Probability and hiding in concurrent processes

Probability and Hiding in Concurrent Processes

Sonja Georgievska

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics). The author was employed at the Eindhoven University of Technology.

IPA Dissertation Series 2011-13
Printed by Eindhoven University of Technology Press Facilities

A catalogue record is available from the
Eindhoven University of Technology Library
ISBN: 978-90-386-2640-6

Probability and Hiding in Concurrent Processes

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 3 oktober 2011 om 16.00 uur

door

Sonja Georgievska
geboren te Strumica, Macedonië

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. J.C.M. Baeten
en
prof.dr. W.J. Fokkink

Copromotor:

dr. S. Andova

Acknowledgements

First of all, I would like to thank Jos Baeten and Suzana Andova for offering me a PhD position, and Wan Fokkink, for accepting to be my second promotor.

Jos Baeten, my first promotor, gave me as much freedom and support to push my limits in the search for "the truth", as a PhD researcher could wish for. I should be extremely lucky to have a future supervisor like Jos.

Wan Fokkink became my second promotor later in my PhD studies; nevertheless, he showed deep interest in my work. His comments and the discussions we had were very valuable for improving the quality and the correctness of the thesis.

The relationship with Suzana Andova, my "daily" supervisor, was indeed on a daily basis, and it is very difficult to summarize four exciting years in a few words. I am very thankful to Suzana for being always there for me, for all the good advices regarding science and papers, for the endless discussions, for the trust and support when they were needed, and for much more.

I thank Pedro D'Argenio, Jan Friso Groote, and Catuscia Palamidessi for accepting to be readers of my thesis, and for their useful comments, that improved the text even further. I thank Twan Basten for accepting to take part in the defense committee.

The first part of this thesis is largely based on joint research with Nikola Trčka; the outcome of that research influenced the direction of my thought for the rest of the PhD studies. I am very thankful to Nikola for the great collaboration. I owe him a lot for teaching me valuable things when I was a novice in formal methods and in scientific research in general.

I was also very lucky to have Jasen Markovski nearby in the first year; he also shared his knowledge unselfishly, even though he was very busy finishing his own PhD project. The discussions with Jasen and Nikola were always inspiring.

Many thanks to Erik de Vink for reading my papers, for the useful comments and advices, and for helping me organize my time in the last year of my PhD studies, when I was faced with two challenges simultaneously.

I would like to thank all members of the former Formal Methods group and the current MDSE group for the pleasant atmosphere. I especially appreciated some stimulating discussions during the Tuesday's "lunch talks" and the ordinary lunch chats. Special thanks to the former FM-ers and OAS-ers with whom I spent most of the time – Astrid, Bas, Francine, Frank, Harsh, Helle, Jing, Kees, Matthias, Meivan, Michiel, Mohammad, Paul, Pieter, Rob, Ronald, Ruurd, Simona, Tijn, Tim, Tineke, Uzma, and my colleagues mentioned above – for their kind help and advices on any matter.

I thank Joost-Pieter Katoen for inviting me to present my research in the MOVES seminar in Aachen, and Pieter Cuijpers for inviting me to give a talk at the IPA Fall Days.

During conferences, workshops and seminars I benefited from, or had interesting research discussions with: Miguel Andrés, Kostas Chatzikokolakis, Pepijn Crouzen, Pedro D'Argenio, Holger Hermanns, Joost-Pieter Katoen, Manuel Núñez, Peter van Rossum, Ana Sokolova, and Marielle Stoelinga. I also met many inspirational people during scientific events with whom the discussions may not have been on a particular research topic. It is not possible to provide a full list, but I thank them all.

I thank Ana, Biba & Dragan, Daniela, Maja, Meri & Jasen, Mile, Nataša & Žarko, Natka, Niksa, Peni & Goce, Sandra & Ace, Vesna & Vlado for simply being my friends – helping me resolve dilemmas, make decisions, and enjoying life. I am especially happy to be able to call most of you "friends for many years".

I thank Marija, Nadežda, Simona and Suzana for their generous help with baby-issues, which also eased the writing of this thesis.

I thank my parents Violeta and Dragi for always supporting me to follow my dreams. I thank my sister Radmila, for always being there for me, and making me feel a very rich person. I thank Zoki and Maja & Zo for being so great, both as family-in-law and as friends!

Finally, I thank my husband Zvezdan and my daughter Kristina. For the things that really matter.

Sonja Georgievska

Eindhoven, August 2011

Summary

Action hiding and probabilistic choice have independently established their roles in process algebraic modeling and verification of concurrent systems. While action hiding allows abstraction from unimportant details and model reduction, and the induced nondeterminism enables modeling uncertainty in the system behaviour, probabilistic choice allows quantification of nondeterminism. However, as not all of the nondeterministic behaviour has a random nature, we are faced with the challenge to combine the above two aspects of concurrent systems, such that one can take maximal advantage of both.

This thesis addresses two problems regarding concurrent processes that exhibit both hidden and probabilistic behaviour, or *probabilistic processes* for short. Namely, a proper reduction of a model, by elimination of the hidden actions, requires a semantical equivalence that preserves the process properties of interest and is a congruence for the process operators. For non-probabilistic processes it has been shown that such an equivalence is *branching bisimilarity*. However, in the presence of probabilistic choice, more concretely in the *alternating* model of probabilistic processes, the intuitive notion of branching bisimulation is not a congruence for parallel composition. In this thesis a new branching bisimulation for this model is defined, and it is shown that this is the coarsest congruence for parallel composition that is included in the former. To achieve the congruence result, a hidden action preceding directly a non-trivial probabilistic choice cannot be eliminated. The new branching bisimulation preserves the properties expressible in the probabilistic *computation tree logic*, and is decidable in polynomial time. Similar to the non-probabilistic case, a single axiom characterizes branching bisimilarity for finite probabilistic processes.

The previous results imply that branching bisimilarity, although potentially useful for model reduction, may be in fact too strong to serve as an equivalence relation for probabilistic processes. Another view, taken in the *may/must testing* theory (as well as in the process calculus CSP), is to distinguish two processes only if they can be distinguished when interacting with their environment, i.e. with another process. However, although processes

that differ only in the moment an internal (nondeterministic) choice is made are not distinguished by this theory, for probabilistic processes this is no longer valid. The problem stems from an earlier observation that the *schedulers* that resolve the nondeterminism in concurrent probabilistic processes are too powerful and yield unrealistic overestimations of the probabilities with which a process can pass a test. The power of the schedulers comes from the fact that they allow the same choice to be resolved in different manners in different futures. In order to restrict the schedulers and thus to obtain the right probabilities, this thesis proposes integrating the information, based on which a nondeterministic choice is resolved, in labels on the nondeterministic transitions. In this way, choices using the same information are resolved in the same way, regardless of the considered future. As a result, the new testing preorder relation can be characterized by a probabilistic *ready-trace* preorder, a relation that is insensitive to the moment an internal choice is made, yet sensitive to deadlock and to action priorities. In other words, it combines useful features of both the bisimulation-style and the trace-style relations. The parallel composition is also generalized here to include both interleaving and action hiding after synchronization, and it is shown that probabilistic ready-trace preorder is a precongruence with respect to it. Finally, the CSP-style axiomatic characterization shows that all the distributivity laws for nondeterministic choice from CSP are preserved and no new laws are added.

Contents

Chapter 1

Introduction

This chapter intends to provide a general introduction to the field of concurrency theory, more particular to process algebras, process semantics, and their probabilistic extensions, in order to position the present work. It also briefly explains the motivations of the author for conducting the research presented in the thesis, and connects the two parts of the thesis. The contributions and the structure of the thesis are given and the papers on which this thesis is based are stated.

Depending on the preferences, a reader may skip this chapter and proceed directly to Part I or Part II, as each part has its own introduction.

1.1 Background

Concurrent processes are processes which execute in parallel and potentially interact with each other or share resources. Different mathematical formalisms have been developed for modeling and analysis of concurrent processes, such as Petri nets, process algebras, temporal logics, etc. In process algebras (e.g. CCS [85, 86], CSP [25, 71], ACP [10, 20]), processes are represented by so-called labeled transition systems: the process can undergo several states while executing, and a transition from one state to another can be made by performing actions (see e.g. Fig. 1.1). Processes can be composed in several ways, most notably via *parallel* operators, that can capture various ways of interactions: processes may synchronize on a given set of actions and perform the rest of the actions independently [71], or may synchronize via a handshaking mechanism that *hides* the synchronized action, i.e. makes it invisible [85], or may synchronize in a general way, via a predefined communication function [10]. No matter how the parallel composition is performed, it usually gives rise to a considerable amount of *nondeterminism*, which is an essential factor that makes the analysis of concurrent processes complex.

The nondeterminism causes many possible execution paths at every state of the composed process. This makes the problem of "when to equate two processes", or "how to check whether the *implementation* conforms to the *specification*", rather complicated [58]. Namely, since two processes may have different internal structures, but exhibit the same behaviour, the question is

> "How much of the internal structure of processes can be ignored when processes are compared?"

Up to this moment, there is no consensus on the right answer to this question (see e.g. [88]), or *the right semantical equivalence*, but there are several well-argued approaches. The *bisimulation equivalence* [85] relates two processes, or states, only if they can mimic each other's action steps by progressing again to related sub-processes, or states. The internal branching structure is thus preserved by related processes. An argument in favor of bisimulation is that this approach is safe and robust to adding new process operators [60, 85]. Moreover, two states are bisimilar if and only if they satisfy the same formulas of the well-known modal logic CTL [35] for describing properties of the systems; in other words, bisimilarity is completely characterized by CTL (as shown in [26]). Other approaches [25, 39] have more relaxed criteria: processes are distinguished only if they can be distinguished when being tested by the environment, that is, by interacting with other processes. The internal structure plays a less significant role in this case. Yet the least strict approach is to equate two processes only if they show the same observable behaviour, viz. have the same sets of traces (a trace being a sequence of performed actions) [71][1]. Figure 1.1 gives several processes and relations between them with respect to different semantical equivalences: bisimilarity, testing equivalence [39], and trace equivalence. See [58] for an extensive study of various process equivalences and their properties.

Even more discrepancies on which processes are equivalent occur when taking into consideration that some actions in the system are *internal* (invisible, or hidden). Such actions occur, for example, when two actions have synchronized and the resulting action is no longer able to synchronize with other actions [85] and thus becomes hidden (e.g. sending and receiving of a message). Thus, within the bisimulation approach, we have *weak* bisimulation [85] (originally called observational equivalence), and *branching* bisimulation [62].[2] Weak bisimulation relaxes the conditions of (strong) bisimulation by allowing internal τ-actions to precede or follow the observable action

[1] Trace equivalence stems from language equivalence in automata theory, while in concurrency theory it is of tangential interest.

[2] Other types of bisimulations have been proposed, too, but those two are the most popular.

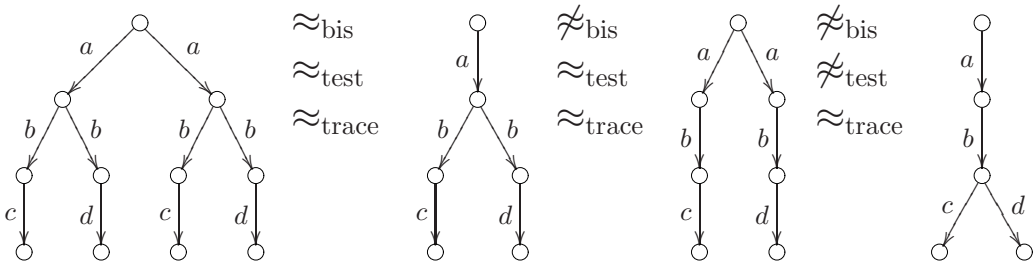

Figure 1.1: Several processes and relations between them w.r.t different equivalences

when simulating an action step. By branching bisimulation, however, the internal actions themselves must connect related states. It has been argued that adding the latter criterium to weak bisimulation preserves the branching structure better [60, 62]. Moreover, branching bisimilarity is completely characterized by the logic CTL without the "next" operator [40].

