}_n\}), C^b_{??} = \mathrm{true}$, $C^a_{??} = \mathrm{true}$, and $p_{??} = \checkmark$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L)$. From the definition of the function $\mathcal{M}_{\mathrm{tr}}$ and using Rule 6, we have $(C, J, L, H, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma\rangle \xrightarrow{\xi,\ \mathrm{ira}(h,[\xi'(\mathbf{x}_n]),\{\mathbf{x}_n\}),\ \xi'} \langle \checkmark, \xi'_\sigma\rangle$ and $\sigma' = \xi'_\sigma$.

- $p \equiv [q]$ for some q, and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}[q]}, \xi')$ for some $l_{\mathrm{a}[q]}$. According to the definition of the function Tr_{a}, we know that there exist $c_{[q]}, W_{[q]}$, $r_{[q]}$, $C^b_{[q]}, C^a_{[q]}$, $p_{[q]}$ such that $(c_{[q]}, W_{[q]}, r_{[q]}, l_{\mathrm{a}[q]}, C^b_{[q]}, C^a_{[q]}, p_{[q]}) \in \mathcal{S}_{\mathrm{a}}(\langle [q], E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_{[q]}, \xi' \in \Xi(\sigma, C, J \cup W_{[q]}, L), \xi^- \cup \xi' \models r_{[q]}, \xi \models C^b_{[q]}$, $\xi' \models C^a_{[q]}$, and $p_{[q]} = \checkmark$ necessarily. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle [q], E\rangle)$, we know that $\mathcal{S}_{\mathrm{a}}(\langle [q], E\rangle) = \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle)$. So, it is not hard to see that $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle \in \mathrm{Tr}_{\mathrm{a}}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$. Using Rule 10.1, we obtain $\langle [q], \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$.

- $p \equiv u \curvearrowright q$ for some u, q and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}\curvearrowright}, \xi')$ for some $l_{\mathrm{a}[q]}$. According to the definition of the function Tr_{a}, we know that there exist $c_\curvearrowright, W_\curvearrowright$, $r_\curvearrowright$, $C^b_\curvearrowright, C^a_\curvearrowright$, $p_\curvearrowright$ such that $(c_\curvearrowright, W_\curvearrowright, r_\curvearrowright, l_{\mathrm{a}\curvearrowright}, C^b_\curvearrowright, C^a_\curvearrowright, p_\curvearrowright) \in \mathcal{S}_{\mathrm{a}}(\langle u \curvearrowright q, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_\curvearrowright, \xi' \in \Xi(\sigma, C, J \cup W_\curvearrowright, L), \xi^- \cup \xi' \models r_\curvearrowright, \xi \models C^b_\curvearrowright$, $\xi' \models C^a_\curvearrowright$, and $p_\curvearrowright = \checkmark$ necessarily. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle u \curvearrowright q, E\rangle)$, we know there exist c_q, W_q, $r_q, l_{\mathrm{a}q}$, C^b_q, C^a_q, p_q such that $(c_q, W_q, r_q, l_{\mathrm{a}q}, C^b_q, C^a_q, p_q) \in \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle)$ with $c_\curvearrowright = u \wedge c_q$, $W_\curvearrowright = W_q$, $r_\curvearrowright = r_q$, $l_{\mathrm{a}\curvearrowright} = l_{\mathrm{a}q}$, $C^b_\curvearrowright = C^b_q$, $C^a_\curvearrowright = C^a_q$, and $p_\curvearrowright = p_q$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_q, \xi' \in \Xi(\sigma, C, J \cup W_q, L), \xi^- \cup \xi' \models r_q, \xi \models C^b_q$, $\xi' \models C^a_q$, and $\xi \models u$. So, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle \in \mathrm{Tr}_{\mathrm{a}}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$. Using Rule 13.1 and $\xi \models u$, we obtain $\langle u \curvearrowright q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E\rangle$.

- $p \equiv b \to q$ for some guard b and q. The proof is similar to the proof of the case that $p \equiv u \curvearrowright q$, because both definitions of $\mathcal{S}_{\mathrm{a}}$ and deduction rules for termination transition for both operators are similar.

- $p \equiv q \parallel r$ for some q, r, and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}\parallel}, \xi')$ for some $l_{\mathrm{a}\parallel}$. According to the definition of the function Tr_{a}, we know that there exist $c_\parallel, W_\parallel$, $r_\parallel$, $C^b_\parallel, C^a_\parallel$, $p_\parallel$ such that $(c_\parallel, W_\parallel, r_\parallel, l_{\mathrm{a}\parallel}, C^b_\parallel, C^a_\parallel, p_\parallel) \in \mathcal{S}_{\mathrm{a}}(\langle q \parallel r, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_\parallel, \xi' \in \Xi(\sigma, C, J \cup W_\parallel, L), \xi^- \cup \xi' \models r_\parallel, \xi \models C^b_\parallel$, $\xi' \models C^a_\parallel$, and $p_\parallel = \checkmark$ necessarily. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle q \parallel r, E\rangle)$, we can distinguish two cases:

  - there exist c_q, W_q, $r_q, l_{\mathrm{a}q}$, C^b_q, C^a_q, p_q such that $(c_q, W_q, r_q, l_{\mathrm{a}q}, C^b_q, C^a_q, p_q) \in \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle)$ with $c_\parallel = c_q$, $W_\parallel = W_q$, $r_\parallel = r_q$, $l_{\mathrm{a}\parallel} = l_{\mathrm{a}q}$, $C^b_\parallel = C^b_q \wedge \mathcal{C}_{\mathrm{c}}(r, E)$, $C^a_\parallel = C^a_q$, and $p_\parallel = p_q$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_q, \xi' \in \Xi(\sigma, C, J \cup$

$W_q, L), \xi^- \cup \xi' \models r_q, \xi \models C_q^b,\ \xi' \models C_q^a$, and $\xi \models \mathcal{C}_{\mathrm{c}}(r, E)$. So, $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma', E\rangle \in \mathrm{Tr}_{\mathrm{a}}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma', E\rangle$. From Theorem 6.1.1 and $\xi \models \mathcal{C}_{\mathrm{c}}(r, E)$ and $\xi \upharpoonright \mathrm{dom}(\sigma)$ (see also Lemma 3.5.1), we know that $\langle r, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 25.1, we obtain $\langle q \between r, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma', E\rangle$.

– there exist $c_r, W_r,\ r_r, l_{\mathrm{a}r},\ C_r^b, C_r^a,\ p_r$ such that $(c_r,\ W_r, r_r,\ l_{\mathrm{a}r},\ C_r^b,\ C_r^a,\ p_r) \in \mathcal{S}_{\mathrm{a}}(\langle r, E\rangle)$ with $c_{\between} = c_r$, $W_{\between} = W_r$, $r_{\between} = r_r$, $l_{\mathrm{a}\between} = l_{\mathrm{a}r}$, $C_{\between}^b = C_r^b \wedge \mathcal{C}_{\mathrm{c}}(q, E)$, $C_{\between}^a = C_r^a$, and $p_{\between} = p_r$. The proof of this case is similar to the previous case.

- $p \equiv q \parallel r$ for some q and r, and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}\parallel}, \xi')$ for some $l_{\mathrm{a}\parallel}$. According to the definition of the function Tr_{a}, we know that there exist $c_\parallel, W_\parallel,\ r_\parallel,\ C_\parallel^b, C_\parallel^a,\ p_\parallel$ such that $(c_\parallel,\ W_\parallel, r_\parallel,\ l_{\mathrm{a}\parallel},\ C_\parallel^b,\ C_\parallel^a,\ p_\parallel) \in \mathcal{S}_{\mathrm{a}}(\langle q \parallel r, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_\parallel, \xi' \in \Xi(\sigma, C, J \cup W_\parallel, L), \xi^- \cup \xi' \models r_\parallel, \xi \models C_\parallel^b, \xi' \models C_\parallel^a$, and $p_\parallel = \surd$ necessarily. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle q \parallel r, E\rangle)$, we can distinguish more cases. Since the proofs for all cases are similar, we only give the proof for the following case that the left argument of the parallel composition performs a send action, right argument of the parallel composition performs a receive action, and this leads to communication between the left argument and right argument of the parallel composition.

  – there exist $c_q, W_q,\ r_q, l_{\mathrm{a}q},\ C_q^b, C_q^a,\ p_q$ such that $(c_q,\ W_q, r_q,\ l_{\mathrm{a}q},\ C_q^b,\ C_q^a,\ p_q) \in \mathcal{S}_{\mathrm{a}}(\langle q, (C, J \cup \{\mathbf{x}_n\}, L, H, R)\rangle)$; and $c_r, W_r,\ r_r, l_{\mathrm{a}r},\ C_r^b, C_r^a,\ p_r$ such that $(c_r,\ W_r, r_r,\ l_{\mathrm{a}r},\ C_r^b,\ C_r^a,\ p_r) \in \mathcal{S}_{\mathrm{a}}(\langle r, E\rangle)$ with $l_{\mathrm{a}q} = \mathrm{isa}(h, [\mathbf{e}_n])$, $l_{\mathrm{a}r} = \mathrm{ira}(h, \{\mathbf{x}_n\})$, $p_q = p_r = \surd$, $c_\parallel = c_p \wedge c_q$, $W_\parallel = \{\mathbf{x}_n\}$, $r_\parallel = r_p \wedge r_q \wedge \mathbf{x}_n = \mathbf{e}_n^-$, $l_{\mathrm{a}\parallel} = \mathrm{ca}(h, [\mathbf{e}_n])$, $C_\parallel^b = C_p^b \wedge C_q^b,\ C_\parallel^a = C_q^a \wedge C_r^a,\ p_\parallel = \surd$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_q, \xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L), \xi^- \cup \xi' \models r_q, \xi \models C_q^b$, and $\xi' \models C_q^a$. So, $\langle q, \sigma, (C, J \cup \{\mathbf{x}_n\}, L, H, R)\rangle \xrightarrow{\xi, l_{\mathrm{a}q}, \xi'} \langle \surd, \sigma', (C, J \cup \{\mathbf{x}_n\}, L, H, R)\rangle \in \mathrm{Tr}_{\mathrm{a}}(\langle q, \sigma, (C, J \cup \{\mathbf{x}_n\}, L, H, R)\rangle)$, by induction and the definition of the function $\mathcal{M}_{\mathrm{tr}}$ we then have $(C, J \cup \{\mathbf{x}_n\}, L, H, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi, \mathrm{isa}(h, [\xi(\mathbf{e}_n)]), \xi'} \langle \surd, \sigma'\rangle$. Also, $\xi \models c_r, \xi' \in \Xi(\sigma, C, J \cup W_r, L)$ (see also $W_r = \{\mathbf{x}_n\}$), $\xi^- \cup \xi' \models r_r, \xi \models C_r^b$, and $\xi' \models C_r^a$. So, $\langle r, \sigma, (C, J \cup W_r, L, H, R)\rangle \xrightarrow{\xi, l_{\mathrm{a}r}, \xi'} \langle \surd, \sigma', (C, J \cup W_r, L, H, R)\rangle \in \mathrm{Tr}_{\mathrm{a}}(\langle r, \sigma, (C, J \cup W_r, L, H, R)\rangle)$, by induction and the definition of the function $\mathcal{M}_{\mathrm{tr}}$, we then have $\langle r, \sigma, (C, J \cup W_r, L, H, R)\rangle \xrightarrow{\xi, \mathrm{ira}(h, [\xi'(\mathbf{x}_n)], \{\mathbf{x}_n\}), \xi'} \langle \surd, \sigma', (C, J \cup W_r, L, H, R)\rangle$. From the definition of $\mathcal{M}_{\mathrm{tr}}$, we have $[\xi(\mathbf{e}_n)] = [\xi'(\mathbf{x}_n)]$. Using Rule 28.1.l, we obtain $(C, J \cup W_r, L, H, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi, \mathrm{ca}(h, [\xi(\mathbf{e}_n)], \xi'} \langle \surd, \sigma'\rangle$.