Originally, the focus in concurrency theory was only on modeling qualitative properties of systems. Due to the presence of unreliable components, but also because many protocols use randomization to achieve their goals, probabilistic behaviour started being considered in processes [34, 53, 66, 80, 107]. In the beginning, it was usual to assume that all nondeterminism has a random nature [14,34,53]. However, the range of applications in this way is narrowed, for example nondeterminism might be due to decision making (as in Markov Decision Processes [19, 73]), and thus cannot be treated as random. When a consensus was reached that both probabilistic and nondeterministic choice are important for modeling concurrent processes, research was spanned on several relevant questions: how to add probabilistic behaviour on the top of labeled transition systems [37,66,93,97], how to define plausible operators for composing processes [3, 37, 66, 97], and how to properly extend the existing semantical equivalences in the new setting (see [68, 97, 108] for early work). See [43, 102, 109] for extensive overviews of research in these topics. Yet, there are still open questions, and in this thesis we address some of them.

1.2 Motivations and the approaches

Several models for extending labeled transition systems with probabilistic behaviour have been proposed [37, 66, 93, 97]. One of those that gave gained attention is the *alternating model* [66,68]. The probabilistic transitions in [68]

have been added orthogonally to the action transitions: there are probabilistic states, originating probabilistic transitions, in addition to the nondeterministic states, originating action transitions (see e.g. Fig. 1.2). In [68] strong probabilistic bisimilarity has been defined, based on the probabilistic bisimulation defined in [80] for a more restricted model. Two states are bisimilar by [68] if they can mimic each other's action steps by proceeding again to bisimilar states (in the same style as in non-probabilistic bisimulation), or if they enter the same classes of equivalent states with the same probabilities. The parallel composition operator extends the CCS operator, such that the probabilistic transitions have precedence over the action transitions in parallel (see e.g. the parallel composition $s \parallel u$ of states s and u in Fig. 1.2). It has

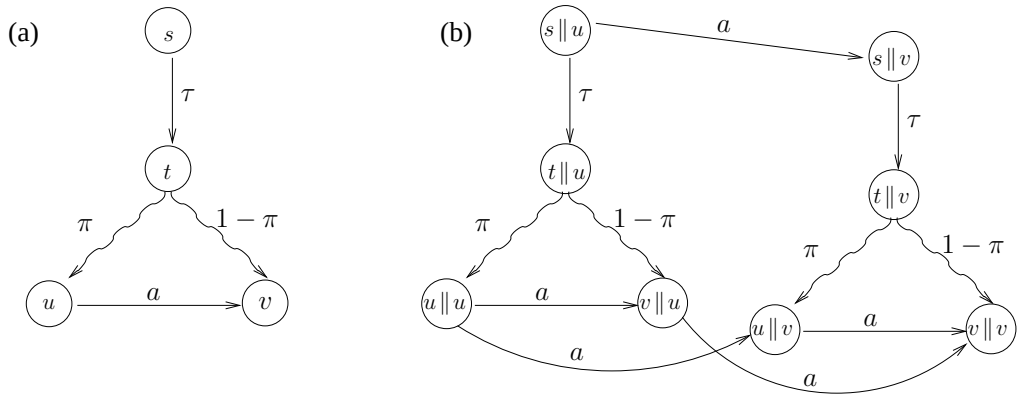

Figure 1.2: Probabilistic systems in the alternating model: (a) equivalent states s and t, (b) parallel composition and failure of the congruence property.

been shown that strong probabilistic bisimilarity is a congruence for parallel composition [68], meaning that it is preserved under parallel composition. This property is essential for equational reasoning about processes and for compositional analysis.

Later, notions of weak bisimulation and branching bisimulation for the alternating model were defined in [91], resp. [8]. However, although intuitive, they turned out not to be congruence relations w.r.t. parallel composition, as shown in [4]. For example, states s and t in Fig. 1.2 are related, but $s \parallel u$ and $t \parallel u$, their parallel compositions with state u, are not related by [8,91]. This is because by performing action a, state $s \parallel u$ can reach state $s \parallel v$, that eventually performs action a with probability π; on the other hand, state $t \parallel u$ by performing action a cannot reach a state that behaves as $s \parallel v$.

To solve the above problem, in this thesis we propose to restrict probabilistic branching bisimilarity of [8] such that it becomes a congruence for

parallel composition. Note that it is not unusual to restrict process equivalences to become congruences [22, 84, 85, 97]. The relation in [8] is strengthened in such a way that states s and t are no longer equivalent, or, in general, a τ-action preceding a probabilistic state with non-trivial distribution cannot be ignored. Further, we investigate the newly defined probabilistic branching bisimilarity from other aspects. We investigate whether it can be used as a model reduction technique, by providing a polynomial-time algorithm for deciding probabilistic branching bisimilarity and by showing that branching bisimilar states satisfy the same probabilistic CTL [67] formulas (without the "next" operator). We also give an axiomatic characterization of probabilistic branching bisimilarity, by which it becomes easily comparable to other process equivalences.

Although the new branching bisimilarity has nice properties, it distinguishes between states s and t, which is rather counterintuitive from the perspective of "the right process equivalence". Another way to solve the congruence problem discussed above is to allow states $s \parallel u$ and $t \parallel u$ in Fig. 1.2 to be related, as well as states s and t. However, note that in $t \parallel u$ the probabilistic choice occurs *before* any execution of action a, while in $s \parallel u$ this is not always the case. This means that we have to shift our attention to equivalences that are not sensitive to the exact moment an internal probabilistic choice occurs, as originally for non-probabilistic processes in [25, 39]. This, certainly, would bring us away from bisimulation-like equivalences.

It turned out, however, that finding an equivalence that has the above property and is compositional at the same time is far from trivial. To explain this, let us consider the following example. Player x tosses a fair coin without revealing the outcome and waits. Player y waits while the coin is being tossed, and then writes down his guess about the outcome of the flipping without showing it to x. Then, both players agree to reveal their outcomes, i.e. x to uncover the coin and y to show what he has written. Players x and y are modeled in Fig. 1.3. Processes synchronize on their common actions except on action ω reporting success, and the synchronized actions are hidden afterwards, resulting in process $x \parallel y$ in Fig. 1.3. Obviously, the probability that player y guesses correctly equals $\frac{1}{2}$. However, this is not what is suggested by process $x \parallel y$. From process $x \parallel y$ it follows that, if the process takes the left transition in the left-most nondeterministic choice and the right transition in the right-most nondeterministic process, then ω, or success, will *always* be reported, i.e. with probability 1. Note that this overestimation of probability to report ω occurs because the nondeterministic choice that y makes has been copied in both futures of the probabilistic choice of x in process $x \parallel y$. On the other hand, the composition of processes $\bar{x}$ and y (Fig. 1.3) yields that ω is reported with probability $\frac{1}{2}$ (or $\bar{x}$ *passes test* y with

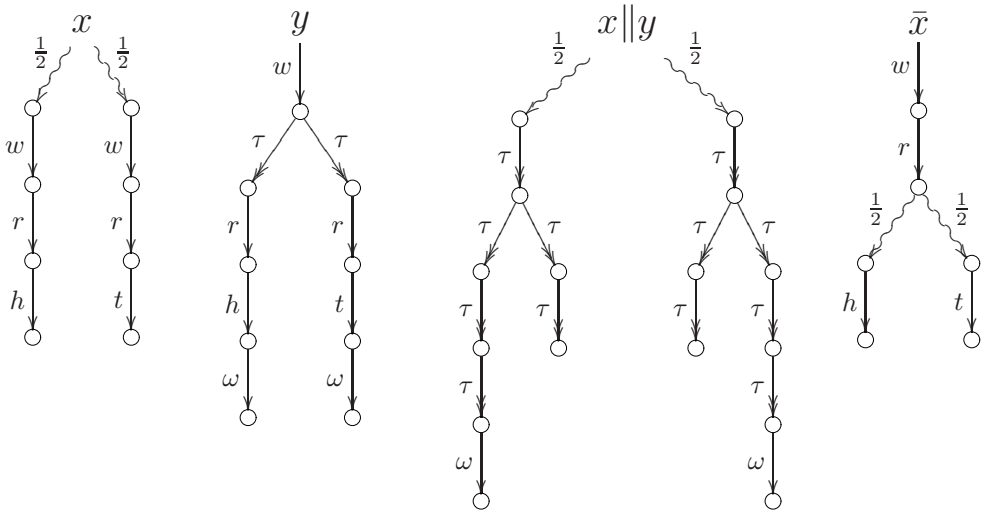

Figure 1.3: The coin-flipper and result-guesser game

probability $\frac{1}{2}$), that is, this time the right answer is obtained. Thus, due to above artefact of the parallel composition operator, processes x and $\bar{x}$ cannot be related by the probabilistic extensions [42, 44, 74, 90, 98, 108] of the testing theory of [39].[3] Thus, by the probabilistic extensions of the may/must testing theory, the moment an internal probabilistic choice is made is *observable*.

To solve the above problem, in this thesis we propose a *labeling method*. First, the τ-transitions are enriched with labels, by which they are identified in a parallel context. Thus, the resolution of a local nondeterministic choice is remembered in the parallel composition. Second, as nondeterministic choices arise due to parallel composition itself, we also propose how to properly label the new nondeterminism: each new nondeterministic choice is labeled with labels reflecting the information based on which it is resolved. Two choices, thus, that use the same information, will be resolved in the same manner, no matter where they appear in the considered process. Based on the labeling method, we define a testing semantics for probabilistic processes in the style of [39], aiming at obtaining realistic probabilities to pass a test, such as $\frac{1}{2}$ for $x \parallel y$ in Fig. 1.3 instead of 1. We then show that the new testing preorder relation can be characterized by a probabilistic version of the ready-trace preorder relation [13, 58, 92]. From this characterization it follows that the induced equivalence relation is insensitive to the moment an internal nondeterministic or probabilistic choice happens. We also explore how to extend the labeling method in case of a generalized parallel composition

[3] Variants of this example were initially discussed in [83, 87, 97].

(with both action interleaving and synchronization with hiding), to preserve the probability information about the composed system, but also to achieve compositionality of the ready-trace preorder. We give a CSP-style axiomatic characterization of the ready-trace equivalence, by which it becomes easily comparable to the other equivalences.

Remark The problem induced by using all-mighty *schedulers* for resolution of global nondeterminism, as illustrated by the example with the two players, has already been discussed from other points of view [9,31,55]. Namely, note that by using all-mighty schedulers one cannot prove that the probability that player y guesses correctly the outcome of coin-flipping is $\frac{1}{2}$. Thus, probabilistic verification of composed systems becomes difficult. This problem becomes especially apparent in the context of verifying security protocols (see e.g. [9, 31]), where usage of all-mighty schedulers deems classical protocols, which have otherwise been proven to be secure, as insecure. Thus, the solution proposed in this thesis can be also seen as a method to improve probabilistic verification of systems, by restricting the schedulers in order to obtain realistic estimates of the probabilistic behaviour of composed systems.

Remark In [56] it has been shown that the verification problem, for systems with infinite behaviour, is in general undecidable under schedulers with restricted power. However, in [57] it has been shown that usage of restricted schedulers allows for a more aggressive reduction of the state space than does usage of all-mighty schedulers. Thus, indirectly, by applying standard probabilistic verification (assuming all-mighty schedulers) on the reduced space, better estimates of the probabilistic behaviour are still obtained [57].

1.3 Contributions

In this thesis the following contributions are made.

1. We propose a new definition of branching bisimilarity for the alternating model of probabilistic systems [68] that is a congruence for parallel composition (Chapter 3);

2. We show that our probabilistic branching bisimilarity is the biggest equivalence relation that is a congruence for parallel composition and is included in the intuitive branching bisimilarity [8] for the same model (Chapter 3);

3. We give a polynomial-time algorithm for deciding our probabilistic branching bisimilarity (Chapter 4);

4. We show that probabilistic branching bisimilar states satisfy the same formulas of a probabilistic extension [67] of the modal logic CTL [35], without the "next" operator (Chapter 4);

5. We give an axiomatic characterization of probabilistic branching bisimilarity for finite processes for a process language in the style of ACP [10, 11] (Chapter 4);

6. We propose a new probabilistic extension of the may/must testing preorder [39], that is unsensitive to the exact moment an internal probabilistic or nondeterministic choice happens (Chapter 7);

7. We propose a labeling method to be applied on the alternating model of probabilistic systems such that realistic probabilities with which a process passes a test are obtained; the labels include the information based on which the internal choice is resolved (Chapter 7);

8. We propose a definition of probabilistic ready-trace preorder relation for our model (Chapter 7);

9. We show that the new probabilistic testing preorder relation and the probabilistic ready-trace preorder coincide (Chapter 7);

10. We propose a generalized parallel composition for our model, by defining a method for labeling the internal transitions arising from parallelism; inÍ this way, realistic estimates for the probabilistic behaviour of the composition are obtained;

11. We show that probabilistic ready-trace preorder is a precongruence for the generalized parallel composition (Chapter 8);

12. We give a CSP-style axiomatic characterization of probabilistic ready-trace equivalence, from which it follows that the distributivity and the idempotence properties of internal choice are preserved from CSP [71], and no new laws regarding the interplay between the different choice operators are added (Chapter 8).

1.3.1 Structure of the thesis

This thesis is divided into two parts, that can be read independently from each other. The first part includes the results 1–5 stated above, while the second part includes the results 6–12. Each part has an introductory chapter and a concluding chapter; each concluding chapter includes also a discussion on the related work to the research relevant to the particular part.

1.4 Origin of the thesis

The results presented in this thesis appeared before in several papers. Part I is based on papers [105], [6] and [7], while Part II is based on papers [50], [49], and the submission [52].

The results presented in the papers [106] and [51] also contributed in shaping the research presented in this thesis, although they are not included in the thesis. The research [106] partly motivated the need for a compositional branching bisimulation for the alternating model of probabilistic systems. The result that deterministic tests suffice to distinguish between processes (Chapter 7) originally did not appear in [50], but in [51], for a restricted model.

.

Part I

Branching-time semantics

Chapter 2

Introduction

One of the major benefits of process theory is the notion of abstraction and the corresponding equivalence relations defined on labeled transition systems. Abstraction, on the one hand, allows one to reason about systems in which details that are unimportant for the purposes at hand have been hidden. On the other hand, the corresponding equivalence relations allow for model reduction, which is often the only way to analyze complex or large systems. The efficiency of the analysis can be improved even further if this reduction technique is applied on the system's components before they are composed into a whole system model. This compositional analysis is particularly useful when the system consists of a number of interactive components.

In order to benefit from model reduction before analysis, several criteria have to be satisfied. First, it must be guaranteed that the properties of interest are preserved after the reduction. In other words, the equivalence relation used for model reduction must be sound with respect to the property specification language. Second, for a reduction method to be useful in practice, it is important that equivalence reduction can be performed efficiently. Finally, in order to apply modular reduction per component, it must be guaranteed that composition after reduction generates a model equivalent to the original one, namely, the equivalence relation must be preserved under composition. The compositionality, or congruence property of the equivalence is in fact essential for equational reasoning about processes.

Branching bisimulation equivalence for labeled transition systems [18,62], that abstracts away from internal steps, has the three properties listed above, and a number of other desirable features (see e.g. [60, 64]). In particular, branching bisimilarity is characterized by the logic CTL* without the "next" operator, as shown in [40].[1] While strong bisimilarity requires exact simu-

[1]Further on in the text the phrase "without the 'next' operator" is assumed implicitly

lation of the action transitions between the related system states, branching bisimilarity relaxes this condition: it allows the sequences of internal steps that possibly precede the action transition and connect equivalent states to be ignored. On the other hand, this relaxed condition can be seen as a restriction on top of the definition of weak bisimilarity [85], which allows ignorance of any sequence of internal steps that may precede the action transition. In other words, branching bisimilarity adds a *branching* condition to weak bisimilarity.

To model random behavior, several probabilistic extensions of transition systems have been proposed, that differ in the way they combine probability with nondeterminism (see [102] for an overview). One of the models that have attracted attention is the *alternating* model (see Figure 2.1), introduced in [107]. This model makes a distinction between nondeterministic states, in which nondeterministic choice is resolved, and probabilistic states, in which probabilistic choice is resolved according to a given distribution. In [68] a probabilistic process theory is defined on the alternating model, including, among others, the notions of parallel composition and communication. The definition of parallel composition, thereafter considered as standard, is based on the intuition that if a process p behaves as process p' with probability π, and process q behaves as process q' with probability ϱ, then the parallel composition $p\|q$ behaves as process $p'\|q'$ with probability $\pi\varrho$. For example, in Figure 2.1, $s\|u$ is the result of the parallel composition of processes s and u without communication.

The underlying semantics in [68] is based on a strong bisimulation in the style of [80]: action transitions are exactly simulated, while the related probabilistic states must have the same total probabilities to reach an arbitrary equivalence class. Abstraction and equivalence relations that abstract away from internal behavior were later defined in [91] and [8]. Reference [91] defines a probabilistic version of weak bisimilarity, and [8] strengthens the definition of [91] by adding the branching condition. A basic concept used in these definitions is the notion of a *scheduler*, which selects an action transition each time the process resides in a nondeterministic state, and thus yields a fully probabilistic process. An action transition then can be simulated by a set of scheduled paths that have a total probability one, and possibly include internal or probabilistic steps before the action itself. A scheduler is also used in the simulation of the probabilistic steps. Namely, the total probability to reach an equivalence class can be simulated by a set of scheduled paths, that have the same total probability to reach the corresponding class, and where the paths may include internal or probabilistic steps. For example, states s

when CTL or CTL* are referred to.

and t in Figure 2.1a are equivalent according to [91] and [8], because they have the same potential.

However, it has been shown later in [4] that the equivalence relations of [91] and [8] are not preserved by the parallel composition operator of [68]. This is explained in the following example.

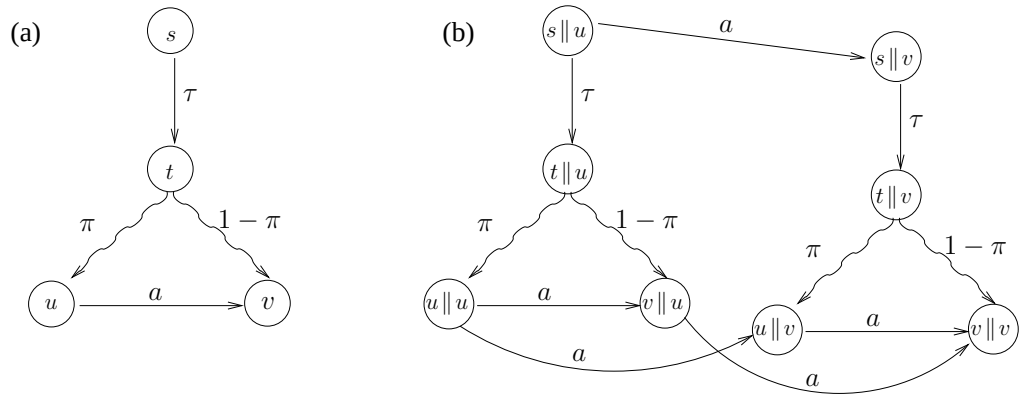

Figure 2.1: Probabilistic systems in the alternating model: (a) equivalent states s and t, (b) parallel composition and failure of the congruence property.

Example 2.0.1. Even though s and t in Figure 2.1 are equivalent, states $s\|u$ and $t\|u$ are neither equivalent with respect to the equivalence in [91], nor with respect to the equivalence in [8]. Namely, note that state $s\|u$ can perform action a and reach state $s\|v$, which may deadlock with probability $1-\pi$ (via the equivalent state $t\|v$). Thus, in order for state $t\|u$ to simulate this a-transition, there must be a scheduler starting in it, that generates a set of paths that perform action a (possibly via internal or probabilistic steps) with total probability 1, such that the states reached afterwards are equivalent to $s\|v$. Clearly, this condition is not satisfied by any of the two schedulers, that resolve the nondeterminism in different ways at state $u\|u$.

There are two ways to solve the congruence problem for a given equivalence. One way is to adapt the operator in question, in this case the parallel composition operator, by changing its semantics. However, this approach is rather radical as the current definition is well-established and natural. Another approach is to change the equivalence in consideration, preferably in such a way that the obtained notion is the coarsest congruence contained in the original, intuitive equivalence. For branching bisimulation this idea has already been employed several times, for instance in the extensions of non-probabilistic process theory with priorities [22] and with timing [104].

The same approach has been taken to achieve precongruence for parallel composition for the trace distribution inclusion relation on probabilistic automata [84, 97].

In this part we define a notion of branching bisimilarity for the alternating model of probabilistic systems that is a congruence for parallel composition, as well as for the rest of the standard operators in a probabilistic process algebra [3, 68]. The idea is to sufficiently strengthen the branching bisimilarity of [8] to achieve the congruence property. While action transitions are mimicked in a similar manner, by paths allowed to contain probabilistic as well as internal transitions, a stronger condition is imposed when mimicking probabilistic transitions, similar to the one for strong bisimilarity [80]. This condition implies that a probabilistic state that leads to different equivalence classes cannot be related to a nondeterministic state. Accordingly, for example, states s and t in Figure 2.1a are not branching bisimilar by our definition. Thus, we follow a similar line of reasoning as in [22], where non-probabilistic branching bisimilarity has been adapted to become compatible with action priorities. To justify our approach, we also show that this strengthened variant of probabilistic branching bisimilarity is the coarsest congruence contained in the equivalence of [8]. To make the comparison, we give a definition of our branching bisimilarity that involves schedulers, although they are not necessary in the original definition.

The branching bisimilarity defined here also has the other properties mentioned earlier, that make it suitable for practical implementation. We define an algorithm for deciding branching bisimilarity of polynomial time complexity $O(n^4)$ w.r.t. the number of states n of the model. We also present a probabilistic extension of the CTL modal logic, which is a variant of the pCTL logic of [23], and show that branching bisimilarity preserves all the properties expressible in this logic. To support usage of the equivalence in a process algebraic setting, we give a complete axiomatization for finite processes, where the process language contains a rich set of operators needed to reason on concurrent probabilistic processes: alternative composition, sequential composition, probabilistic choice, parallel composition with communication, hiding, and encapsulation. In particular, here the sequential composition using the termination constant is defined for the first time in a probabilistic setting. As an intermediate result we also give an alternative definition of branching bisimilarity based on colouring of the states [62], which shows how the branching structure of the processes is preserved. Regarding usage for simplification of systems, our branching bisimilarity may appear to be too strong at first, since, in general, it eliminates fewer τ-transitions than the one from [8]. However, the examples we provide illustrate that the equivalence

is still powerful enough for elimination of internal nondeterminism.

Structure of Part I In Chapter 3 we define our branching bisimilarity (Section 3.2) and show that it is compositional w.r.t. the merge operator, i.e. parallel composition without communication [68] (Section 3.3). We also show that it is the coarsest congruence for this operator that is included in the equivalence of [8] (Section 3.4). Then, in Chapter 4, we show the other characteristics of branching bisimilarity: we define the algorithm for partitioning the state space (Section 4.1), give the colouring definition (Section 4.2), show soundness for pCTL (Section 4.3) and provide a complete axiomatization (Section 4.4). Chapter 5 concludes Part I with a discussion on related work and concluding remarks.

Chapter 3

Compositional probabilistic branching bisimilarity

We define a notion of branching bisimilarity for the alternating model of probabilistic systems, compatible with parallel composition. For a congruence result, an internal transition immediately followed by a non-trivial probability distribution is not considered inert. A weaker, intuitive definition of branching bisimilarity for the same model has been defined by Andova & Willemse. Here we show that the proposed branching bisimilarity is the coarsest congruence for parallel composition that is included in the weaker version.