- $p \equiv \partial_A(q)$ for some A, q and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}\partial}, \xi')$ for some $l_{\mathrm{a}\partial}$. According to the definition of the function Tr_{a}, we know that there exist $c_\partial, W_\partial,\ r_\partial, l_{\mathrm{a}\partial},\ C_\partial^b, C_\partial^a,\ p_\partial$ such that $(c_\partial,\ W_\partial, r_\partial,\ l_{\mathrm{a}\partial},\ C_\partial^b,\ C_\partial^a,\ p_\partial) \in \mathcal{S}_{\mathrm{a}}(\langle \partial_A(q), E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_\partial, \xi' \in \Xi(\sigma, C, J \cup W_\partial, L), \xi^- \cup \xi' \models r_\partial, \xi \models C_\partial^b,\ \xi' \models C_\partial^a$, and $p_\partial = \surd$ necessarily. From the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle \partial_A(q), E\rangle)$, we know there exist $c_q, W_q,\ r_q, l_{\mathrm{a}q},\ C_q^b, C_q^a,$

p_q such that $(c_q, W_q, r_q, l_{aq}, C^b_q, C^a_q, p_q) \in \mathcal{S}_a(\langle q, E\rangle)$, action label with $l_{aq} \notin A$ with $c_\partial = c_q$, $W_\partial = W_q$, $r_\partial = r_q$, $l_{a\partial} = l_{aq}$, $C^b_\partial = C^b_q$, $C^a_\partial = C^a_q$, and $p_\partial = p_q$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_q, \xi' \in \Xi(\sigma, C, J \cup W_q, L), \xi^- \cup \xi' \models r_q, \xi \models C^b_q$, and $\xi' \models C^a_q$, and $a \notin A$. So, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E\rangle \in \mathrm{Tr}_a(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E\rangle$. Using Rule 32.1, we obtain $\langle \partial_A(q), \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E\rangle$.

- $p \equiv \upsilon_H(q)$ for some H and q. The proof is similar to the proof of the case that $p \equiv [q]$.
- $p \equiv X$ for some X. The proof is similar to the proof of the case that $p \equiv q;\ r$.
- $p \equiv \iota_{J^+}(q)$ for some q and set J^+. The proof is trivial and it is similar to the proof of the case that $p \equiv q;\ r$.

Secondly, we give the proofs for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle \in \mathrm{Tr}_a(p, \sigma, E) \Rightarrow \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$. Since there are no action transition rules defined for action predicate, send process term, receive process term, delay predicate and consistent deadlock, the theorem holds trivially for these cases. The proofs for the any delay operator, signal emission operator, guard operator, alternative composition operator, action encapsulation operator, urgent communication operator and jump enabling operator are similar to the proofs of these operators in the proof of $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E'\rangle \in \mathrm{Tr}_a(p, \sigma, E) \Rightarrow \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E'\rangle$. To increase the readability of the proofs, some irrelevant information for the proofs is omitted. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle \in \mathrm{Tr}_a(p, \sigma, E)$ and $E = (C, J, L, H, R)$ for some C, J, L, H, R. Then,

- $p \equiv q;\ r$ for some q, r and $a = \mathcal{M}_{tr}(\xi, l_a;\ , \xi')$ for some $l_a;\ $. According to the definition of the function Tr_a, we know that there exist $c;\ , W;\ ,\ r;\ ,$ and $C^b;\ , C^a;\ $ such that $(c;\ , W;\ , r;\ , l_a;\ , C^b;\ , C^a;\ , p') \in \mathcal{S}_a(\langle q;\ r, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c;\ , \xi' \in \Xi(\sigma, C, J \cup W;\ , L), \xi^- \cup \xi' \models r;\ , \xi \models C^b;\ $ and $\xi' \models C^a;\ $. According to the definition of the function $\mathcal{S}_a(\langle q;\ r, E\rangle)$, we can distinguish two case:
  - there exist $c_q, W_q,\ r_q, l_{aq},\ C^b_q,\ C^a_q,\ p_q$ such that $(c_q,\ W_q, r_q,\ l_{aq},\ C^b_q,\ C^a_q,\ p_q) \in \mathcal{S}_a(\langle q, E\rangle)$, $c;\ = c_q$, $W;\ = W_q$, $r;\ = r_q$, $l_a;\ = l_{aq}$, $C^b;\ = C^b_q$, $C^a;\ = C^a_q \wedge \mathcal{C}_c(r, E)$, $p_q = \surd$ and $p' = r$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_q, \xi' \in \Xi(\sigma, C, J \cup W_q, L), \xi^- \cup \xi' \models r_q, \xi \models C^b_q$, $\xi' \models C^a_q$, and $\xi' \models \mathcal{C}_c(r, E)$. So, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E\rangle \in \mathrm{Tr}_a(q, \sigma, E)$, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma', E\rangle$ (see also the previous proofs of this theorem). From Theorem 6.1.1 and $\xi' \models \mathcal{C}_c(r, E)$, we know that $\langle r, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 16, we obtain $\langle q;\ r, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle r, \sigma', E\rangle$.
  - there exist $c_q, W_q,\ r_q, l_{aq},\ C^b_q,\ C^a_q$ and p' such that $(c_q,\ W_q, r_q,\ l_{aq},\ C^b_q,\ C^a_q,\ p_q) \in \mathcal{S}_a(\langle q, E\rangle)$ with $c;\ = c_q$, $W;\ = W_q$, $r;\ = r_q$, $l_a;\ = l_{aq}$, $C^b;\ = C^b_q$, $C^a;\ = C^a_q$, $p_q \neq \surd$ and $p' = p_q;\ r$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models$

$c_q, \xi' \in \Xi(\sigma, C, J \cup W_q, L), \xi^- \cup \xi' \models r_q, \xi \models C_q^b$, and $\xi' \models C_q^a$. So, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p_q, \sigma', E\rangle \in \mathrm{Tr_a}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p_q, \sigma', E\rangle$. Using Rule 17, we obtain $\langle q;\ r, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p_q;\ r, \sigma', E\rangle$.

- $p \equiv q \parallel r$ for some q, r, and $a = \mathcal{M}_{\mathrm{tr}}(\xi, l_{\mathrm{a}\parallel}, \xi')$ for some $l_{\mathrm{a}\parallel}$. According to the definition of the function $\mathrm{Tr_a}$, we know that there exist $c_\parallel, W_\parallel, r_\parallel, C_\parallel^b, C_\parallel^a, p'$ such that $(c_\parallel, W_\parallel, r_\parallel, l_{\mathrm{a}\parallel}, C_\parallel^b, C_\parallel^a, p') \in \mathcal{S}_{\mathrm{a}}(\langle q \parallel r, E\rangle)$, $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_\parallel, \xi' \in \Xi(\sigma, C, J \cup W_\parallel, L), \xi^- \cup \xi' \models r_\parallel, \xi \models C_\parallel^b$, and $\xi' \models C_\parallel^a$. According to the definition of the function $\mathcal{S}_{\mathrm{a}}(\langle q \parallel r, E\rangle)$, we can distinguish more cases. Since the proofs for most cases are similar, we only give the proof for the following case :
  - there exist $c_q, W_q, r_q, l_{\mathrm{a}q}, C_q^b, C_q^a, p_q$ such that $(c_q, W_q, r_q, l_{\mathrm{a}q}, C_q^b, C_q^a, p_q) \in \mathcal{S}_{\mathrm{a}}(\langle q, E\rangle)$ with $c_\parallel = c_q$, $W_\parallel = W_q$, $r_\parallel = r_q$, $l_{\mathrm{a}\parallel} = l_{\mathrm{a}q}$, $C_\parallel^b = C_q^b \wedge \mathcal{C}_{\mathrm{c}}(r, E)$, $C_\parallel^a = C_q^a \wedge \mathcal{C}_{\mathrm{c}}(r, E)$, $p_q \neq \checkmark$ and $p' = p_q \parallel r$. It is not hard to see that we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi \models c_q, \xi' \in \Xi(\sigma, C, J \cup W_q, L), \xi^- \cup \xi' \models r_q, \xi \models C_q^b$, $\xi' \models C_q^a$, $\xi \models \mathcal{C}_{\mathrm{c}}(r, E)$, and $\xi' \models \mathcal{C}_{\mathrm{c}}(r, E)$. So, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p_q, \sigma', E\rangle \in \mathrm{Tr_a}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p_q, \sigma', E\rangle$. From Theorem 6.1.1, $\xi \models \mathcal{C}_{\mathrm{c}}(r, E)$ and $\xi' \models \mathcal{C}_{\mathrm{c}}(r, E)$, we know that $\langle r, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, and $\langle r, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.l, we obtain $\langle q \parallel r, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p_q \parallel r, \sigma', E\rangle$.

D.4 Proof of Conjecture 6.1.1

Let $p, p' \in P_{\mathrm{T}}$, σ, σ' be valuations, $t \in T$, ρ be a trajectory, and E be an environment. Then

$$\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle \in \mathrm{Tr_d}(p, \sigma, E) \Rightarrow \langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle.$$

PROOF. We prove this conjecture by induction on the norm of p. Since there are no time transition rules defined for action predicate, send process term, receive process term and consistent deadlock, the theorem holds trivially for these cases. To increase the readability of the proofs, some irrelevant information for the proofs is omitted. We assume $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle \in \mathrm{Tr_d}(p, \sigma, E)$ and $E = (C, J, L, H, R)$ for some C, J, L, H, R. Then,

- $p \equiv u$ for some u. According to the definition of the function $\mathrm{Tr_d}$, there exist $c_u^{[0]}$, $c_u^{(0,t)}$, $c_u^{[t]}$, $c_u^{[0,t]}$, c_u such that $(c_u^{[0]}, c_u^{(0,t)}, c_u^{[t]}, c_u^{[0,t]}, c_u, p') \in \mathcal{S}_{\mathrm{d}}(\langle u, E\rangle)$, $\rho \in \Omega_{FG}(\sigma, C, L, c_u^{[0,t]}, t)$, $\rho(0) \models c_u^{[0]}, \forall_{s\in(0,t)}\rho(s) \models c_u^{(0,t)}$, $\rho(t) \models c_u^{[t]}$, and $\exists_{s\in[0,t]}\rho(s) \models c_u$. We also know that $\mathcal{S}_{\mathrm{d}}(\langle u, E\rangle) = \{(u, u, u, \mathrm{true}, u)\}$ with $c_u^{[0]} = u$, $c_u^{(0,t)} = u$, $c_u^{[t]} = u$, $c_u^{[0,t]} = u$, $c_u = \mathrm{true}$ and $p' \equiv u$. It is not hard to see that $\rho \in \Omega_{FG}(\sigma, C, L, u, t)$. Using Rule 3, we get $(C, J, L, H, R) \Vdash \langle u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle u, \rho_\sigma(t)\rangle$ and $\sigma' = \rho_\sigma(t)$.