3.1 Probabilistic transition systems

As semantical model we use probabilistic transition systems that are based on the alternating model in [68], more specifically on the non-strictly alternating regime of [91]. The execution of the system can undergo two types of states: probabilistic and nondeterministic. In a probabilistic state a choice among the possible next nondeterministic states is made according to some probability distribution, while in a nondeterministic state an action transition is performed.

Given a directed graph, by $s \xrightarrow{l} t$ we denote that there is an edge originating from a node s and ending in a node t, labeled with l; we may omit s, t, or l from the notation to denote that they are arbitrary. Note that multiple equally labeled edges are possible between two nodes. We presuppose a finite set of action labels $\mathcal{A}$. Internal activity, as usual, is denoted by τ, and it is assumed that $\tau \notin \mathcal{A}$. We denote $\mathcal{A}_\tau = \mathcal{A} \cup \{\tau\}$.

Definition 3.1.1 (Probabilistic transition system)**.** A *probabilistic transition system* (PTS) is a finite-state and finite-transition directed graph, such that

(i) there are two types of *states* (or nodes): *nondeterministic* and *probabilistic*;

(ii) there are two types of *transitions* (or edges): *action* transitions, $\rightarrow$, originating from nondeterministic states and ending in arbitrary states, and *probabilistic* transitions, $\rightsquigarrow$, originating from probabilistic states and ending in nondeterministic states;

(iii) the action transitions are labeled with actions from $\mathcal{A}_\tau$;

(iv) the probabilistic transitions are labeled with scalars from $(0, 1]$, such that for each probabilistic state s, the sum of all the labels on the outgoing probabilistic transitions is equal to 1; that is, $\sum_{s \overset{\pi}{\rightsquigarrow}} \pi = 1$.

Given a PTS, by S_n, respectively by S_p, we denote the set of nondeterministic, respectively probabilistic states in the PTS, and we write S for $S_n \cup S_p$. A *deadlock* state without outgoing transitions, denoted by d, belongs to S_n. By $s \dashrightarrow t$ we denote that either $s \xrightarrow{\tau} t$ or $s \rightsquigarrow t$; $s \xrightarrow{(a)} t$ denotes that either $s \xrightarrow{a} t$, or $s = t$ and $a = \tau$.

To be able to reason about the probabilistic behaviour of a system specified by a PTS, the non-determinism that appears in the model must be first resolved by means of schedulers. The rest of this section is meant to give a concise presentation of the notion of scheduler, and other related notions. In the sequel we assume that a PTS is given.

Definition 3.1.2 (Paths)**.** An *infinite path* from a state $s_0 \in S$ is an infinite sequence $s_0\ l_1\ s_1\ \ldots$ such that $s_i \in S$, and $s_i \xrightarrow{l_{i+1}} s_{i+1}$ or $s_i \overset{l_{i+1}}{\rightsquigarrow} s_{i+1}$ for all $1 \leq i$. A *finite path* from a state s_0 is a finite sequence $s_0\ l_1\ s_1\ \ldots\ l_k\ s_k$ satisfying the same conditions as above. A *path* is a finite or infinite path. The set of all finite paths that start in a state s is denoted by $\mathrm{Paths_f}(s)$. The set of all finite paths is denoted by $\mathrm{Paths_f}$. Let $c = s_0\ l_1\ s_1\ \ldots\ l_k\ s_k$ be a finite path. We define $\mathrm{last}(c) = s_k$. The probability of c is the product of all probability labels on it, if any, or 1 otherwise, that is,

$$\mathrm{Prob}(c) = \begin{cases} \prod_{l_i \in (0,1]} l_i, & \text{if } l_j \in (0,1] \text{ for some } 1 \leq j \leq k \\ 1, & \text{otherwise.} \end{cases}$$

A scheduler resolves a nondeterministic choice in a nondeterministic state by selecting the next action to be executed. A scheduler can also stop an execution, which is denoted by assigning a $\perp$. In fact, as we will see, for a notion of branching bisimulation it is enough to consider only a certain type of finite paths, which can be extracted by allowing the scheduler to stop the execution when needed. If a path ends with a probabilistic state, a scheduler can either schedule nothing, in which case the next state of the execution is determined by the corresponding probability distribution, or it can schedule $\perp$ and thus stop the execution.

Definition 3.1.3 (Scheduler)**.** A *scheduler* is a partial function $\sigma \colon \mathrm{Paths_f} \mapsto (\rightarrow \cup \{\perp\})$, such that, if $\sigma(c) = s \xrightarrow{a} t$ for some $s, t \in S$ and $a \in \mathcal{A}_\tau$, then $\mathrm{last}(c) = s$.

Definition 3.1.4 (Scheduled paths)**.** Let σ be a scheduler. A *scheduled path* by σ is a finite path $s_0\ l_1\ s_1\ \ldots\ s_k$ or an infinite path $s_0\ l_1\ s_1\ \ldots$, where, for arbitrary i, $s_i \in S_n$ implies $\sigma(s_0\ l_1\ s_1\ \ldots\ s_i) = s_i \xrightarrow{l_{i+1}} s_{i+1}$, and for arbitrary i, if $s_i \in S_p$ then $\sigma(s_0\ l_1\ s_1\ \ldots\ s_i)$ is not defined, unless s_i is the last state of the scheduled path. A *maximal scheduled path* is either an infinite scheduled path, or a finite scheduled path c for which $\sigma(c) = \perp$. The set of all maximal paths scheduled by σ is denoted by $\mathrm{Paths_m}(\sigma)$.

Every scheduler σ induces a probability space on the set of all maximal scheduled paths that start in a state s. The probability measure Prob is defined by means of path prefixes and the cones induced by them in a usual way. The precise definitions and the measure property of the Prob function can be found in [8, 16, 94].

3.2 Branching bisimilarity for PTS

In this section we define a branching bisimulation relation on the set of states of a given PTS.

Recall from Fig. 2.1 that the problem with compositionality occurs when a probabilistic state with a nontrivial distribution (as t) is related to a nondeterministic state (as s). The parallel composition of state t with an action state will first resolve the probabilistic choice, while the parallel composition of state s with an action state can perform the action before resolving the probabilistic choice. However, the problem does not occur if the considered probabilistic state leads to equivalent states. We conclude that a nondeterministic state can be related to a probabilistic one only if the latter enters its own class with probability 1 via a probabilistic transition.

To formalize the above discussion, we first define a probability measure for an arbitrary state. Given a PTS with a set of states S, function $P : S \times S \to [0,1]$ is defined in the following way.

$$P(s,t) = \begin{cases} \sum_{s \overset{\pi}{\rightsquigarrow} t} \pi, & \text{if } s \in S_p, \\ 1, & \text{if } s \in S_n \text{ and } s = t, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, for a probabilistic state s, $P(s,t)$ gives the total probability to reach state t via one probabilistic transition, while for a nondeterministic state s, $P(s,s) = 1$ and $P(s,t) = 0$ for $t \neq s$. For a set $D \subseteq S$, we can now measure the total probability to reach an element in D from a given state $s \in S$ by

$$P(s,D) = \sum_{t \in D} P(s,t).$$

Given an equivalence relation R on a set X, we denote by $X_{/R}$ the partitioning of X induced by R, and, for an $x \in X$, we denote by $[x]_R$ the equivalence class of x.

Definition 3.2.1 (Branching bisimulation)**.** An equivalence relation $R \subseteq S \times S$ is a *branching bisimulation* iff for every $(s,t) \in R$ the following two conditions hold:

(i) if $s \xrightarrow{a} s'$ for $a \in \mathcal{A}_\tau$, then there exist $t_0, \ldots, t_n, t' \in S$ such that

- $t = t_0 \dashrightarrow t_1 \dashrightarrow \ldots \dashrightarrow t_n \xrightarrow{(a)} t'$,
- $(s, t_i) \in R$ for all $0 \leq i \leq n$, and
- $(s', t') \in R$,

(ii) for all $D \in S_{/R}$, $P(s,D) = P(t,D)$.

States s and t are *branching bisimilar*, denoted by $s \sim_b t$, iff $(s,t) \in R$ for some branching bisimulation relation R.

The first condition says that, as in [62], when an action transition is simulated, it can be preceded by a sequence of unobservable transitions that connect equivalent states. The second condition requires that all related states must have the same total probability to reach an equivalence class in one P-step, including their own equivalence class. It expresses, besides the rest, that for a probabilistic state to be related to a non-deterministic one, it must reach its own class with probability 1. This implies that a τ-step that is immediately followed by a nontrivial probability distribution is not considered inert, i.e. it cannot be ignored. Thus, due to this condition, states s and t in Fig. 2.1a cannot be related.

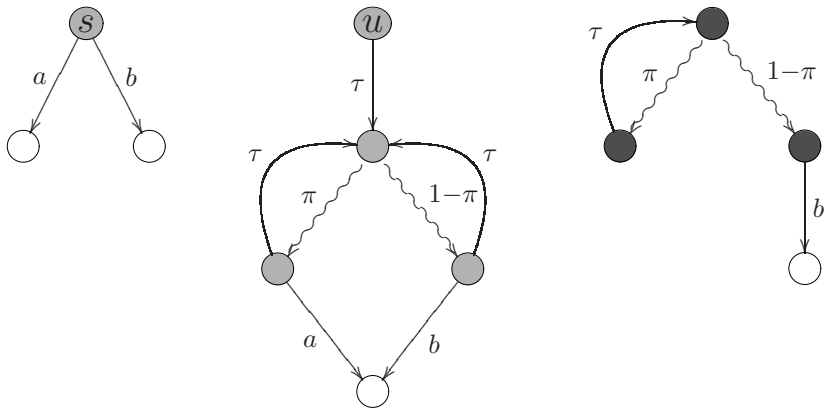

Figure 3.1: Examples of branching bisimilar states

Example 3.2.2. Figure 3.1 is an example of a PTS, where the bisimilar states are given the same colouring pattern. It can be seen that, although the definition of branching bisimilarity seems restrictive, probabilistic states can be related to nondeterministic states in rather nontrivial systems.

We proceed by showing that relation $\sim_b$ is itself a branching bisimulation. First we formally state that a probabilistic state related to a nondeterministic state cannot escape its own class via a probabilistic transition.

Lemma 3.2.3. *Let $R \subseteq S \times S$ be a branching bisimulation and let $s \in S_p$. If, for any $t \in S_n$, $(s,t) \in R$, then $P(s,[s]_R) = 1$.*

Proof. From $t \in S_n$ we have $P(t,t) = 1$. Therefore, $P(t,[t]_R) = 1$. From Def. 3.2.1 and $s \in [t]_R$, we have $P(s,[s]_R) = P(t,[s]_R) = P(t,[t]_R) = 1$. □

The following proposition plays an essential role in the proof that $\sim_b$ is a branching bisimulation.

Proposition 3.2.4. *Let $\{R_i\}_{i\in I}$ be a set of branching bisimulations. Then, $R = \left(\bigcup_{i\in I} R_i\right)^*$, the transitive closure of their union, is again a branching bisimulation.*

Proof. Since R_i, for every $i \in I$, is an equivalence relation, it follows that R is also an equivalence relation. Let $i \in I$. By definition, if $(s,t) \in R_i$ then $(s,t) \in R$. Therefore, every class in $S_{/R_i}$ is contained in some class in $S_{/R}$. Moreover, it follows that every class $D \in S_{/R}$ is a union of classes in $S_{/R_i}$, i.e. $D = \bigcup_{j\in J_i} D_i^j$ for some index set J_i, where $D_i^j \in S_{/R_i}$ for each $j \in J_i$.

Suppose $(s,t) \in R$. Then, there is some $n > 0$ such that $(s,t) \in \left(\bigcup_{i\in I} R_i\right)^n$. By induction on n we prove that s and t satisfy the conditions of Def. 3.2.1.