- $p \equiv [q]$ for some q. According to the definition of the function $\mathrm{Tr_d}$, there exists $c^{[0]}_{[q]}, c^{(0,t)}_{[q]}, c^{[t]}_{[q]}, c^{[0,t]}_{[q]}, c_{[q]}$ such that $(c^{[0]}_{[q]},\ c^{(0,t)}_{[q]},\ c^{[t]}_{[q]},\ c^{[0,t]}_{[q]}, c_{[q]}, p') \in \mathcal{S}_\mathrm{d}([q], E),\ \rho \in \Omega_{FG}(\sigma, C, L, c^{[0,t]}_{[q]}, t),\ \rho(0) \models c^{[0]}_{[q]}, \forall_{s\in(0,t)}\rho(s) \models c^{(0,t)}_{[q]},\ \rho(t) \models c^{[t]}_{[q]}$, and $\exists_{s\in[0,t]}\rho(s) \models c_{[q]}$. We also know that $\mathcal{S}_\mathrm{d}(\langle [q], E\rangle) = \{(\mathrm{true}, \mathrm{true}, \mathrm{true}, \mathrm{true}, \mathrm{true}, [q])\}$ with $c^{[0]}_{[q]} = \mathrm{true}$, $c^{(0,t)}_{[q]} = \mathrm{true}$, $c^{[t]}_{[q]} = \mathrm{true}$, $c^{[0,t]}_{[q]} = \mathrm{true}$, $c_{[q]} = \mathrm{true}$ and $p' \equiv [q]$. It is not hard to see that $\rho \in \Omega_{FG}(\sigma, C, L, \mathrm{true}, t)$. Using Rule 11, we get $(C, J, L, H, R) \Vdash \langle [q], \sigma\rangle \xmapsto{t,\rho} \langle [q], \rho_\sigma(t)\rangle$ and $\sigma' = \rho_\sigma(t)$.

- $p \equiv u \curvearrowright q$ for some u and q. According to function $\mathrm{Tr_d}$, we know that there exist $c^{[0]}_{\curvearrowright},\ c^{(0,t)}_{\curvearrowright},\ c^{[t]}_{\curvearrowright},\ c^{[0,t]}_{\curvearrowright}, c_{\curvearrowright}$ such that $(c^{[0]}_{\curvearrowright},\ c^{(0,t)}_{\curvearrowright},\ c^{[t]}_{u\curvearrowright q},\ c^{[0,t]}_{\curvearrowright}, c_{\curvearrowright}, p') \in \mathcal{S}_\mathrm{d}(\langle u \curvearrowright q, E\rangle)$, $\rho \in \Omega_{FG}(\sigma, C, L, c^{[0,t]}_{u\curvearrowright q}, t),\ \rho(0) \models c^{[0]}_{\curvearrowright}, \forall_{s\in(0,t)}\rho(s) \models c^{(0,t)}_{\curvearrowright},\ \rho(t) \models c^{[t]}_{\curvearrowright}$, and $\exists_{s\in[0,t]}\rho(s) \models c_{\curvearrowright}$ From the definition of $\mathcal{S}_\mathrm{d}(\langle u \curvearrowright q\rangle, E)$, we know that there exist $c^{[0]}_q,\ c^{(0,t)}_q, c^{[t]}_q,\ c^{[0,t]}_q, c_q, q'$ such that $(c^{[0]}_q,\ c^{(0,t)}_q, c^{[t]}_q,\ c^{[0,t]}_q, c_q, q') \in \mathcal{S}_\mathrm{d}(q, E)$, $c^{[0]}_{\curvearrowright} = u \wedge c^{[0]}_q$, $c^{(0,t)}_{\curvearrowright} = c^{(0,t)}_q$, $c^{[t]}_{\curvearrowright} = c^{[t]}_q$, and $c^{[0,t]}_{\curvearrowright} = c^{[0,t]}_q$, $c_{\curvearrowright} = c_q$ and $p' \equiv q'$. It is not hard to see that we have $\rho \in \Omega_{FG}(\sigma, C, L, c^{[0,t]}_q, t),\ \rho(0) \models c^{[0]}_q,\ \rho(0) \models u,\ \forall_{s\in(0,t)}\rho(s) \models c^{(0,t)}_q,\ \rho(t) \models c^{[t]}_q$, and $\exists_{s\in[0,t]}\rho(s) \models c_q$. So, $\langle q, \sigma, E\rangle \xmapsto{t,\rho} \langle q', \sigma', E\rangle \in \mathrm{Tr_d}(\langle q, \sigma, E\rangle)$, by induction, we then have $\langle q, \sigma, E\rangle \xmapsto{t,\rho} \langle q', \sigma', E\rangle$. Using Rule 14, we get $\langle u \curvearrowright q, \sigma, E\rangle \xmapsto{t,\rho} \langle q', \sigma', E\rangle$.

- $p \equiv q;\ r$ for some q and r. According to the definition of the function $\mathrm{Tr_d}$, we know that there exist $c^{[0]}_;, c^{(0,t)}_;, c^{[t]}_;, c^{[0,t]}_;, c_;$ such that $(c^{[0]}_;, c^{(0,t)}_;, c^{[t]}_;, c^{[0,t]}_;, c_;\ , p') \in \mathcal{S}_\mathrm{d}(\langle q;\ r, E\rangle),\ \rho \in \Omega_{FG}(\sigma, C, L, c^{[0,t]}_;, t),\ \rho(0) \models c^{[0]}_;, \forall_{s\in(0,t)}\rho(s) \models c^{(0,t)}_;,\ \rho(t) \models c^{[t]}_;$, and $\exists_{s\in[0,t]}\rho(s) \models c_;$. We know that $\mathcal{S}_\mathrm{d}(q;\ r, E) = \mathcal{S}_\mathrm{d}(q, E)$ and then $p' \equiv q';\ r$ for some q'. It is not hard to see that, by induction, we have $\langle q, \sigma, E\rangle \xmapsto{t,\rho} \langle q', \sigma', E\rangle$ for some q'. Using Rule 18, we get $\langle q;\ r, \sigma, E\rangle \xmapsto{t,\rho} \langle q';\ r, \sigma', E\rangle$.

- $p \equiv b \to q$ for some guard b and q. According to the definition of the function $\mathrm{Tr_d}$, there exist $c^{[0]}_\rightarrow, c^{(0,t)}_\rightarrow, c^{[t]}_\rightarrow, c^{[0,t]}_\rightarrow, c_\rightarrow$ such that $(c^{[0]}_\rightarrow, c^{(0,t)}_\rightarrow, c^{[t]}_\rightarrow, c^{[0,t]}_\rightarrow, c_\rightarrow, p') \in \mathcal{S}_\mathrm{d}(b \to q, E)$, $\rho \in \Omega_{FG}(\sigma, C, L, c^{[0,t]}_\rightarrow, t), \rho(0) \models c^{[0]}_\rightarrow, \forall_{s\in(0,t)}\rho(s) \models c^{(0,t)}_\rightarrow,\ \rho(t) \models c^{[t]}_\rightarrow$, and $\exists_{s\in[0,t]}\rho(s) \models c_\rightarrow$. From the definition of $\mathcal{S}_\mathrm{d}(\langle b \to q, E\rangle)$, we can distinguish two cases:

  - there exist $c^{[0]}_q, c^{(0,t)}_q, c^{[t]}_q, c^{[0,t]}_q, c_q, q'$ such that $(c^{[0]}_q, c^{(0,t)}_q, c^{[t]}_q, c^{[0,t]}_q, c_q, q') \in \mathcal{S}_\mathrm{d}(\langle q, E\rangle)$, $c^{[0]}_\rightarrow = b \wedge c^{[0]}_q$, $c^{(0,t)}_\rightarrow = b \wedge c^{(0,t)}_q, c^{[t]}_\rightarrow = b \wedge c^{[t]}_q, c^{[0,t]}_\rightarrow = b \wedge c^{[0,t]}_q$, $c_\rightarrow = c_q$ and $p' \equiv b \to q'$. It is not hard to see that we have $\rho \in \Omega_{FG}(\sigma, C, L, b, t), \rho \in \Omega_{FG}(\sigma, C, L, c^{[0,t]}_q, t), \rho(0) \models c^{[0]}_q, \forall_{s\in(0,t)}\rho(s) \models c^{(0,t)}_q,\ \rho(t) \models c^{[t]}_q$, and $\exists_{s\in[0,t]}\rho(s) \models c_q$. So, $\langle q, \sigma, E\rangle \xmapsto{t,\rho} \langle q',\ \sigma', E\rangle \in \mathrm{Tr_d}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma\rangle \xmapsto{t,\rho} \langle q', \sigma'\rangle$. From the definition of the function $\rho \in \Omega_{FG}(\sigma, C, L, b, t)$, we know that $\forall_{s\in[0,t]}\,\rho(s) \models b$. Using Rule 21, we get $\langle b \to q, \sigma\rangle \xmapsto{t,\rho} \langle b \to q', \sigma'\rangle$.
  - there exist $c^{[0]}_q, c^{(0,t)}_q, c^{[t]}_q, c^{[0,t]}_q, c_q, q'$ such that $(c^{[0]}_q, c^{(0,t)}_q, c^{[t]}_q, c^{[0,t]}_q, c_q, q') \in \mathcal{S}_\mathrm{d}(\langle q, E\rangle)$, $c^{[0]}_\rightarrow = b \implies c^{[0]}_q$, $c^{(0,t)}_\rightarrow = \neg b$, $c^{[t]}_\rightarrow = b \implies \mathcal{C}_\mathrm{c}(\langle q, E\rangle)$, $c^{[0,t]}_\rightarrow = \mathrm{true}$, $c_\rightarrow = c_q = \neg b$

and $p' \equiv b \to q$. It is not hard to see that we have $\rho \in \Omega_{FG}(\sigma, C, L, \mathrm{true}, t), \rho(0) \models b \implies c_q^{[0]}, \forall_{s\in(0,t)}\rho(s) \models \neg b$, $\exists_{s\in[0,t]}\rho(s) \models \neg b$ and $\rho(t) \models b \implies \mathcal{C}_{\mathrm{c}}(\langle q, E\rangle)$. From $c_{\rightarrow}^{[0]} = b \implies c_q^{[0]}$, we get $\rho(0) \models b \Rightarrow \langle q, \sigma, E\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q'', \sigma', E\rangle$ for some q'' using Conjecture D.1.1. From $\rho(t) \models b \implies \mathcal{C}_{\mathrm{c}}(\langle q, E\rangle)$, we get $\rho(t) \models b \Rightarrow \langle q, \rho_\sigma(t), E\rangle \overset{\rho(t)}{\rightsquigarrow}$ by Theorem 6.1.1. Then we can apply Rule 22, we obtain $\langle b \to q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle b \to q, \rho_\sigma(t), E\rangle$, and $\sigma' = \rho_\sigma(t)$.

- $p \equiv q \mathbin{[\![} r$ for some q and r. According to the definition of the function Tr_{d}, we know that there exist $c_{[\![}^{[0]}, c_{[\![}^{(0,t)}, c_{[\![}^{[t]}, c_{[\![}^{[0,t]}, c_{[\![}, p'$ such that $(c_{[\![}^{[0]}, c_{[\![}^{(0,t)}, c_{[\![}^{[t]}, c_{[\![}^{[0,t]}, c_{[\![}, p') \in \mathcal{S}_{\mathrm{d}}(\langle q \mathbin{[\![} r, E\rangle)$, $\rho \in \Omega_{FG}(\sigma, C, L, c_{[\![}^{[0,t]}, t)$, $\rho(0) \models c_{[\![}^{[0]}$, $\forall_{s\in(0,t)}\rho(s) \models c_{[\![}^{(0,t)}$, $\exists_{s\in[0,t]}\rho(s) \models c_{[\![}$ and $\rho(t) \models c_{[\![}^{[t]}$. From the definition of the function $\mathcal{S}_{\mathrm{d}}(\langle q \mathbin{[\![} r, E\rangle)$, we also know that there exist $c_q^{[0]}, c_q^{(0,t)}, c_q^{[t]}, c_q^{[0,t]}, c_q, q', c_r^{[0]}, c_r^{(0,t)}, c_r^{[t]}, c_r^{[0,t]}, c_r, r'$ such that $(c_q^{[0]}, c_q^{(0,t)}, c_q^{[t]}, c_q^{[0,t]}, c_q, q') \in \mathcal{S}_{\mathrm{d}}(\langle q, E\rangle)$, $(c_r^{[0]}, c_r^{(0,t)}, c_r^{[t]}, c_r^{[0,t]}, c_r, r') \in \mathcal{S}_{\mathrm{d}}(\langle r, E\rangle)$, $c_{q[\![r}^{[0]} = c_q^{[0]} \wedge c_r^{[0]}$, $c_{q[\![r}^{(0,t)} = c_q^{(0,t)} \wedge c_r^{(0,t)}$, $c_{q[\![r}^{[t]} = c_q^{[t]} \wedge c_r^{[t]}$, $c_{[\![}^{[0,t]} = c_q^{[0,t]} \wedge c_r^{[0,t]}$, $c_{[\![} = c_q \wedge c_r$ and $p' \equiv q' \mathbin{[\![} r'$. It is not hard to see that we can have $\rho \in \Omega_{FG}(\sigma, C, L, c_q^{[0,t]}, t)$, $\rho(0) \models c_q^{[0]}$, $\forall_{s\in(0,t)}\rho(s) \models c_q^{(0,t)}$, $\rho(t) \models c_q^{[t]}$, $\exists_{s\in[0,t]}\rho(s) \models c_q$, $\rho \in \Omega_{FG}(\sigma, C, L, c_r^{[0,t]}, t)$, $\rho(0) \models c_r^{[0]}$, $\forall_{s\in(0,t)}\rho(s) \models c_r^{(0,t)}$, $\rho(t) \models c_r^{[t]}$, and $\exists_{s\in[0,t]}\rho(s) \models c_r$. So, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle \in \mathrm{Tr}_{\mathrm{d}}(q, \sigma, E)$ and $\langle r, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle r', \sigma', E\rangle \in \mathrm{Tr}_{\mathrm{d}}(r, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ and $\langle r, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle r', \sigma', E\rangle$. Using Rule 27, we get $\langle q \mathbin{[\![} r, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q' \mathbin{[\![} r', \sigma', E\rangle$.