Suppose $n = 1$. Then $(s,t) \in \bigcup_{i\in I} R_i$. This means that there exists $h \in I$ such that $(s,t) \in R_h$.

(i) Assume that $s \xrightarrow{a} s'$. Then, since $(s,t) \in R_h$, there exist $t_1, \ldots, t_m$ (for some $m > 0$) and t', such that $t \dashrightarrow t_1 \dashrightarrow \ldots \dashrightarrow t_m \xrightarrow{(a)} t'$, $(s,t_i) \in R_h \subseteq R$ for all $1 \leq i \leq m$, and $(s',t') \in R_h \subseteq R$.

(ii) Let $D \in S_{/R}$. By the above discussion, $D = \bigcup_{j\in J_h} D_h^j$ for some index set J_h, where each D_h^j is a class in $S_{/R_h}$. Then,

$$P(s,D) = \sum_{j\in J_h} P\left(s, D_h^j\right) = \sum_{j\in J_h} P\left(t, D_h^j\right) = P(t,D).$$

Suppose now that $n > 1$. We assume that for all $k < n$ it holds that, if $(u,v) \in \left(\bigcup_{i\in I} R_i\right)^k$, then

(i) if $u \xrightarrow{a} u'$ then there exist $v_0, \ldots, v_m$ (for some $m > 0$) and v' such that $v = v_0 \dashrightarrow v_1 \dashrightarrow \ldots \dashrightarrow v_m \xrightarrow{(a)} v'$, $(u,v_i) \in R$ for all $0 \leq i \leq m$, and $(u',v') \in R$, and

(ii) $P(u,D) = P(v,D)$ for all $D \in S_{/R}$.

By assumption, $(s,t) \in \left(\bigcup_{i\in I} R_i\right)^n$. Then, there exists $r \in S$ such that $(s,r) \in \left(\bigcup_{i\in I} R_i\right)^{n-1}$, while $(r,t) \in R_h$ for some $h \in I$.

(i) Assume $s \xrightarrow{a} s'$. By the inductive assumption, there exist $r_0, r_1, \ldots, r_m, r'$ such that $r = r_0 \dashrightarrow r_1 \dashrightarrow \ldots \dashrightarrow r_m \xrightarrow{(a)} r'$, $(s,r_i) \in R$ for all $0 \leq i \leq m$, and $(s',r') \in R$. Now, from $(r,t) \in R_h$, by induction on m we show that there exist $t_0, \ldots, t_l$ (for some $l > 0$) and t' such that $t = t_0 \dashrightarrow t_1 \dashrightarrow \ldots \dashrightarrow t_l \xrightarrow{(a)} t'$, $(r,t_i) \in R$ for all $0 \leq i \leq l$, and $(r',t') \in R$, which suffices.

Suppose $m = 0$. Then $r \xrightarrow{(a)} r'$. The proof follows from the facts that $(r,t) \in R_h$, which is a branching bisimulation, and $R_h \subseteq R$.

Suppose now that $m > 0$. We distinguish two cases: when $r_0 \xrightarrow{\tau} r_1$ and when $r_0 \rightsquigarrow r_1$.

Assume first that $r_0 \xrightarrow{\tau} r_1$. Then, from $(r_0,t) \in R_h$ and because R_h is a branching bisimulation, it follows that there exist $t_0, t_1, \ldots, t_k$ such that $t = t_0 \dashrightarrow t_1 \dashrightarrow \ldots \dashrightarrow t_{k-1} \xrightarrow{(\tau)} t_k$, $(r,t_i) \in R_h \subseteq R$ for all $0 \leq i < k$ and $(r_1,t_k) \in R_h \subseteq R$. The rest follows by the inductive assumption, using that $(r,r_1) \in R$ and that R is an equivalence.

Assume now that $r_0 \rightsquigarrow r_1$. There are two subcases: when $t \in S_n$ and when $t \in S_p$. In the first case, from Lemma 3.2.3 it follows that $P(r, [t]_{R_h}) = 1$, from which it follows that $(t, r_1) \in R_h \subseteq R$. The rest follows by the inductive assumption. In the second case, when $t \in S_p$, by the second condition of Def. 3.2.1, there must exist $t_1 \in S$ such that $t \rightsquigarrow t_1$, and $(r_1, t_1) \in R_h \subseteq R$. The rest follows by the inductive assumption.

(ii) It is left to show that $P(s, D) = P(t, D)$ for all $D \in S_{/R}$. Let $D \in S_{/R}$. Since $(s, r) \in \left(\bigcup_{i\in I} R_i\right)^{n-1}$, by the inductive assumption it follows that for all $D \in S_{/R}$ it holds $P(s, D) = P(r, D)$. On the other hand, since $(r, t) \in R_h$, and $D = \bigcup_{j\in J_h} D_h^j$ for some index set J_h, where each $D_h^j \in S_{/R_h}$, we have that

$$P(r, D) = \sum_{j\in J_h} P\left(r, D_h^j\right) = \sum_{j\in J_h} P\left(t, D_h^j\right) = P(t, D).$$

Therefore, $P(s, D) = P(r, D) = P(t, D)$.

Thus, R is a branching bisimulation. □

Theorem 3.2.5. *Relation $\sim_b$ is a branching bisimulation.*

Proof. Let $\{R_i\}_{i\in I}$ be the set of all branching bisimulations. By definition,

$$\sim_b = \bigcup_{i\in I} R_i. \tag{3.1}$$

From Proposition 3.2.4 we have that $\left(\bigcup_{i\in I} R_i\right)^*$ is a branching bisimulation. Therefore,

$$\left(\bigcup_{i\in I} R_i\right)^* \subseteq \sim_b . \tag{3.2}$$

On the other hand, we have that

$$\bigcup_{i\in I} R_i \subseteq \left(\bigcup_{i\in I} R_i\right)^* . \tag{3.3}$$

From (3.1), (3.2), and (3.3) we obtain that

$$\sim_b = \left(\bigcup_{i\in I} R_i\right)^* ,$$

i.e. $\sim_b$ is a branching bisimulation.

□

3.3 Compositionality

In this section we give the definition of the merge operator (parallel composition without communication) [68] for probabilistic transition systems, and prove that $\sim_b$ is compositional with respect to this operator. The results extend to a parallel composition with communication in a straightforward way. In Section 8.1

Definition 3.3.1 (Merge)**.** The operation *merge* transforms a PTS with set of states S into a PTS with set of states $S \times S$, whose transitions are defined as follows (we standardly write $s \parallel t$ instead of (s,t)):

1. $s \parallel t \xrightarrow{a} u$ iff $s,t \in S_n$ and
   - there exists $s' \in S$ such that $s \xrightarrow{a} s'$ and $u = s' \parallel t$, or
   - there exists $t' \in S$ such that $t \xrightarrow{a} t'$ and $u = s \parallel t'$; and
2. for all $\pi \in (0,1]$, $s \parallel t \overset{\pi}{\rightsquigarrow} u$ iff
   - $t \in S_n$, there exists $s' \in S$ such that $s \overset{\pi}{\rightsquigarrow} s'$, and $u = s' \parallel t$, or
   - $s \in S_n$, there exists $t' \in S$ such that $t \overset{\pi}{\rightsquigarrow} t'$, and $u = s \parallel t'$, or
   - there exist $\pi_1, \pi_2 \in (0,1]$ and $s', t' \in S$, such that $s \overset{\pi_1}{\rightsquigarrow} s'$, $t \overset{\pi_2}{\rightsquigarrow} t'$, $\pi = \pi_1\pi_2$, and $u = s' \parallel t'$.

Example 3.3.2. As already stated, state $s \parallel u$ in Figure 2.1 is the merge of the states s and u. Figure 3.2 gives an example of a merge (state v) of two probabilistic states, s and u.

The next lemma shall be needed in the proof of the congruence theorem.

Lemma 3.3.3. *For all* $s,t,s',t' \in S$, $P(s \parallel t, s' \parallel t') = P(s,s') \cdot P(t,t')$.

Proof. We distinguish four cases, depending on whether s and t are nondeterministic or probabilistic states. In case both s and t are nondeterministic, we have $P(s,s') \cdot P(t,t') = 1$ if $s = s'$ and $t = t'$, and $P(s,s') \cdot P(t,t') = 0$, otherwise. From Def. 3.3.1 we have $P(s \parallel t, s' \parallel t') = 1$ if $s \parallel t = s' \parallel t'$, that is, if $s = s'$ and $t = t'$, and $P(s \parallel t, s' \parallel t') = 0$, otherwise. In case s is nondeterministic and t is a probabilistic state, we have $P(s,s') = 1$ if $s = s'$, and $P(s,s') = 0$, otherwise, and $P(t,t') = \sum_{t \overset{\rho}{\rightsquigarrow} t'} \rho$. Thus, $P(s,s') \cdot P(t,t') = \sum_{t \overset{\rho}{\rightsquigarrow} t'} \rho$ if $s = s'$, and $P(s,s') \cdot P(t,t') = 0$, otherwise. The case when t is nondeterministic and s is a probabilistic state is similar

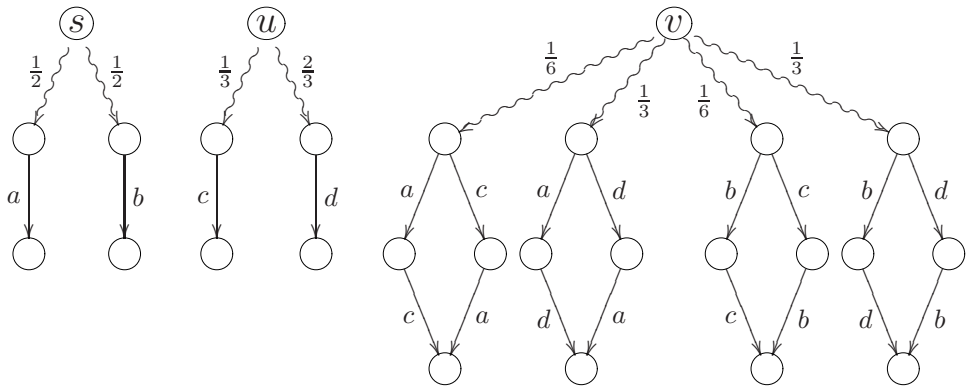

Figure 3.2: State v as the merge of states s and u.

to the previous one. In case both s and t are probabilistic states, we have $P(s,s') = \sum_{s \overset{\pi}{\rightsquigarrow} s'} \pi$ and $P(t,t') = \sum_{t \overset{\rho}{\rightsquigarrow} t'} \rho$. Thus,

$$P(s,s') \cdot P(t,t') = \sum_{s \overset{\pi}{\rightsquigarrow} s'} \pi \sum_{t \overset{\rho}{\rightsquigarrow} t'} \rho = \sum_{s \overset{\pi}{\rightsquigarrow} s'} \sum_{t \overset{\rho}{\rightsquigarrow} t'} \pi\rho.$$

From Def. 3.3.1 we have that

$$P(s \parallel t, s' \parallel t') = \sum_{s \overset{\pi}{\rightsquigarrow} s'} \sum_{t \overset{\rho}{\rightsquigarrow} t'} \pi\rho,$$

and thus the proof is complete. □

Theorem 3.3.4 (Congruence theorem)**.** *Branching bisimilarity $\sim_b$ is a congruence with respect to the merge operator, i.e. if $s \sim_b t$ and $u \sim_b v$ then $s \parallel u \sim_b t \parallel v$.*

Proof. Let $R = \{(s \parallel u, t \parallel v) \mid s,t,u,v \in S,\ s \sim_b t,\ u \sim_b v\}$. We show that R is a branching bisimulation relation. It is clearly an equivalence relation. Let $s,t,u,v \in S$ be such that $(s \parallel u, t \parallel v) \in R$.