- $p \equiv q \parallel r$ for some q and r. The proof is similar to the proof of the case that $p \equiv q \mathbin{[\![} r$, because both definitions of $\mathcal{S}_{\mathrm{d}}$ and deduction rules for time transition for both operators are similar.

- $p \equiv \partial_A(q)$ for some A and q. The proof is similar to the proof of the case that $p \equiv q;\, r$.

- REMOVED $p \equiv \upsilon_H(q)$ for some H and q. According to the definition of the function Tr_{d}, we know that there exist $c_{\upsilon_H(q)}^{[0]}$, $c_{\upsilon_H(q)}^{(0,t)}$, $c_{\upsilon_H(q)}^{[t]}, c_{\upsilon_H(q)}^{[0,t]}, p'$ such that $(c_{\upsilon_H(q)}^{[0]}, c_{\upsilon_H(q)}^{(0,t)}, c_{\upsilon_H(q)}^{[t]}, c_{\upsilon_H(q)}^{[0,t]}, p') \in \mathcal{S}_{\mathrm{d}}(\langle \upsilon_H(q), E\rangle)$, $\rho \in \Omega_{FG}(\sigma, C, L, c_{\upsilon_H(q)}^{[0,t]}, t)$, $\rho(0) \models c_{\upsilon_H(q)}^{[0]}$, $\forall_{s\in(0,t)}\rho(s) \models c_{\upsilon_H(q)}^{(0,t)}$, and $\rho(t) \models c_{\upsilon_H(q)}^{[t]}$. From the definition of $\mathcal{S}_{\mathrm{d}}(\langle \upsilon_H(q), E\rangle)$, we know that there exist $c_q^{[0]}, c_q^{(0,t)}, c_q^{[t]}, c_q^{[0,t]}, q'$ such that $(c_q^{[0]}, c_q^{(0,t)}, c_q^{[t]}, c_q^{[0,t]}, q') \in \mathcal{S}_{\mathrm{d}}(q, E)$, $c_{\upsilon_H(q)}^{[0]} = c_q^{[0]} \wedge \neg c_q^{\mathrm{u}}$ for some $c_q^{\mathrm{u}} = \vee_{c:c\in\{c_q|(c_q,W_q,r_q,\mathrm{ca}(h,cs),C_q^b,C_q^a,q')\in\mathcal{S}_{\mathrm{a}}(\langle q,E\rangle),cs\in\Lambda^*,h\in H\}^{\mathcal{C}}}$, $c_{\upsilon_H(q)}^{(0,t)} = c_q^{(0,t)} \wedge \neg c_q^{\mathrm{u}}$, $c_{\upsilon_H(q)}^{[t]} = c_q^{[t]}$, and $c_{\upsilon_H(q)}^{[0,t]} = c_q^{[0,t]}$. It is not hard to see that $\rho \in \Omega_{FG}(\sigma, C, L, c_q^{[0,t]}, t)$, $\rho(0) \models c_q^{[0]}$, $\rho(0) \models \neg c_q^{\mathrm{u}}$, $\forall_{s\in(0,t)}\rho(s) \models c_q^{(0,t)}$, $\forall_{s\in(0,t)}\rho(s) \models \neg c_q^{\mathrm{u}}$ and $\rho(t) \models c_q^{[t]}$. So, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle \in \mathrm{Tr}_{\mathrm{d}}(q, \sigma, E)$, by induction, we then have $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$. $\rho(0) \models \neg c_q^{\mathrm{u}}$ and $\forall_{s\in(0,t)}\rho(s) \models \neg c_q^{\mathrm{u}}$ together mean that time

transitions are allowed only if at each intermediate state while delaying no communication actions via channels from H are possible. This is equivalent to say that $\forall_{s\in[0,t)}(\langle q,\sigma\rangle \overset{s,\rho\restriction[0,s]}{\longmapsto} \langle q_s,\sigma_s\rangle, \langle q_s,\sigma_s\rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle q',\sigma'\rangle, \forall_{h\in\mathcal{H}} \langle q_s,\sigma_s,E\rangle \overset{\mathrm{ca}(h,*)}{\not\mapsto})$. Using Rule 37, we get $\langle \upsilon_H(q),\sigma,E\rangle \overset{t,\rho}{\longmapsto} \langle \upsilon_H(q'),\sigma',E\rangle$ and $p' \equiv \upsilon_H(q')$.

- $p \equiv X$ for some X. The proof is similar to the proof of the case that $p \equiv q;\ r$.
- $p \equiv \iota_{J^+}(q)$ for some q and set J^+. The proof is trivial and is similar to the proof of the case that $p \equiv q;\ r$.

APPENDIX

E

Proofs of the elimination of Chi

E.1 Proof of Proposition 8.3.1

Let $p \in N$, then $\exists_{q:q\in N}\ [p] \leftrightarrows q$.

PROOF. This proof is by induction on structure of p.

- $p \equiv u$. By Lemma B.1.2, we know that $[u] \leftrightarrows \text{true}$, which is a term in N.
- $p \equiv b \rightarrow t^*$, where b is a guard and $t^* \in P_{\text{bs}}$. It is not hard to see that $[b \rightarrow t^*]$ is a term in N.
- $p \equiv [b \rightarrow t^*]$, where b is a guard and $t^* \in P_{\text{bs}}$. By Lemma B.1.1, we know that $[[b \rightarrow t^*]] \leftrightarrows [b \rightarrow t^*]$, which is a term in N.
- $p \equiv b \rightarrow t^*;\ t_*$, where b is a guard, $t^* \in P_{\text{bs}}$ and $t_* \in N$. It is not hard to see that $[b \rightarrow t^*;\ t_*]$ is a term in N.
- $p \equiv [b \rightarrow t^*;\ t_*]$, where b is a guard, $t^* \in P_{\text{bs}}$ and $t_* \in N$. By Lemma B.1.1, we know that $[[b \rightarrow t^*;\ t_*]] \leftrightarrows [b \rightarrow t^*;\ t_*]$, which is a term in N.
- $p \equiv t_1 \between t_2$, where $t_1, t_2 \in N$. By Lemma B.3.5, we know that $[t_1 \between t_2] \leftrightarrows [t_1] \between [t_2]$. By induction, we know that there exist $t_1', t_2' \in N$: $[t_1] \leftrightarrows t_1' \wedge [t_2] \leftrightarrows t_2'$. Then $t_1' \between t_2'$ is also a term in N (because $\between$ of two terms in N is by definition also a term in N).

E.2 Proof of Proposition 8.3.2

Let $p_1, p_2 \in N$, then $\exists_{q:q\in N}\ p_1;\ p_2 \leftrightarrows q$.

PROOF. This proof is by induction on structure of p_1.

- $p_1 \equiv u$. By Lemma B.5.5, we know that $u;\ p_2 \leftrightarrows u$, which is a term in N.
- $p_1 \equiv b \rightarrow t^*$, where b is a guard and $t^* \in P_{\text{bs}}$. It is not hard to see that $b \rightarrow t^*;\ p_2$ is a term in N.

- $p_1 \equiv [b \to t^*]$, where b is a guard and $t^* \in P_{\mathrm{bs}}$. By Lemma B.5.6, we can have $[b \to t^*];\ p_2 \leftrightarrows [b \to t^*;\ p_2]$. It is not hard to see that $[b \to t^*;\ p_2]$ is a term in N.

- $p_1 \equiv b \to t^*;\ t_*$, where b is a guard, $t^* \in P_{\mathrm{bs}}$ and $t_* \in N$. By Lemma B.5.2, we can have $(b \to t^*;\ t_*);\ p_2 \leftrightarrows b \to t^*;\ (t_*;\ p_2)$. By induction, there exists a $r \in N$ such that $t_*;\ p_2 \leftrightarrows r$. It is not hard to see that $b \to t^*;\ r$ is a term in N.

- $p_1 \equiv [b \to t^*;\ t_*]$, where b is a guard, $t^* \in P_{\mathrm{bs}}$ and $t_* \in N$. By Lemmas B.5.6 and B.5.2, we can have $[b \to t^*;\ t_*];\ p_2 \leftrightarrows [b \to t^*;\ (t_*;\ p_2)]$. By induction, there exists a $r \in N$ such that $t_*;\ p_2 \leftrightarrows r$. It is not hard to see that $[b \to t^*;\ r]$ is a term in N.

- $p_1 \equiv t_1 \between t_2$, where $t_1, t_2 \in N$. By Lemma B.5.3, we can have $(t_1 \between t_2);\ p_2 \leftrightarrows t_1;\ p_2 \between t_2;\ p_2$. By induction, we know that there exist $t_1', t_2' \in N$: $t_1;\ p_2 \leftrightarrows t_1' \wedge t_2;\ p_2 \leftrightarrows t_2'$. Then $t_1' \between t_2'$ is also a term in N (because $\between$ of two terms in N is by definition also a term in N).

E.3 Proof of Proposition 8.3.5

Let A be a set of actions and $p \in N$, then $\exists_{q:q\in N}\ \partial_A(p) \leftrightarrows q$.

PROOF. This proof is by induction on structure of p.

- $p \equiv u$. By Lemma B.7.6, we know that $\partial_A(u) \leftrightarrows u$, which is a term in N.

- $p \equiv b \to t^*$, where b is a guard and $t^* \in P_{\mathrm{bs}}$. By Lemma B.7.8, we get $\partial_A(b \to t^*) \leftrightarrows b \to \partial_A(t^*)$. From Lemma 8.4.2, we know that there exists $t_* \in P_{\mathrm{bs}}$: $\partial_A(t^*) \leftrightarrows t_*$. Then, it is not hard to see that $b \to t_*$ is a term in N.

- $p \equiv [b \to t^*]$, where b is a guard and $t^* \in P_{\mathrm{bs}}$. By Lemma B.7.7, we know that $\partial_A([b \to t^*]) \leftrightarrows [\partial_A(b \to t^*)]$. We can have $[\partial_A(b \to t^*)] \leftrightarrows [b \to \partial_A(t^*)\ ]$ using Lemma B.7.8. From Lemma 8.4.2, we know that there exists $t_* \in P_{\mathrm{bs}}$: $\partial_A(t^*) \leftrightarrows t_*$. Then, it is not hard to see that $\partial_A([b \to t^*]) \leftrightarrows [b \to t_*]$, which is a term in N.

- $p \equiv b \to t^*;\ t_*$, where b is a guard, $t^* \in P_{\mathrm{bs}}$ and $t_* \in N$. By Lemma B.7.5, we know that $\partial_A(b \to t^*;\ t_*) \leftrightarrows \partial_A(b \to t^*);\ \partial_A(t_*)$. Using Lemma B.7.8, we have $\partial_A(b \to t^*)\ ;\ \partial_A(t_*) \leftrightarrows (b \to \partial_A(t^*));\ \partial_A(t_*)$. From Lemma 8.4.2, we know that there exists $t_1 \in P_{\mathrm{bs}}$: $\partial_A(t^*) \leftrightarrows t_1$, and $b \to t_1$ is a term in N. By induction we know that there exists $t_2 \in N : \partial_A(t_*) \leftrightarrows t_2$. Putting them together, we obtain $\partial_A(b \to t^*;\ t_*) \leftrightarrows b \to t_1;\ t_2$, and we know that $b \to t_1;\ t_2$ is a term in N.

- $p \equiv [b \to t^*;\ t_*]$, where b is a guard, $t^* \in P_{\mathrm{bs}}$ and $t_* \in N$. By Lemma B.7.7, we know that $\partial_A([b \to t^*;\ t_*]) \leftrightarrows [\partial_A(b \to t^*;\ t_*)]$. From the proof of the case that $p \equiv b \to t^*;\ t_*$, we know that there exists a $t_1 \in N : \partial_A(b \to t^*;\ t_*) \leftrightarrows t_1$. Due to Proposition 8.3.1, we know that there exists $t_1' \in N$: $[t_1] \leftrightarrows t_1'$. Putting them together, we can have $\partial_A([b \to t^*;\ t_*]) \leftrightarrows t_1'$, which is a term in N.

- $p \equiv t_1 \,[\!]\, t_2$, where $t_1, t_2 \in N$. By Lemma B.7.4, we know that $\partial_A(t_1 \,[\!]\, t_2) \leftrightarrows \partial_A(t_1) \,[\!]\, \partial_A(t_2)$. By induction, we know that there exist $t_1', t_2' \in N$: $\partial_A(t_1) \leftrightarrows t_1' \wedge \partial_A(t_2) \leftrightarrows t_2'$. Observe that $t_1' \,[\!]\, t_2'$ is a term in N.

E.4 Proof of Lemma 8.4.1

For arbitrary guards b_1, b_2, channel h, expression(s) $\mathbf{e}_n$, variable(s) $\mathbf{x}_n$, and p_1, p_2 such that $(p_1 \equiv h\,!!\,\mathbf{e}_n \wedge p_2 \equiv h\,??\,\mathbf{x}_n) \vee (p_1 \equiv h\,??\,\mathbf{x}_n \wedge p_2 \equiv h\,!!\,\mathbf{e}_n)$, we have

$b_1 \to p_1 \parallel b_2 \to p_2 \leftrightarrows (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1) \,[\!]\, [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)]$.

PROOF. Let $R = \{(b_1 \to p_1 \parallel b_2 \to p_2, (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1) \,[\!]\, [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)]) \mid (p_1 \equiv h\,!!\,\mathbf{e}_n \wedge p_2 \equiv h\,??\,\mathbf{x}_n) \vee (p_1 \equiv h\,??\,\mathbf{x}_n \wedge p_2 \equiv h\,!!\,\mathbf{e}_n)$, guards b_1, b_2, channel h, expression(s) $\mathbf{e}_n$, variable(s) $\mathbf{x}_n\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since the proofs for the case that $p_1 \equiv h\,!!\,\mathbf{e}_n \wedge p_2 \equiv h\,??\,\mathbf{x}_n$ and $p_1 \equiv h\,??\,\mathbf{x}_n \wedge p_2 \equiv h\,!!\,\mathbf{e}_n$ are similar, we only give the proofs for the case that $p_1 \equiv h\,!!\,\mathbf{e}_n \wedge p_2 \equiv h\,??\,\mathbf{x}_n$. The proofs of the left implication of conditions 1 and 6 are similar to the proofs of the right implication of conditions 1 and 6. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle b_1 \to p_1 \parallel b_2 \to p_2, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some E, σ, ξ, a, ξ', σ', which means that Rule 28.1.l has been applied necessarily. Then, we have $(C, J \cup W, L, H, R) \Vdash \langle b_1 \to p_1, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\ \checkmark,\ \sigma'\ \rangle$, $(C, J, L, H, R) \Vdash \langle b_2 \to p_2, \sigma\ \rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle \checkmark, \sigma'\rangle$ for some C, J, W, L, H, R, cs such that $E = (C, J, W, L, H, R)$ and $a = \mathrm{ca}(h, cs)$. From Rule 20.1 (see also Rules 5 and 6), we further obtain $\xi \models b_1$, $\xi \models b_2$, $(C, J, L, H, R) \Vdash \langle h\,!!\,\mathbf{e}_n, \sigma\rangle \xrightarrow{\xi,\ \mathrm{isa}(h,[\xi(\mathbf{e}_n)]),\ \xi'} \langle \checkmark, \xi'_\sigma\rangle$ and $(C, J, L, H, R) \Vdash \langle\ h\,??\mathbf{x}_n,\ \sigma\ \rangle \xrightarrow{\xi,\mathrm{ira}(h,[\mathbf{c}_n],\{\mathbf{x}_n\}),\xi'} \langle\ \checkmark,\ \xi'_\sigma\rangle$ for some $\mathbf{c}_n$, $cs = [\xi(\mathbf{e}_n)] = [\mathbf{c}_n]$, $W = \{\mathbf{x}_n\}$ and $\sigma' = \xi'_\sigma$. Due to Rules 5 and 6, we have $\xi = \sigma \cup \xi^{\dot{C}L}, \xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L), \xi'(\mathbf{x}_n) = \mathbf{c}_n$. It is not hard to see that we have also $\xi'(\mathbf{x}_n) = \xi(\mathbf{e}_n)$. Using Rules 54 and 20.1, we obtain $(C, J, L, H, R) \Vdash \langle b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,[\xi(\mathbf{e}_n)]),\xi'} \langle \checkmark, \xi'_\sigma\rangle$. According Rule 10.1, we have $(C, J, L, H, R) \Vdash \langle [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,[\xi(\mathbf{e}_n)]),\xi'} \langle \checkmark, \xi'_\sigma\rangle$. Applying Rules 7,8,19 and 23, it is not hard to see that we have $E \Vdash \langle b_1 \ \to p_1\ ;\ \ b_2 \to p_2, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle b_2 \to p_2;\ b_1 \to p_1, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we get $E \Vdash \langle b_1 \to p_1;\ b_2 \to p_2 \,[\!]\, b_2 \to p_2;\ b_1 \to p_1, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Then we conclude that $E \Vdash \langle (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1) \,[\!]\, [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ using Rule 25.1.r.

Condition 2: We assume $E \Vdash \langle b_1 \to p_1 \parallel b_2 \to p_2, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some E, σ, ξ, a, ξ', k_1 σ', which means that Rule 29.1.l or Rule 29.1.r has been applied necessarily. We distinguish two cases:

- Rule 29.1.l has been applied. Then, we have $E \Vdash \langle b_1 \to p_1, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $E \Vdash \langle b_2 \to p_2, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle b_2 \to p_2, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv b_2 \to p_2$. Using Rule 16, we have $E \Vdash \langle b_1 \to p_1;\ b_2 \to p_2, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle b_2 \to p_2, \sigma'\rangle$. According to Rule 19 and Lemma 3.5.6,

we get $E \Vdash \langle b_2 \to p_2;\ b_1 \to p_1, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We also know that process term $[p]$ for any $p \in P$ is consistent with any extended valuation with respect to σ in any environment. Due to Rule 27, we get $E \Vdash \langle (b_2 \to p_2;\ b_1 \to p_1) \,[\!]\, [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 25.1.l, we $E \Vdash \langle (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1) \,[\!]\, [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle b_2 \to p_2, \sigma'\rangle$. Take $k_2 \equiv b_2 \to p_2$ and observe that $(k_1, k_2) \in R$.

- Rule 29.1.r has been applied. The proof of this case is similar to the proof of the previous case.

Condition 4: We assume $E \Vdash \langle b_1 \to p_1 \parallel b_2 \to p_2, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some E, σ, t, ρ, k_1, σ', which means Rule 30 has been applied necessarily. Then we get $E \Vdash \langle b_1 \to p_1, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$ and $E \Vdash \langle b_2 \to p_2, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_2', \sigma'\rangle$ for some k_1', k_2' such that $k_1 \equiv k_1' \parallel k_2'$. Since p_1, p_2 are undelayable process terms, Rule 21 cannot have been applied. From Rule 22, we know that $\rho \in \Omega_{\sigma E t}, \forall_{s\in(0,t)}\, \rho(s) \models \neg b_1, \exists_{s\in[0,t]}\, \rho(s) \models \neg b_1, \forall_{s\in(0,t)} \rho(s) \models \neg b_2, \exists_{s\in[0,t]}\, \rho(s) \models \neg b_2$, $k_1' \equiv b_1 \to p_1$, $k_2' \equiv b_1 \to p_2$, $\sigma' = \rho_\sigma(t)$, and some unimportant information for this proof is omitted. We also have $E \Vdash \langle [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma'\rangle$ (see also Rule 11). Applying Rules 18 and 26, we get $E \Vdash \langle (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1), \sigma'\rangle$. Again, by Rule 26, we obtain $E \Vdash \langle (b_1 \to p_1;\ b_2 \to p_2) [\!] (b_2 \to p_2;\ b_1 \to p_1) [\!] [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle (b_1 \to p_1;\ b_2 \to p_2) \,[\!]\, (b_2 \to p_2;\ b_1 \to p_1) \,[\!]\, [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma'\rangle$. Take $k_2 \equiv (b_1 \to p_1;\ b_2 \to p_2) [\!] (b_2 \to p_2;\ b_1 \to p_1) [\!] [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)]$ and observe that $(k_1, k_2) \in R$.

Condition 6: First, we assume $E \Vdash \langle b_1 \to p_1 \parallel b_2 \to p_2, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 31 has been applied necessarily. Then, we get $E \Vdash \langle b_1 \to p_1, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle b_2 \to p_2, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rules 19 and 27, we have $E \Vdash \langle\ (b_1 \to p_1;\ b_2 \to p_2) [\!] (b_2 \to p_2;\ b_1 \to p_1), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We also know that process term $[p]$ for any $p \in P$ is consistent with any extended valuation with respect to σ in any environment. We conclude $E \Vdash \langle (b_1 \to p_1;\ b_2 \to p_2) [\!] (b_2 \to p_2;\ b_1 \to p_1) [\!] [b_1 \wedge b_2 \to \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)], \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 27.

E.5 Proof of Lemma 8.4.2

Let A be a set of actions and $p_* \in P_{\mathrm{bs}}$, then $\exists_{q:q\in P_{\mathrm{bs}}}\, \partial_A(p_*) \leftrightarrows q$.

PROOF. This proof is by induction on structure of p_*.

- $p_* \equiv W : r \gg l_{\mathrm{a}}$. We distinguish two cases:
  - if $l_{\mathrm{a}} \notin A$, then it is not hard to see that we can have $\partial_A(W : r \gg l_{\mathrm{a}}) \leftrightarrows W : r \gg l_{\mathrm{a}}$ (proof is omitted for this trivial property), which is a term in P_{bs}.

  - if $l_{\mathrm{a}} \in A$, then it is not hard to see that we have $\partial_A(W : r \gg l_{\mathrm{a}}) \leftrightarrow \delta$ (proof is omitted for this trivial property), which is a term in P_{bs}.