(i) Suppose that $s \parallel u \xrightarrow{a} r$ for some $r \in S \times S$ and $a \in \mathcal{A}_\tau$. Without loss of generality, we can assume that $s \xrightarrow{a} s'$ for some $s' \in S$. Then $u \in S_n$ and $r = s' \parallel u$. From $s \sim_b t$ it follows that there exist $t_0, \ldots t_n, t' \in S$ such that $t_0 = t$, $t \sim_b t_i$ for $0 \leq i \leq n$, $s' \sim_b t'$, and $t_0 \dashrightarrow t_1 \dashrightarrow \ldots \dashrightarrow t_n \xrightarrow{(a)} t'$. By induction on n we show now that there exist $\bar{t}_0 = t, \bar{t}_1, \ldots, \bar{t}_k$, $\bar{v}_0 = v, \bar{v}_1, \ldots, \bar{v}_k$, and $\bar{r} \in S \times S$, such that $(t \parallel v, \bar{t}_i \parallel \bar{v}_i) \in R$ for $0 \leq i \leq k$, $(r, \bar{r}) \in R$, and $\bar{t}_0 \parallel \bar{v}_0 \dashrightarrow \bar{t}_1 \parallel \bar{v}_1 \dashrightarrow \ldots \dashrightarrow \bar{t}_k \parallel \bar{v}_k \xrightarrow{(a)} \bar{r}$. We distinguish two cases: when $v \in S_n$ and when $v \in S_p$.

– Assume first that $v \in S_n$. Suppose that $n = 0$. Then, there exists $t' \in S$ such that $t \xrightarrow{(a)} t'$ and $t' \sim_b s'$. From the last and from Definition 3.3.1 it follows that $t \parallel v \xrightarrow{(a)} t' \parallel v$, which was enough to prove. Suppose now that $n > 0$. If $t_0 \xrightarrow{\tau} t_1$, then we have $t_0 \parallel v \xrightarrow{\tau} t_1 \parallel v$. If $t_0 \rightsquigarrow t_1$, then $t_0 \parallel v \rightsquigarrow t_1 \parallel v$. The rest follows from the inductive assumption.

– Assume now that $v \in S_p$. From $v \sim_b u$ and Lemma 3.2.3, it follows that there exists $\bar{v} \in S_n$ such that $P(v, \bar{v}) > 0$ and $v \sim_b \bar{v}$. Suppose that $n = 0$. Then, there exists $t' \in S$ such that $t \xrightarrow{(a)} t'$ and $t' \sim_b s'$. If $t = t'$, then $t \parallel v \xrightarrow{(a)} t' \parallel v$. If $t \neq t'$, then $t \in S_n$, and $t \parallel v \rightsquigarrow t \parallel \bar{v} \xrightarrow{(a)} t' \parallel \bar{v}$. Suppose now that $n > 0$. If $t_0 \xrightarrow{\tau} t_1$, then $t_0 \parallel v \rightsquigarrow t_0 \parallel \bar{v} \xrightarrow{\tau} t_1 \parallel \bar{v}$, while if $t_0 \rightsquigarrow t_1$, then $t_0 \parallel v \rightsquigarrow t_1 \parallel \bar{v}$. In either case, the rest follows from the inductive assumption.

(ii) In the proof of the second condition, the most involved case is when $s \parallel u$ is a probabilistic state. Let $p, q \in S$ and $D = [p \parallel q]_R$. Then, using Lemma 3.3.3, we have

$$P(s \parallel u, D) = \sum_{\bar{p} \parallel \bar{q} \in D} P(s \parallel u, \bar{p} \parallel \bar{q}) = \sum_{\bar{p} \parallel \bar{q} \in D} P(s, \bar{p}) \cdot P(u, \bar{q}). \tag{3.4}$$

By the definition of R, we have

$$\begin{aligned} \sum_{\bar{p} \parallel \bar{q} \in D} P(s, \bar{p}) \cdot P(u, \bar{q}) &= \sum_{\bar{p} \sim_b p, \bar{q} \sim_b q} P(s, \bar{p}) \cdot P(u, \bar{q}) \\ &= \left(\sum_{\bar{p} \sim_b p} P(s, \bar{p}) \right) \cdot \left(\sum_{\bar{q} \sim_b q} P(u, \bar{q}) \right). \end{aligned} \tag{3.5}$$

Similarly as above, we obtain

$$P(t \parallel v, D) = \left(\sum_{\bar{p} \sim_b p} P(t, \bar{p}) \right) \cdot \left(\sum_{\bar{q} \sim_b q} P(v, \bar{q}) \right). \tag{3.6}$$

From $s \sim_b t$ and $u \sim_b v$, we have

$$\left(\sum_{\bar{p} \sim_b p} P(s, \bar{p}) \right) \cdot \left(\sum_{\bar{q} \sim_b q} P(u, \bar{q}) \right) = \left(\sum_{\bar{p} \sim_b p} P(t, \bar{p}) \right) \cdot \left(\sum_{\bar{q} \sim_b q} P(v, \bar{q}) \right). \tag{3.7}$$

From equations (3.4), (3.5), (3.6) and (3.7) we obtain that $P(s \parallel u, D) = P(t \parallel v, D)$. Thus, the proof is complete. □

3.4 The coarsest congruence result

In this subsection we present one of the main results, namely that branching bisimilarity $\sim_b$ is the coarsest congruence sub-relation of the equivalence relation (denoted here by $\leftrightarrows_b$) defined in [8]. The comparison of the two relations requires a characterization of $\sim_b$ in terms of schedulers. Thus, we also give an alternative definition of $\sim_b$.

3.4.1 Weaker branching bisimilarity

To avoid any confusion, in the sequel we refer to the branching bisimilarity $\leftrightarrows_b$ of [8], discussed in Chapter 2, as a *weaker branching bisimilarity* or *wb bisimilarity* in short. From now on, branching bisimilarity refers *only* to the $\sim_b$ relation (Def. 3.2.1). The major difference between $\leftrightarrows_b$ and $\sim_b$ is the following: in $\leftrightarrows_b$ a one-step probabilistic transition can be simulated by a set of internal paths, which is not the case with $\sim_b$. As for the simulation of an action transition, there are no essential differences, which will become clear in the next subsection.

We introduce several abbreviations. Assume that a PTS is given and that R is an equivalence relation on the set of states S of the PTS and $D \in S_{/R}$. By $s_0 \stackrel{a}{\Rightarrow} D$ we denote a silent path that traverses states equivalent to s_0 before performing an action a and reaching class D; in the case $a = \tau$, it is not necessary to perform the action. Formally, let $c = s_0\ l_1\ s_1\ \ldots\ l_k\ s_k$ be a finite path such that $s_k \in D$, and for all $1 \leq i \leq k-1$, $s_i \in [s_0]_R$ and $l_i = \tau$ if $l_i \in \mathcal{A}_\tau$. For a given $a \in \mathcal{A}$, we say that c *is of type* $s_0 \stackrel{a}{\Rightarrow} D$ if $l_k = a$. We say that c *is of type* $s_0 \stackrel{\tau}{\Rightarrow} D$ if either $k = 0$ or $l_k = \tau$ or $l_k \in (0, 1]$. For a scheduler σ, by

$$\mathrm{Paths_m}\,(\sigma)_{/t \stackrel{a}{\Rightarrow} D},$$

where $a \in \mathcal{A}_\tau$, we denote the set of all paths in $\mathrm{Paths_m}\,(\sigma)$, i.e. the maximal paths scheduled by σ, that are of type $t \stackrel{a}{\Rightarrow} D$. The probability function $\mu_R \colon S \times S_{/R} \mapsto [0, 1]$ is defined as:

$$\mu_R\,(s, D) = \begin{cases} \frac{P(s,D)}{1-P(s,[s]_R)}, & \text{if } s \in S_p,\ D \neq [s]_R, \text{ and } P\,(s, [s]_R) \neq 1 \\ 0, & \text{otherwise.} \end{cases}$$

Note that, when $s \in S_p$, $D \neq [s]_R$, and $P\,(s, [s]_R) \neq 1$, $\mu_R\,(s, D)$ represents the conditional probability with which state s reaches class D in one step, under condition that it leaves its own class $[s]_R$. However, in any other case $\mu_R\,(s, D)$ is defined and equal to zero.

Definition 3.4.1 (WB bisimulation [8])**.** An equivalence relation $R \subseteq S \times S$ is a *wb bisimulation* iff, for every $(s, t) \in R$ the following two conditions hold:

(i) if $s \xrightarrow{a} s'$ for some $a \in \mathcal{A}_\tau$ and $s' \in S$, then there is a scheduler σ such that

$$\mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \stackrel{a}{\Rightarrow} [s']_R}\right) = 1;$$

(ii) if $s \in S_p$, then there is a scheduler σ such that for all $D \in S_{/R} \setminus \{[s]_R\}$,

$$\mu_R\left(s, D\right) = \mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \stackrel{\tau}{\Rightarrow} D}\right).$$

s and t are wb bisimilar, denoted by $s \underline{\leftrightarrow}_b t$, iff there exists a wb bisimulation relation $R \subseteq S \times S$ such that $(s,t) \in R$.

We recap the conditions of the definition. An action transition can be simulated by a set of σ-scheduled paths, for some scheduler σ, that traverse silently through the equivalence class of the initial state before the same action is performed, as long as the probability of the set of all such σ-scheduled paths is 1. The probabilistic potential of a probabilistic state, to reach other equivalence classes, can be simulated if a single scheduler can be found, which generates silent paths through the equivalence class of the originating state before reaching other equivalence classes – of course, the probability of entering a certain equivalence class must match the corresponding probability for the related probabilistic state.

3.4.2 Comparing the two equivalences

In order to compare the two notions of bisimulation relations, we reformulate the definition of our branching bisimulation. The following lemma prepares the ground for the new alternative definition. Then, Theorem 3.4.4 redefines branching bisimulation in terms of schedulers.

Lemma 3.4.2. *Let $R \subseteq S \times S$ be a branching bisimulation relation and let $s \xrightarrow{a} s'$ for some $a \in \mathcal{A}_\tau$ and $s, s' \in S$. Let $t \in S$ such that $(s,t) \in R$. There exists a scheduler σ such that*

$$\mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \stackrel{a}{\Rightarrow} [s']_R}\right) = 1.$$

Proof. Since $s \xrightarrow{a} s'$ and $(s,t) \in R$, there exists at least one path of type $t \stackrel{a}{\Rightarrow} [s']_R$. From Def. 3.2.1 it follows that all probabilistic states on this path are related to s. Moreover, by Lemma 3.2.3 it follows that they enter only their own class with a probabilistic step. The proof goes by induction on the maximal number of nondeterministic states that appear on a path of type $t \stackrel{a}{\Rightarrow} [s']_R$, not counting the last state. More precisely, for a

given path c, we define $\mathrm{Nstates}(c) = \{r \mid r \in S_n,\ r \text{ appears in } c,\ r \neq \mathrm{last}(c)\}$, and for $x \in S$ such that $(x,s) \in R$, we define $\max_{\mathrm{n}}(x) = \max\{|\mathrm{Nstates}(c)|, \text{where } c \text{ is of type } x \stackrel{a}{\Rightarrow} [s']_R\}$. The proof is by induction on $\max_{\mathrm{n}}(t)$.

Suppose $\max_{\mathrm{n}}(t) = 0$, inducing that $t \in S_p$. From $(s,t) \in R$ it follows that $P(t,[s']_R) > 0$ and $a = \tau$. As s is a nondeterministic state $P(s,[s]_R) = 1$. Thus, from $(s,t) \in R$ is follows that also $P(t,[t]_R) = 1$. Since $P(t,[s']_R) > 0$, we obtain $[s']_R = [t]_R$. Then, the required scheduler is defined by $\sigma(c) = \bot$ for every path c.

Suppose now that $\max_{\mathrm{n}}(t) = m > 0$. We distinguish the following two cases.