- Since the proofs for the cases $p_* \equiv h \,!!\, \mathbf{e}_n$, $p_* \equiv h \,??\, \mathbf{x}_n$ and $p_* \equiv \mathrm{ca}(h, \mathbf{e}_n, \mathbf{x}_n)$ are similar, we only give the proof for the case $p_* \equiv h \,!!\, \mathbf{e}_n$. We distinguish two cases:

  - if $\mathrm{isa}(h, [\xi(\mathbf{e}_n)]) \notin A$, then it is not hard to see that we can have $\partial_A(h \,!!\, \mathbf{e}_n) \leftrightarrow h \,!!\, \mathbf{e}_n$ (proof is omitted for this trivial property), which is a term in P_{bs}.
  - if $\mathrm{isa}(h, [\xi(\mathbf{e}_n)]) \in A$, then it is not hard to see that we have $\partial_A(h \,!!\, \mathbf{e}_n) \leftrightarrow \delta$ (proof is omitted for this trivial property), which is a term in P_{bs}.

- $p_* \equiv \delta$. Trivial.

E.6 Proof of Lemma 8.4.3

For some finite index sets $I, J, K, L, M, I^*, J^*, K^*, L^*, M^*$, arbitrary predicates u_i, u_{i^*}, arbitrary guards b_j, b_k , b_l, b_m, b_{j^*}, $b_{k^*}, b_{l^*}, b_{m^*}$; p_j, p_k, p_l, p_m, p_{j^*}, $p_{k^*}, p_{l^*}, p_{m^*} \in P_{\mathrm{bs}}$; n_l, n_{l^*}, n_m, $n_{m^*} \in N$, $s_1 \equiv (\llbracket_{i \in I}\ u_i) \rrbracket (\llbracket_{j \in J}\ b_j \to p_j) \rrbracket (\llbracket_{k \in K}\ [b_k \to p_k]) \rrbracket (\llbracket_{l \in L}\ b_l \to p_l;\ n_l) \rrbracket (\llbracket_{m \in M}\ [b_m \to p_m;\ n_m])$, $s_2 \equiv (\llbracket_{i^* \in I^*}\ u_{i^*}) \rrbracket (\llbracket_{j^* \in J^*}\ b_{j^*} \to p_{j^*}) \rrbracket (\llbracket_{k^* \in K^*}\ [b_{k^*} \to p_{k^*}]) \rrbracket (\llbracket_{l^* \in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) \rrbracket (\llbracket_{m^* \in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}])$, we have

$s_1 \parallel s_2 \leftrightarrow s_3$, where

$s_3 \equiv (\llbracket_{i \in I}\ u_i) \rrbracket (\llbracket_{i^* \in I^*}\ u_{i^*})$
$\rrbracket (\llbracket_{j \in J}\ b_j \to p_j;\ s_2)$
$\rrbracket (\llbracket_{k \in K}\ [b_k \to p_k;\ s_2])$
$\rrbracket (\llbracket_{l \in L}\ b_l \to p_l;\ (n_l \parallel s_2))$
$\rrbracket (\llbracket_{m \in M}\ [b_m \to p_m];\ (n_m \parallel s_2))$
$\rrbracket (\llbracket_{j^* \in J^*}\ b_{j^*} \to p_{j^*};\ s_1)$
$\rrbracket (\llbracket_{k^* \in K^*}\ [b_{k^*} \to p_{k^*};\ s_1])$
$\rrbracket (\llbracket_{l^* \in L^*}\ b_{l^*} \to p_{l^*};\ (n_{l^*} \parallel s_1))$
$\rrbracket (\llbracket_{m^* \in M^*}\ [b_{m^*} \to p_{m^*}];\ (n_{m^*} \parallel s_1))$
$\rrbracket (\llbracket_{(j,j^*) \in \Gamma_{J,J^*}}\ [b_j \wedge b_{j^*} \to\ \mathrm{ca}(h_{jj^*}, \mathbf{e}_{n_{jj^*}}, \mathbf{x}_{n_{jj^*}})])$
$\rrbracket (\llbracket_{(j,k^*) \in \Gamma_{J,K^*}}\ [b_j \wedge b_{k^*} \to \mathrm{ca}(h_{jk^*}, \mathbf{e}_{n_{jk^*}}, \mathbf{x}_{n_{jk^*}})])$
$\rrbracket (\llbracket_{(j,l^*) \in \Gamma_{J,L^*}}\ [b_j \wedge b_{l^*} \to \mathrm{ca}(h_{jl^*}, \mathbf{e}_{n_{jl^*}}, \mathbf{x}_{n_{jl^*}});\ n_{l^*}])$
$\rrbracket (\llbracket_{(j,m^*) \in \Gamma_{J,M^*}}\ [b_j \wedge b_{m^*} \to \mathrm{ca}(h_{jm^*}, \mathbf{e}_{n_{jm^*}}, \mathbf{x}_{n_{jm^*}});\ n_{m^*}])$
$\rrbracket (\llbracket_{(k,j^*) \in \Gamma_{K,J^*}}\ [b_k \wedge b_{j^*} \to \mathrm{ca}(h_{kj^*}, \mathbf{e}_{n_{kj^*}}, \mathbf{x}_{n_{kj^*}})])$
$\rrbracket (\llbracket_{(k,k^*) \in \Gamma_{K,K^*}}\ [b_k \wedge b_{k^*} \to \mathrm{ca}(h_{kk^*}, \mathbf{e}_{n_{kk^*}}, \mathbf{x}_{n_{kk^*}})])$
$\rrbracket (\llbracket_{(k,l^*) \in \Gamma_{K,L^*}}\ [b_k \wedge b_{l^*} \to \mathrm{ca}(h_{kl^*}, \mathbf{e}_{n_{kl^*}}, \mathbf{x}_{n_{kl^*}});\ n_{l^*}])$
$\rrbracket (\llbracket_{(k,m^*) \in \Gamma_{K,M^*}}\ [b_k \wedge b_{m^*} \to \mathrm{ca}(h_{km^*}, \mathbf{e}_{n_{km^*}}, \mathbf{x}_{n_{km^*}});\ n_{m^*}])$
$\rrbracket (\llbracket_{(l,j^*) \in \Gamma_{L,J^*}}\ [b_l \wedge b_{j^*} \to \mathrm{ca}(h_{lj^*}, \mathbf{e}_{n_{lj^*}}, \mathbf{x}_{n_{lj^*}});\ n_l])$
$\rrbracket (\llbracket_{(l,k^*) \in \Gamma_{L,K^*}}\ [b_l \wedge b_{k^*} \to \mathrm{ca}(h_{lk^*}, \mathbf{e}_{n_{lk^*}}, \mathbf{x}_{n_{lk^*}});\ n_l])$
$\rrbracket (\llbracket_{(l,l^*) \in \Gamma_{L,L^*}}\ [b_l \wedge b_{l^*} \to \mathrm{ca}(h_{ll^*}, \mathbf{e}_{n_{ll^*}}, \mathbf{x}_{n_{ll^*}});\ (n_l \parallel n_{l^*})])$

$[] ([]_{(l,m^*)\in\Gamma_{L,M^*}} [b_l \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{lm^*}, \mathbf{e}_{n_{lm^*}}, \mathbf{x}_{n_{lm^*}});\ (n_l \parallel n_{m^*})])$
$[] ([]_{(m,j^*)\in\Gamma_{M,J^*}} [b_m \wedge b_{j^*} \rightarrow \mathrm{ca}(h_{mj^*}, \mathbf{e}_{n_{mj^*}}, \mathbf{x}_{n_{mj^*}});\ n_m])$
$[] ([]_{(m,k^*)\in\Gamma_{M,K^*}} [b_m \wedge b_{k^*} \rightarrow \mathrm{ca}(h_{mk^*}, \mathbf{e}_{n_{mk^*}}, \mathbf{x}_{n_{mk^*}});\ n_m])$
$[] ([]_{(m,l^*)\in\Gamma_{M,L^*}} [b_m \wedge b_{l^*} \rightarrow \mathrm{ca}(h_{ml^*}, \mathbf{e}_{n_{ml^*}}, \mathbf{x}_{n_{ml^*}});\ (n_m \parallel n_{l^*})])$
$[] ([]_{(m,m^*)\in\Gamma_{M,M^*}} [b_m \wedge b_{m^*} \rightarrow \mathrm{ca}(h_{mm^*}, \mathbf{e}_{n_{mm^*}}, \mathbf{x}_{n_{mm^*}});\ (n_m \parallel n_{m^*})])$
$[] ([]_{(j^*,j)\in\Gamma_{J^*,J}} [b_{j^*} \wedge b_j \rightarrow \mathrm{ca}(h_{j^*j}, \mathbf{e}_{n_{j^*j}}, \mathbf{x}_{n_{j^*j}})])$
$[] ([]_{(j^*,k)\Gamma_{J^*,K}} [b_{j^*} \wedge b_k \rightarrow \mathrm{ca}(h_{j^*k}, \mathbf{e}_{n_{j^*k}}, \mathbf{x}_{n_{j^*k}})])$
$[] ([]_{(j^*,l)\in\Gamma_{J^*,L}} [b_{j^*} \wedge b_l \rightarrow\ \mathrm{ca}(h_{j^*l}, \mathbf{e}_{n_{j^*l}}, \mathbf{x}_{n_{j^*l}});\ n_l])$
$[] ([]_{(j^*,m)\in\Gamma_{J^*,M}} [b_{j^*} \wedge b_m \rightarrow \mathrm{ca}(h_{j^*m}, \mathbf{e}_{n_{j^*m}}, \mathbf{x}_{n_{j^*m}});\ n_m])$
$[] ([]_{(k^*,j)\in\Gamma_{K^*,J}} [b_{k^*} \wedge b_j \rightarrow \mathrm{ca}(h_{k^*j}, \mathbf{e}_{n_{k^*j}}, \mathbf{x}_{n_{k^*j}})])$
$[] ([]_{(k^*,k)\in\Gamma_{K^*,K}} [b_{k^*} \wedge b_k \rightarrow \mathrm{ca}(h_{k^*k}, \mathbf{e}_{n_{k^*k}}, \mathbf{x}_{n_{k^*k}})])$
$[] ([]_{(k^*,l)\in\Gamma_{K^*,L}} [b_{k^*} \wedge\ b_l \rightarrow \mathrm{ca}(h_{k^*l},\ \mathbf{e}_{n_{k^*l}}, \mathbf{x}_{n_{k^*l}});\ n_l])$
$[] ([]_{(k^*,m)\in\Gamma_{K^*,M}} [b_{k^*} \wedge b_m \rightarrow \mathrm{ca}(h_{k^*m}, \mathbf{e}_{n_{k^*m}},\ \mathbf{x}_{n_{k^*m}});\ n_m])$
$[] ([]_{(l^*,j)\in\Gamma_{L^*,J}} [b_{l^*} \wedge b_j \rightarrow \mathrm{ca}(h_{l^*j}, \mathbf{e}_{n_{l^*j}}, \mathbf{x}_{n_{l^*j}});\ n_{l^*}])$
$[] ([]_{(l^*,k)\in\Gamma_{L^*,K}} [b_{l^*} \wedge b_k \rightarrow \mathrm{ca}(h_{l^*k}, \mathbf{e}_{n_{l^*k}}, \mathbf{x}_{n_{l^*k}});\ n_{l^*}])$
$[] ([]_{(l^*,l)\in\Gamma_{L^*,L}} [b_{l^*} \wedge b_l \rightarrow \mathrm{ca}(h_{l^*l}, \mathbf{e}_{n_{l^*l}}, \mathbf{x}_{n_{l^*l}});\ (n_{l^*} \parallel n_l)])$
$[] ([]_{(l^*,m)\in\Gamma_{L^*,M}} [b_{l^*} \wedge b_m \rightarrow \mathrm{ca}(h_{l^*m}, \mathbf{e}_{n_{l^*m}}, \mathbf{x}_{n_{l^*m}});\ (n_{l^*} \parallel n_m)])$
$[] ([]_{(m^*,j)\in\Gamma_{M^*,J}} [b_{m^*} \wedge b_j \rightarrow \mathrm{ca}(h_{m^*j}, \mathbf{e}_{n_{m^*j}}, \mathbf{x}_{n_{m^*j}});\ n_{m^*}])$
$[] ([]_{(m^*,k)\in\Gamma_{M^*,K}} [b_{m^*} \wedge b_k \rightarrow \mathrm{ca}(h_{m^*k}, \mathbf{e}_{n_{m^*k}}, \mathbf{x}_{n_{m^*k}});\ n_{m^*}])$
$[] ([]_{(m^*,l)\in\Gamma_{M^*,L}} [b_{m^*} \wedge b_l \rightarrow \mathrm{ca}(h_{m^*l}, \mathbf{e}_{n_{m^*l}}, \mathbf{x}_{n_{m^*l}});\ (n_{m^*} \parallel n_l)])$
$[] ([]_{(m^*,m)\in\Gamma_{M^*,M}} [b_{m^*} \wedge b_m \rightarrow \mathrm{ca}(h_{m^*m}, \mathbf{e}_{n_{m^*m}}, \mathbf{x}_{n_{m^*m}});\ (n_{m^*} \parallel n_m)])$

PROOF. (Sketch) To increase the readability of the proof, we often apply the termination transition rule (Rule 54) and consistency rule (Rule 55) of communication process term, and Lemma 3.5.6 without mentioning them explicitly. Similarly, we also apply the associativity property of the alternative composition without referring to Lemma B.3.4.

Let $R = \{(s_1 \parallel s_2, s_3) \mid s_1, s_2, s_3 \in N\} \cup \{(i_d, i_d) \mid i_d \in N\}$. The proof of the left implication of conditions 1 and 6 are similar to the proofs of the right implication of conditions 1 and 6. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4.

Condition 1: First, we assume $E \Vdash \langle (([]_{i\in I}\ u_i) [] ([]_{j\in J}\ b_j \rightarrow p_j) [] ([]_{k\in K}\ [b_k \rightarrow p_k]) [] ([]_{l\in L}\ b_l \rightarrow p_l;\ n_l) [] ([]_{m\in M}\ [b_m \rightarrow p_m;\ n_m])) \parallel (([]_{i^*\in I^*}\ u_{i^*}) [] ([]_{j^*\in J^*}\ b_{j^*} \rightarrow p_{j^*}) [] ([]_{k^*\in K^*}\ [b_{k^*} \rightarrow p_{k^*}]) [] ([]_{l^*\in L^*}\ b_{l^*} \rightarrow p_{l^*};\ n_{l^*}) [] ([]_{m^*\in M^*}\ [b_{m^*} \rightarrow p_{m^*};\ n_{m^*}])), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma' \rangle$ for some $E = (C, J, L, H, R), \sigma, \xi, a, \xi', \sigma'$, which means either Rule 28.1.l or Rule 28.1.r has been applied necessarily. Based on the deduction rule that has been applied and sub-process terms (in the alternative composition) of s_1 (or s_2) that perform a send action or receive action, we can distinguish more cases. Since the proofs for all cases are similar, we only give the proof for the following case. Rule 28.1.l has been applied, then (let us say) $(C, J \cup W, L, H, R) \Vdash \langle []_{j\in J}\ b_j \rightarrow p_j, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h_j,cs),\xi'} \langle \surd, \sigma' \rangle$, $(C, J, L, H, R) \Vdash \langle []_{j^*\in J^*}\ b_{j^*} \rightarrow p_{j^*}, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h_{j^*},cs,W),\xi'} \langle \surd, \sigma' \rangle$, $a = \mathrm{ca}(h_{jj^*}, cs)$, $p_j \equiv h_j\ !!\ \mathbf{e}_{n_j}$, $p_{j^*} \equiv h_{j^*}\ ??\ \mathbf{x}_{n_{j^*}}$, for

some $W, h_j = h_{j^*}, cs, \mathbf{e}_{n_j}, \mathbf{x}_{n_{j^*}}$, $\xi \models b_j$, $\xi \models b_{j^*}$ (see also Rule 20.1), $E \Vdash \langle (\llbracket_{i\in I}\ u_i) \rrbracket (\llbracket_{k\in K}\ [b_k \to p_k]) \rrbracket (\llbracket_{l\in L}\ b_l \to p_l;\ n_l) \rrbracket (\llbracket_{m\in M}\ [b_m \to p_m;\ n_m]), \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle (\llbracket_{i^*\in I^*}\ u_{i^*}) \rrbracket (\llbracket_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}]) \rrbracket (\llbracket_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) \rrbracket (\llbracket_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}]), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 25.1.l or 25.2.r, it is not relevant which rule has been applied, because the alternative composition is associative). Also, process term $[p]$ for any $p \in P$ is consistent with any extended valuation with respect to σ in any environment. Applying Rule 27 (many times) and from Rule 23, we further obtain $E \Vdash \langle \llbracket_{i\in I}\ u_i, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{k\in K}\ [b_k \to p_k], \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{l\in L}\ b_l \to p_l;\ n_l, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{m\in M}\ [b_m \to p_m;\ n_m], \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{i^*\in I^*}\ u_{i^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}], \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}], \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and we know $\xi \models b_l$ and $\xi \models b_{l^*}$, $E \Vdash \langle \llbracket_{l\in L}\ p_l, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, and $E \Vdash \langle \llbracket_{l^*\in L}\ p_{l^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 19) , respectively. From $\xi \models b_j$ and $\xi \models b_{j^*}$, we can have $\xi \models b_j \wedge b_{j^*}$. Using Rule 20.1, we obtain $(C, J, L, H, R) \Vdash \langle \llbracket_{(j,j^*)\in\Gamma_{J,J^*}}\ b_j \wedge b_{j^*} \to \mathrm{ca}(h_{jj^*}, \mathbf{e}_{n_{jj^*}}, \mathbf{x}_{n_{jj^*}}), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h_{jj^*},cs),\xi'} \langle \checkmark, \sigma'\rangle$, because process terms p_j and p_{j^*} communicate and this leads to a communication process term $\mathrm{ca}(h_{jj^*}, \mathbf{e}_{n_j}, \mathbf{x}_{n_{j^*}})$ and $cs = [\xi(\mathbf{e}_{n_{jj^*}})]$. Due to the above-mentioned consistency predicates, we can conclude that each sub-process term (let us say) s_i' of $s_3 \equiv s_0' \rrbracket \ldots \rrbracket s_i' \rrbracket \ldots \rrbracket s_n'$ is a consistency predicate for such E, σ, ξ, i.e. $\forall_{i:[0..n]}\ E \Vdash \langle s_i', \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 25.l.1 many times (or 25.1.r, it is not relevant which rule has been applied, because the alternative composition is associative), we get $E \Vdash \langle s_3, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h_{jj^*},cs),\xi'} \langle \checkmark, \sigma'\rangle$.
Condition 2: We assume $E \Vdash \langle ((\llbracket_{i\in I}\ u_i) \rrbracket (\llbracket_{j\in J}\ b_j \to p_j) \rrbracket (\llbracket_{k\in K}\ [b_k \to p_k]) \rrbracket (\llbracket_{l\in L}\ b_l \to p_l;\ n_l) \rrbracket (\llbracket_{m\in M}\ [b_m \to p_m;\ n_m])) \parallel ((\llbracket_{i^*\in I^*}\ u_{i^*}) \rrbracket (\llbracket_{j^*\in J^*}\ b_{j^*} \to p_{j^*}) \rrbracket (\llbracket_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}]) \rrbracket (\llbracket_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) \rrbracket (\llbracket_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}])), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E = (C, J, L, H, R), \sigma, \xi,\ a, \xi', \sigma', k_1$, which means either Rules 28.2.l, 28.2.r, 28.3.l, 28.3.r, 28.4.l, 28.4.r, 29.1.l, 29.1.r, 29.2.l, and 29.2.r. Based on the deduction rule that has been applied, we can distinguish more cases. Since the proofs for most cases are similar, we only give the proofs for the following cases:

- Rule 28.4.l has been applied. Then (let us say for the case that) we obtain $(C, J \cup W, L, H, R) \Vdash \langle\ \llbracket_{m\in M}\ [b_m \to p_m;\ n_m], \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h_m,cs),\xi'} \langle n_m, \sigma'\rangle$, $(C, J, L, H, R) \Vdash \langle\ \llbracket_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ \ n_{m^*}], \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h_{m^*},cs,W),\xi'} \langle n_{m^*}, \sigma'\rangle$, $a = \mathrm{ca}(h_{mm^*}, cs)$, $p_m \equiv h_m \,!!\, \mathbf{e}_{n_m}$, $p_{m^*} \equiv h_{m^*} \,??\, \mathbf{x}_{n_{m^*}}$, for some $W, h_m = h_{m^*}, cs, \mathbf{e}_{n_m}, \mathbf{x}_{n_{m^*}}$, $\xi \models b_m$, $\xi \models b_{m^*}$ (see also Rules 10.2, 16, 20.2), $E \Vdash \langle p_m, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle p_{m^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rules 19 and 23), $k_1 \equiv n_m \parallel n_{m^*}$, $E \Vdash \langle (\llbracket_{i\in I}\ u_i) \rrbracket (\llbracket_{j\in J}\ b_j \to p_j) \rrbracket (\llbracket_{k\in K}\ [b_k \to p_k]) \rrbracket (\llbracket_{l\in L}\ b_l \to p_l;\ n_l), \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle (\llbracket_{i^*\in I^*}\ u_{i^*}) \rrbracket (\llbracket_{j^*\in J^*}\ b_{j^*} \to p_{j^*}) \rrbracket (\llbracket_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}]) \rrbracket (\llbracket_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 25.1.l or 25.2.r, it is not relevant which rule has been applied, because the alternative composition is associative). Also, process term $[p]$ for any $p \in P$ is consistent with any extended valuation with respect to σ in any environment. Applying Rule 27 (many times) and from Rule 23 (also note that p_j, p_{j^*} are undelayable process terms), we further obtain $E \Vdash \langle \llbracket_{i\in I}\ u_i, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle \llbracket_{j\in J}\ b_j \to p_j, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash$

$\langle [\!]_{k\in K}\ [b_k \to p_k], \sigma\rangle \overset{\xi}{\rightsquigarrow}, E \Vdash \langle [\!]_{l\in L}\ b_l \to p_l;\ n_l, \sigma\rangle \overset{\xi}{\rightsquigarrow}, E \Vdash \langle [\!]_{i^*\in I^*}\ u_{i^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}, E \Vdash \langle [\!]_{j^*\in J^*}\ b_{j^*} \to p_{j^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}, E \Vdash \langle [\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}], \sigma\rangle \overset{\xi}{\rightsquigarrow}, E \Vdash \langle [\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, and we know $\xi \models b_j$, $\xi \models b_{j^*}$, $E \Vdash \langle p_j, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle p_{j^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rules 23 and 19), respectively. From $\xi \models b_m$ and $\xi \models b_{m^*}$, we can have $\xi \models b_m \wedge b_{m^*}$. Using Rule 20.2 and 16, we obtain $(C, J, L, H, R) \Vdash \langle [\!]_{(m,m^*)\in\Gamma_{M,M^*}}\ b_m \wedge b_{m^*} \to \mathrm{ca}(h_{mm^*}, \mathbf{e}_{n_{mm^*}}, \mathbf{x}_{n_{mm^*}});\ (n_m \parallel n_{m^*}), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h_{mm^*},cs),\xi'} \langle n_m \parallel n_{m^*}, \sigma'\rangle$, because process terms p_m and p_{m^*} communicate and this leads to a communication process term $\mathrm{ca}(h_{mm^*}, \mathbf{e}_{n_{mm^*}}, \mathbf{x}_{n_{mm^*}})$ and $cs = [\xi(\mathbf{e}_{n_{mm^*}})]$. Due to the above-mentioned consistency predicates, we conclude that each sub-process term (let us say) s_i' of $s_3 \equiv s_0' [\!] \ldots [\!] s_i' [\!] \ldots [\!] s_n'$ is a consistency predicate for such E, σ, ξ, i.e. $\forall_{i:[0..n]}\ E \Vdash \langle s_i', \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 25.2.l many times (or 25.2.r, it is not relevant which rule has been applied, because the alternative composition is associative), we get $E \Vdash \langle s_3, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h_{mm^*},cs),\xi'} \langle n_m \parallel n_{m^*}, \sigma'\rangle$. Take $k_2 \equiv n_m \parallel n_{m^*}$, and observe that $(k_1, k_2) \in R$.