(i) $t \in S_n$. Then either $t \stackrel{a}{\rightarrow} t'$, for some $t' \in S$ such that $(s',t') \in R$, or there exists $t'' \in S$ such that $(t'',t) \in R$, $t \stackrel{\tau}{\rightarrow} t''$, and $\max_{\mathrm{n}}(t'') < m$. In the first case, the required scheduler is any scheduler σ that satisfies $\sigma(c) = t \stackrel{a}{\rightarrow} t'$ when $\mathrm{last}(c) = t$. In the second case, by the inductive assumption, there exists a scheduler ρ, such that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\rho)_{/t'' \stackrel{a}{\Rightarrow} [s']_R}\right) = 1.$$

The required scheduler is now defined by

$$\sigma(c) = \begin{cases} t \stackrel{\tau}{\rightarrow} t'', & \text{if } \mathrm{last}(c) = t \\ \rho(c), & \text{otherwise.} \end{cases}$$

(ii) $t \in S_p$. Since $(s,t) \in R$ and $P(s,[s]_R) = 1$ we have $P(t,[t]_R) = 1$. Let $U = \{u \mid t \rightsquigarrow u\}$ be the set of all states reachable from t in one probabilistic transition. $(u,t) \in R$ for every $u \in U$. Thus, for every $u \in U$, there exists $u' \in S$ such that either $u \stackrel{\tau}{\rightarrow} u'$, $(u,u') \in R$, and $\max_{\mathrm{n}}(u') < m$, or $u \stackrel{a}{\rightarrow} u'$ and $(u',s') \in R$. The rest follows easily by the inductive assumption.

□

Example 3.4.3. Consider state u in Figure 3.1, which is branching bisimilar to state s. There exists a scheduler σ, such that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/u \stackrel{a}{\Rightarrow} [t]_{\sim_b}}\right) = 1,$$

where t is the deadlock state. This scheduler, in particular, always chooses action a between a and τ, and chooses action τ between b and τ in the states where there is nondeterminism.

Theorem 3.4.4. *An equivalence relation $R \subseteq S \times S$ is a* branching bisimulation *iff for every $(s,t) \in R$ the following two conditions hold:*

(i) if $s \xrightarrow{a} s'$ for $a \in \mathcal{A}_\tau$, then there exists a scheduler σ such that

$$\mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \overset{a}{\Rightarrow} [s']_R}\right) = 1;$$

(ii) for all $D \in S_{/R}$, $P\left(s, D\right) = P\left(t, D\right)$.

Proof. One direction follows immediately from Lemma 3.4.2, while the other direction is trivial. □

The following lemma, in addition to Lemma 3.4.2, is necessary to establish that $\sim_b$ is finer than $\leftrightarrows_b$. It expresses that the second condition of Def. 3.2.1 can be stated in the form of the second condition of Def. 3.4.1.

Lemma 3.4.5. *Let $R \subseteq S \times S$ be a branching bisimulation. Let $(s,t) \in R$ for $s \in S_p$ and $t \in S$. Let σ be a scheduler such that $\sigma\left(c\right) = \bot$ for all paths c. Then, for all $D \in S_{/R}$ such that $D \neq [s]_R$,*

$$\mu_R\left(s, D\right) = \mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \overset{\tau}{\Rightarrow} D}\right).$$

Proof. Let $D \neq [s]_R$. If $P\left(s, [s]_R\right) = 0$, since $(s,t) \in R$, it follows that $t \in S_p$, from which we have that

$$\mu_R\left(s, D\right) = P\left(s, D\right) = P\left(t, D\right) = \mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \overset{\tau}{\Rightarrow} D}\right).$$

If $P\left(s, [s]_R\right) > 0$, then, since every probabilistic transition leads to a nondeterministic state, there exists $s' \in S_n$, such that $P\left(s, s'\right) > 0$ and $(s, s') \in R$. Since $P\left(s', s'\right) = 1$, we have $P\left(s, [s]_R\right) = 1$. From this, $\mu_R\left(s, D\right) = 0$ and $P\left(t, D\right) = P\left(s, D\right) = 0$. Then

$$\mathrm{Paths_m}\left(\sigma\right)_{/t \overset{\tau}{\Rightarrow} D}$$

is the empty set, from which it follows that $\mathrm{Prob}\left(\mathrm{Paths_m}\left(\sigma\right)_{/t \overset{\tau}{\Rightarrow} D}\right) = 0$. □

Theorem 3.4.6. $\sim_b \subset \leftrightarrows_b$.

Proof. $\sim_b \subseteq \leftrightarrows_b$ from Lemmas 3.4.2 and 3.4.5. The strict inclusion follows directly from the example in Figure 2.1a, where $s \leftrightarrows_b t$, whereas $s \not\sim_b t$. □

3.4.3 The coarsest congruence proof

The following lemma is crucial for the coarsest congruence proof. It says that the states that, when put in a fresh context, yield states equivalent by $\underline{\leftrightarrow}_b$, are exactly those that are related by $\sim_b$.

Lemma 3.4.7. *Let $s, t \in S$. Let x be a fresh action label, that does not appear in any path in the PTS. Let x be a new state such that $x \xrightarrow{x} \mathsf{d}$ (d is the deadlock state) and x has no other outgoing transitions. If $s \parallel x \mathrel{\underline{\leftrightarrow}_b} t \parallel x$, then $s \sim_b t$.*

Proof. Define the equivalence relation $R \subseteq S \times S$ by

$$R = \{(p, q) \mid p \parallel x \mathrel{\underline{\leftrightarrow}_b} q \parallel x\}.$$

We prove that R is a branching bisimulation. Let $(s, t) \in R$.

(i) Suppose that $s \xrightarrow{a} s'$ for some $s' \in S$ and $a \in \mathcal{A}$. Then $s \parallel x \xrightarrow{a} s' \parallel x$. Since $s \parallel x \mathrel{\underline{\leftrightarrow}_b} t \parallel x$, there exists a scheduler σ such that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/t \parallel x \stackrel{a}{\Rightarrow} [s' \parallel x] \underline{\leftrightarrow}_b}\right) = 1.$$

Now, by induction on the maximal number of transitions in a path in the last path set, it easily follows that there exists a path of type $t \stackrel{a}{\Rightarrow} [s']_R$ as required.

(ii) Suppose now that $s \in S_p$. We show that for all $D \in S_{/R}$, $P(s, D) = P(t, D)$. We distinguish the following two cases.

(1) $t \in S_n$. Since $P(t, [t]_R) = 1$, it suffices to show that $P(s, [s]_R) = 1$. We have $t \parallel x \xrightarrow{x} t \parallel \mathsf{d}$. From $s \parallel x \mathrel{\underline{\leftrightarrow}_b} t \parallel x$ it follows that there exists a scheduler σ such that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/s \parallel x \stackrel{x}{\Rightarrow} [t \parallel \mathsf{d}] \underline{\leftrightarrow}_b}\right) = 1.$$

Assume that $P(s, [s]_R) \neq 1$, i.e. that there exist $s' \in S$ and $\pi \in (0, 1]$ such that $P(s, s') = \pi$ and $s \parallel x \not\mathrel{\underline{\leftrightarrow}_b} s' \parallel x$. Then $P(s \parallel x, s' \parallel x) = \pi$. Therefore,

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/s \parallel x \stackrel{x}{\Rightarrow} [t \parallel \mathsf{d}] \underline{\leftrightarrow}_b}\right) \leq 1 - \pi$$

for every scheduler σ, which is not possible. We conclude that $P(s, [s]_R) = 1$.

(2) $t \in S_p$. The following two subcases are possible.

(2.1) $P(s, [s]_R) > 0$. Then there exists $s' \in S_n$ such that $P(s, s') > 0$ and $s \parallel x \leftrightarrows_b s' \parallel x$. As s and t are related to a nondeterministic state, similarly to the previous case, it follows that $P(s, [s]_R) = P(t, [s]_R) = 1$.

(2.2) $P(s, [s]_R) = 0$. Assume $P(t, [s]_R) > 0$. Then

$$P(t \parallel x, [s \parallel x]_{\underline{\leftrightarrows}_b}) > 0.$$

This means that there is a $t' \in S$, $t' \neq t$, such that $P(t \parallel x, t' \parallel x) > 0$ and $t \parallel x \leftrightarrows_b t' \parallel x$. Since $t' \parallel x \in S_n$, it follows that $t' \parallel x \xrightarrow{x} t' \parallel \mathsf{d}$. From the last, and from $s \parallel x \leftrightarrows_b t' \parallel x$, it follows that there is a scheduler σ such that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/s \parallel x \stackrel{x}{\Rightarrow} [t' \parallel \mathsf{d}]_{\underline{\leftrightarrows}_b}}\right) = 1.$$

But this implies that there exists $s' \in S$, $s' \neq s$, such that $P(s \parallel x, s' \parallel x) > 0$ and $s \parallel x \leftrightarrows_b s' \parallel x$. From this it follows that $P(s', s) > 0$ and $(s, s') \in R$. This contradicts the assumption that $P(s, [s]_R) = 0$. We conclude that $P(t, [s]_R) = 0$.

Now, let $D \in S_{/R}$ be such that $D \neq [s]_R$. Then $\mu_R(s, D) = P(s, D)$. It easily follows that there exists a scheduler σ such that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/t \stackrel{\tau}{\Rightarrow} D}\right) = P(s, D).$$

Since $P(t, [t]_R) = 0$, it must hold that

$$\mathrm{Prob}\left(\mathrm{Paths}_{\mathrm{m}}(\sigma)_{/t \stackrel{\tau}{\Rightarrow} D}\right) = P(t, D),$$

i.e. $P(s, D) = P(t, D)$.

□

We now prove that $\sim_b$ is the largest equivalence included in $\leftrightarrows_b$ which is compatible with the parallel composition operator.

Theorem 3.4.8. *Let $R \subseteq S \times S$ be an equivalence relation that is congruence with respect to the parallel composition operator. If $R \subseteq \leftrightarrows_b$, then $R \subseteq \sim_b$.*

Proof. Suppose $(s, t) \in R$. Let x be as in Lemma 3.4.7. Then $(s \parallel x, t \parallel x) \in R$ because R is a congruence relation. Since $R \subseteq \leftrightarrows_b$, we have $s \parallel x \leftrightarrows_b t \parallel x$. From Lemma 3.4.7, it follows that $s \sim_b t$. □

Chapter 4

Branching bisimilarity: Algorithm, logics, axioms

We continue studying branching bisimilarity defined in the previous chapter, to facilitate its use as a reduction technique. Namely, an algorithm of complexity $O(n^4)$ w.r.t. the number of states for partitioning the state space according to branching bisimilarity is presented. It is also shown that branching bisimilarity preserves the properties expressible in probabilistic Computation Tree Logic. To support usage of the equivalence in a process algebraic setting, a complete axiomatization for finite processes is given, such that the process language contains a rich set of operators needed to reason on concurrent probabilistic processes. As an intermediate result, an alternative definition of branching bisimilarity, based on coloured traces, is given.

4.1 Decidability algorithm

In this section we present an algorithm for partitioning the state space of a PTS according to branching bisimilarity ($\sim_b$) with time complexity $O(n^4)$ on the number of states.

As the definition of $\sim_b$ does not need the notion of a scheduler, we take an approach that is rather different than those taken in [91] and [8] for the same purpose. In fact, observe that the first condition of Def. 3.2.1, i.e. the definition of branching bisimulation, coincides with the condition of the non-probabilistic branching bisimulation [62], if a PTS is turned into an ordinary labeled transition system by replacing the probabilistic transitions with internal τ-transitions. Therefore, we can use an algorithm for non-

```
Π:= {{S}};
Π_f:= ∅;
while Π ≠ Π_f do
     Π_f := GV_standard_bb(Π);
     Π := Π_f;
     if exists (B, B') = FindP_Split(Π_f)
            then Π:= Refine (Π_f, B, B' )
od
return Π
```

Figure 4.1: Partitioning the state space according to $\sim_b$.

probabilistic branching bisimilarity, denoted by $\simeq$, as a step in our algorithm. We use the algorithm defined in [64], to which we refer to as `GV-standard-bb`. For a given partition Π of the set of states S of a labeled transition system, the algorithm `GV-standard-bb` refines Π to the maximal non-probabilistic branching bisimulation equivalence contained in Π. However, due to the second condition of Def. 3.2.1, we have to refine the output partitions of `GV-standard-bb`, by means of a *P-splitter*.