- Rule 29.1.l has been applied. Then (let us say for the case that) we obtain $E \Vdash \langle [\!]_{j\in J}\ b_j \to p_j, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $\xi \models b_j$, $E \Vdash \langle [\!]_{j\in J}\ p_j, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ (see also Rule 20.1), $k_1 \equiv s_2$, $E \Vdash \langle ([\!]_{i\in I}\ u_i) [\!] ([\!]_{k\in K}\ [b_k \to p_k]) [\!] ([\!]_{l\in L}\ b_l \to p_l;\ n_l) [\!] ([\!]_{m\in M}\ b_m \to p_m;\ n_m), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 25.1.l or 25.2.r, it is not relevant which rule has been applied, because the alternative composition is associative), $E \Vdash \langle ([\!]_{i^*\in I^*}\ u_{i^*}) [\!] ([\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}]) [\!] ([\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) [\!] ([\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}], \sigma\rangle \overset{\xi}{\rightsquigarrow}$, and $E \Vdash \langle ([\!]_{i^*\in I^*}\ u_{i^*}) [\!] ([\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}]) [\!] ([\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) [\!] ([\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}], \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ (see also Rule 29.1.l). Again, we know that the alternative composition is associative, and process term $[p]$ for any $p \in P$ is consistent with any extended valuation with respect to σ in any environment. Due to the above-mentioned consistency predicates, it is not hard to see that we can conclude that each sub-process term (let us say) s_i' of $s_3 \equiv s_0' [\!] \ldots [\!] s_i' [\!] \ldots [\!] s_n'$ is a consistency predicate for such (E, σ, ξ) and (E, σ', ξ'), i.e. $\forall_{i:[0..n]}\ E \Vdash \langle s_i', \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\forall_{i:[0..n]}\ E \Vdash \langle s_i', \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 20.2 and 16, we can have $E \Vdash \langle [\!]_{j\in J}\ b_j \to p_j;\ s_2, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle s_2, \sigma'\rangle$. Using Rule 25.2.l many times (or 25.2.r, it is not relevant which rule has been applied, because the alternative composition is associate), we obtian $E \Vdash \langle s_3, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle s_2, \sigma'\rangle$. Take $k_2 \equiv s_2$, and observe that $(s_2, s_2) \in R$.

Condition 4: We assume $E \Vdash \langle ([\!]_{i\in I}\ u_i) [\!] ([\!]_{j\in J}\ b_j \to p_j) [\!] ([\!]_{k\in K}\ [b_k \to p_k]) [\!] ([\!]_{l\in L}\ b_l \to p_l;\ n_l) [\!] ([\!]_{m\in M}\ [b_m \to p_m;\ n_m])) \parallel (([\!]_{i^*\in I^*}\ u_{i^*}) [\!] ([\!]_{j^*\in J^*}\ b_{j^*} \to p_{j^*}) [\!] ([\!]_{k^*\in `K^*}\ [b_{k^*} \to p_{k^*}]) [\!] ([\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) [\!] ([\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}])), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma,\ t, \rho, k_1, \sigma'$, which means Rule 30 has been applied necessary. Then, it is not hard to see that we have $E \Vdash \langle s_1, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle s_1, \sigma'\rangle$, $E \Vdash \langle s_2, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle s_2, \sigma'\rangle$, and $k_1 \equiv s_1 \parallel s_2$. We know that the alternative composition is associative, and process term $[p]$ for any $p \in P$ allows arbitrary time transitions, and thereby does not change. Followed by applying

Rule 26 (many times), we obtian $E \Vdash \langle [\!]_{i\in I}\ u_i, \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{i\in I}\ u_i, \sigma'\rangle$ (see also Rule 3), $E \Vdash \langle [\!]_{j\in J}\ b_j \to p_j, \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{j\in J}\ b_j \to p_j, \sigma'\rangle$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b_j$, $\exists_{s\in[0,t]}\, \rho(s) \models \neg b_j$ (see also Rule 22 and note that p_j is an undelayable process term, and some unimportant information for this proof is omitted), $E \Vdash \langle [\!]_{k\in K}\ [b_k \to p_k], \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{k\in K}\ [b_k \to p_k], \sigma'\rangle$ (see Rule 11), $E \Vdash \langle [\!]_{l\in L}\ b_l \to p_l;\ n_l, \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{l\in L}\ b_l \to p_l;\ n_l, \sigma'\rangle$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b_l$, $\exists_{s\in[0,t]}\, \rho(s) \models \neg b_j$ (see also Rule 22 and note that $p_l; n_l$ is an undelayable process term, and some unimportant information for this proof is omitted), $E \Vdash \langle [\!]_{m\in M}\ [b_m \to p_m;\ n_m], \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{m\in M}\ [b_m \to p_m;\ n_m], \sigma'\rangle$ (see Rule 10), $E \Vdash \langle [\!]_{i^*\in I^*}\ u_{i^*}, \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{i^*\in I^*}\ u_{i^*}, \sigma'\rangle$, $E \Vdash \langle [\!]_{j^*\in J^*}\ b_{j^*} \to p_{j^*}, \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{j^*\in J^*}\ b_{j^*} \to p_{j^*}, \sigma'\rangle$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b_{j^*}$, $E \Vdash \langle [\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}], \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}], \sigma'\rangle$, $E \Vdash \langle [\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}, \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}, \sigma'\rangle$, $\forall_{s\in(0,t)}\, \rho(s) \models \neg b_{l^*}$, $E \Vdash \langle [\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}], \sigma\rangle \xmapsto{t,\rho} \langle [\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}], \sigma'\rangle$. Due to the above-mentioned time transitions, it is not hard to see that we can conclude that each sub-process term (let us say) s'_i of $s_3 \equiv s'_0 \,[\!]\, \ldots \,[\!]\, s'_i \,[\!]\, \ldots \,[\!]\, s'_n$ can perform such a time transition for t, ρ, i.e. $\forall_{i:[0..n]}\ E \Vdash \langle s'_i, \sigma\rangle \xmapsto{t,\rho} \langle s'_i, \sigma'\rangle$. Applying Rule 26 (many times) to the sub-process terms of s_3, we conclude that $E \Vdash \langle s_3, \sigma\rangle \xmapsto{t,\rho} \langle s_3, \sigma'\rangle$. Take $k_2 \equiv s_3$ and observe that $(k_1, k_2) \in R$.

Condition 6: First, we assume $E \Vdash \langle ([\!]_{i\in I}\ u_i) \,[\!]\, ([\!]_{j\in J}\ b_j \to p_j) \,[\!]\, ([\!]_{k\in K}\ [b_k \to p_k]) \,[\!]\, ([\!]_{l\in L}\ b_l \to p_l;\ n_l) \,[\!]\, ([\!]_{m\in M}\ [b_m \to p_m;\ n_m])) \between ((([\!]_{i^*\in I^*}\ u_{i^*}) \,[\!]\, ([\!]_{j^*\in J^*}\ b_{j^*} \to p_{j^*}) \,[\!]\, ([\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}]) \,[\!]\, ([\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}) \,[\!]\, ([\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}])), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some E, σ, ξ, which means Rule 31 has been applied necessary. We know that the alternative composition is associative, and process term $[p]$ for any $p \in P$ is consistent with any extended valuation with respect to σ in any environment. Followed by applying Rule 27 (many times), we obtian $E \Vdash \langle [\!]_{i\in I}\ u_i, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle [\!]_{j\in J}\ b_j \to p_j, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $\xi \models b_j$ and $E \Vdash \langle [\!]_{j\in J}\ p_j, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 23), $E \Vdash \langle [\!]_{k\in K}\ [b_k \to p_k], \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see Rule 12), $E \Vdash \langle [\!]_{l\in L}\ b_l \to p_l;\ n_l, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $\xi \models b_l$ (see also Rule 23) and $E \Vdash \langle [\!]_{l\in L}\ p_l, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see Rules 23 and 19), $E \Vdash \langle [\!]_{m\in M}\ [b_m \to p_m;\ n_m], \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see Rule 12), $E \Vdash \langle [\!]_{i^*\in I^*}\ u_{i^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle [\!]_{j^*\in J^*}\ b_{j^*} \to p_{j^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $\xi \models b_{j^*}$ and $E \Vdash \langle [\!]_{j^*\in J^*}\ p_{j^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle [\!]_{k^*\in K^*}\ [b_{k^*} \to p_{k^*}], \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $E \Vdash \langle [\!]_{l^*\in L^*}\ b_{l^*} \to p_{l^*};\ n_{l^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $\xi \models b_{l^*}$ and $E \Vdash \langle [\!]_{l^*\in L^*}\ p_{l^*}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, and $E \Vdash \langle [\!]_{m^*\in M^*}\ [b_{m^*} \to p_{m^*};\ n_{m^*}], \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Due to the above-mentioned consistency predicates, it is not hard to see that we can conclude that each sub-process term (let us say) s'_i of $s_3 \equiv s'_0 \,[\!]\, \ldots \,[\!]\, s'_i \,[\!]\, \ldots \,[\!]\, s'_n$ is a consistency predicate for such E, σ, ξ, i.e. $\forall_{i:[0..n]}\ E \Vdash \langle s'_i, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 27 (many times) to the sub-process terms of s_3, we conclude that $E \Vdash \langle s_3, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

Curricula vitarum

K.L. Man

Ka Lok Man was born in Hong Kong, China, in 1969. From September 1991 through July 1998, he studied Electronic Engineering, specialization in neural networks and digital signal processing (DSP), Politecnico di Torino, Turin, Italy. From January 1997 through June 1999, he was a programmer and computer administrator in the Idem and Fiat Auto, respectively, Turin, Italy. From June 1999 through October 2000, he was a research assistant at the Electronic Design Automation Group, Department of Electrical and Computer Engineering, Politecnico di Torino, Turin, Italy. From September 1999 through August 2000, he was a visiting researcher at the VLSI/CAD Group, Department of Electrical and Computer Engineering, University of Colorado at Boulder, USA. From October 2000 through January 2002, he was a researcher in the Research and Development Center of STMicroelectronics, Agrate, Milan, Italy. From February 2002 through February 2006, he is a Ph.D. student at the Formal Methods Group, Department of Mathematics and Computer Science, specialization in process algebras and formal analysis of hybrid systems, Eindhoven University of Technology, The Netherlands.

R.R.H. Schiffelers

Ramon Robert Hubert Schiffelers was born on the 9th of September, 1976 in Heerlen, The Netherlands. In 1993, he finished the *HAVO* at the Sintermeerten College in Heerlen, followed by a study in Mechanical Engineering at the *HTS* in Heerlen from 1993 till 1995. From 1995-2001 he studied Mechanical Engineering at the Eindhoven University of Technology and graduated within the Systems Engineering group. During his master program, activities were focused on hybrid modelling and simulation of pipeless batch plants. After graduation, he started his Ph.D. project on formal specification of hybrid systems at the same group. This is a combined project with the Formal Methods group of the Department of Mathematics and Computing Science at the same university.