Definition 4.1.1. Given a PTS, let Π be a partition of the set of states S and let $B, B' \in \Pi$, such that $B \neq B'$.

- B' is a *P-splitter* of B iff there exist $s, t \in B$ for which $P(s, B') \neq P(t, B')$.
- Let B' be a P-splitter of B.
  - $\mathrm{Ref}_\Pi(B, B') = \{B_1, \ldots, B_k\}$ $(k \geq 2)$ is a minimal partition of B such that, for all $i, 1 \leq i \leq k$, B' is not a P-splitter of B_i.
  - $\mathrm{Refine}(\Pi, B, B') = \Pi \setminus \{B\} \cup \mathrm{Ref}_\Pi(B, B')$.

The algorithm, which assumes as an input a PTS and returns the quotient set of its states by $\sim_b$, is given in Fig. 4.1. The following lemma justifies the algorithm.

Lemma 4.1.2. *For an input PTS with a set of states S, the algorithm in Fig. 4.1 outputs $S_{/\sim_b}$.*

Proof. Let Π be the partitioning of S returned by the algorithm in Fig. 4.1. Note that the loop in the algorithm ends when $\Pi = \Pi\_f$, meaning that for

every class B, every pair of states (s, t) in B, and every class B', $P(s, B') = P(t, B')$. Thus, the second condition of Def. 3.2.1 is satisfied by Π. Also, Π is a non-probabilistic branching bisimulation, meaning that the first condition of Def. 3.2.1 is satisfied by Π. Thus, Π defines a branching bisimulation. We need to show that $\Pi = S_{/\sim_b}$.

Let $C \in S_{/\sim_b}$. We show that, after the n-th update of Π, a block D exists from the current partitioning Π_n such that $C \subseteq D$. The proof is by induction on n.

The case $n = 1$ is trivial, since $\Pi_1 = \{\{S\}\}$.

Suppose $n > 1$.

Suppose first that Π_n is obtained from Π_{n-1} via the first step in the while-loop, i.e. via `GV-standard-bb`. Then the proof follows from the fact that `GV-standard-bb` yields the coarsest refinement of Π_{n-1} that does not break the first condition of Def. 3.2.1.

Suppose now that Π_n is obtained from Π_{n-1} via the second step in the while-loop , i.e. by refining $B \in \Pi_{n-1}$ further into $\{B_1 \dots B_k\}$ by the P-splitter B'. If $C \not\subseteq B$ then the proof follows from the inductive assumption. Therefore, suppose $C \subseteq B$. We need to show that there exists $B_i \in \{B_1 \dots B_k\}$ such that $C \subseteq B_i$. Without loss of generality, assume that there exist states $s, t \in C$ such that $s \in B_1$ and $t \in B_2$. Then, $P(s, B') \neq P(t, B')$. By the inductive assumption, there exist $B'_1, \dots B'_m \in S_{/\sim_b}$ such that $B' = B'_1 \cup \dots \cup B'_m$. From this and from $P(s, B') \neq P(t, B')$ it follows that there exists $B'_j \in \{B'_1, \dots B'_m\}$ such that $P\left(s, B'_j\right) \neq P\left(t, B'_j\right)$, which contradicts the fact that $s \sim_b t$ and $B'_j \in S_{/\sim_b}$. □

We calculate the time complexity of the algorithm in Fig. 4.1. Let $n = |S|$, i.e. n is the number of states. The `while` loop can be executed at most n times. `GV-standard-bb` has a worst-case time complexity $O(n^3)$ [64], when the action set is finite. (More concretely, the time complexity of `GV-standard-bb` is $O(nm)$, where m is the number of transitions.) The procedure `FindP_Split`, which finds a pair (B, B') with B' being a P-splitter of B, boils down to comparing elements (rows) in the matrix representation of P. It is easy to show that this procedure can be performed in time $O(n^3)$. Thus, the following result holds.

Theorem 4.1.3. *$\sim_b$ is decidable in time $O(n^4)$, where n is the number of states in the PTS.*

4.2 Colouring definition

In [62] an alternative definition of non-probabilistic branching bisimilarity, exploiting colouring of the states, has been given, which is easier to grasp and reveals how branching bisimilarity indeed preserves the branching structure. It is based on a comparison of the *coloured trace* sets that the states generate, where a coloured trace is an alternating sequence of colours of the states and actions. More precisely, two states are branching bisimilar (by [62]) if and only if they have the same colour under some *consistent colouring*, where by a consistent colouring the same colour has been given only to states that have the same sets of coloured traces. Here, we show that a similar result holds in the probabilistic case.[1] However, for a state in a PTS a blend of colours (rather than a single colour), representing the probability distribution of the subsequent colours, is needed (see also [8] for use of blends).

Definition 4.2.1 (Colours, blends, coloured PTS)**.** Let C be a finite, sufficiently large set of colours. A *blend* is a function $\mathcal{V}\colon \mathsf{C} \mapsto [0,1]$ such that $\sum_{\mathsf{c}\in\mathsf{C}} \mathcal{V}(\mathsf{c}) = 1$. If there exists a colour $\mathsf{c} \in \mathsf{C}$ such that $\mathcal{V}(\mathsf{c}) = 1$, then $\mathcal{V}$ is called a *pure colour*. A PTS is *coloured* if a function, which assigns a blend to each state of the PTS, is associated to it.

Definition 4.2.2 (Coloured traces)**.** A *concrete coloured trace* is a (possibly infinite) path in the PTS, in which the probability labels have been erased and the states have been substituted by their blends. A *coloured trace* is a concrete coloured trace in which *every* subsequence of equal blends (possibly separated by τ-labels) has been replaced by the blend itself.

Note that in a coloured trace there do not appear consecutive equal blends separated by τ, i.e. it is not possible to observe τ-transitions between equal blends. The probability labels are also irrelevant since all the necessary probability information is contained in the blends.

Definition 4.2.3 (Consistent colouring)**.** A colouring of the states of a PTS is *consistent* iff two states have the same blend only if they have the same sets of coloured traces.

Example 4.2.4. The colouring of the states of the PTS given in Figure 3.1 is consistent. The states coloured with "●" have the coloured traces

$$\{\ \bullet\ ,\ \bullet\, a\, \circ\ ,\ \bullet\, b\, \circ\ \}\ ,$$

[1] The colouring definition, besides being interesting on its own, also eases the proofs in sections 4.3 and 4.4.

while the states coloured with "●" have the coloured traces

$$\{\ \bullet\ ,\ \bullet\ a\ \circ\ \}\ .$$

The notion of a proper colouring of a PTS defined below is characteristic for our branching bisimilarity. A colouring of a probabilistic state is considered proper only if the distribution of colours in a probabilistic state corresponds to the probability distribution of the subsequent colours. As already established earlier, a nondeterministic state cannot be related to a probabilistic state with a nontrivial probability distribution. Consequently, a nondeterministic state in a proper colouring always has a pure colour.

Definition 4.2.5 (Proper colouring)**.** A PTS with a colouring function χ is *properly coloured* iff for every nondeterministic state s, $\chi(s)$ is a pure colour, and for every probabilistic state s, $\chi(s) = \sum_{s_i : s \rightsquigarrow s_i} P(s, s_i)\ \ \chi(s_i)$.

Example 4.2.6. The colouring of the states of the PTS given in Figure 3.1 is proper: each probabilistic state has equally coloured subsequent states and is coloured the same as them.

For states s and t let us write $s \equiv_c t$ if for some consistent, proper colouring, s and t have the same blend. The following theorem gives the colouring definition of $\sim_b$.

Theorem 4.2.7 (Colouring definition of $\sim_b$)**.** *Given states s and t, $s \sim_b t$ iff $s \equiv_c t$.*

Proof. The proof builds on the corresponding proof in [62]. Namely, for the direction left-to-right, let χ be a colouring of the states of the PTS such that two nondeterministic states have the same colour if and only if they are branching bisimilar, and each probabilistic state has a blend that represents the probability distribution over the colours of the subsequent nondeterministic states. Clearly, two probabilistic states have the same colour if and only if they are branching bisimilar. We need to show that the colouring χ is consistent. Let $s \sim_b t$ and $\mathcal{V}_0\ \mathsf{c}_0\ a_1\ \mathcal{V}_1\ \mathsf{c}_1\ a_2\ \ldots\ a_k\ \mathcal{V}_k\ \mathsf{c}_k$ be a coloured trace originated from s, where each $\mathcal{V}_i$ is a real blend, i.e. it is not a pure colour, and may not exist in the coloured trace, and each c_i is a pure colour. We show that t also has a coloured trace $\mathcal{V}_0\ \mathsf{c}_0\ a_1\ \mathcal{V}_1\ \mathsf{c}_1\ a_2\ \ldots\ a_k\ \mathcal{V}_k\ \mathsf{c}_k$, such that $\mathcal{V}_i$ appears iff it appears in the coloured trace of s. We first show for the case $k = 1$. Suppose s has a coloured trace $\mathcal{V}_0\ \mathsf{c}_0\ a_1\ \mathcal{V}_1\ \mathsf{c}_1$. Assume that $\mathcal{V}_0$ and $\mathcal{V}_1$ appear in the trace, since this is the most interesting case. Then $s \in S_p$ and therefore, since $\mathcal{V}_0$ is not a pure colour, $t \in S_p$, too. Since χ is a proper colouring and $s \sim_b t$, t has a coloured trace starting with $\mathcal{V}_0\ \mathsf{c}_0$. Since

s has a coloured trace $\mathcal{V}_0\ \mathsf{c}_0\ a_1\ \mathcal{V}_1\ \mathsf{c}_1$, there exists a sequence of transitions $s \rightsquigarrow s_1 \dashrightarrow s_2 \dashrightarrow \ldots s_n \xrightarrow{a_1} s'$, such that $\chi(s_i) = \mathsf{c}_0$ (i.e. $s_i \sim_b s_1$) for each $1 \leq i \leq n$ and $\chi(s') = \mathcal{V}_1$. From $s \sim_b t$, by induction on n it is not hard to show that there exists a sequence of transitions $t \rightsquigarrow t_1 \dashrightarrow t_2 \dashrightarrow \ldots t_m \xrightarrow{a_1} t'$ such that $\chi(t_i) = \mathsf{c}_0$ for each $1 \leq i \leq m$ and $\chi(t') = \mathcal{V}_1$. Therefore, there exists a coloured trace $\mathcal{V}_0\ \mathsf{c}_0\ a_1\ \mathcal{V}_1\ \mathsf{c}_1$ originating from t. By applying the same discussion k times, we can prove that if s has a coloured trace $\mathcal{V}_0\ \mathsf{c}_0\ a_1\ \mathcal{V}_1\ \mathsf{c}_1\ a_2\ \ldots\ a_k\ \mathcal{V}_k\ \mathsf{c}_k$ then t has the same trace, too. Thus, colouring χ is consistent.

For the direction right-to-left, let χ be a proper and consistent colouring of the PTS. Let R be a relation that relates two states iff they have the same colour by χ. It is easy to show that R is a branching bisimulation. □

Corollary 4.2.8. *Given a PTS, there exists a proper, consistent colouring of its states, called* canonical colouring*, such that two states have the same blend if and only if they are branching bisimilar.*

4.3 Branching bisimilarity and pCTL

We show that branching bisimilar states satisfy the same pCTL formulas. First we present our variant of pCTL, and then we show the soundness property.

4.3.1 pCTL

We present the logic that we use to express properties of probabilistic systems. The logic we consider is a variant of the probabilistic CTL logic defined in [23] and a simplification of the probabilistic logic of [16], both defined on Kripke-like structures. However, we need to interpret pCTL on a PTS, taking into account that τ transitions have a special treatment. To this end, we follow the approach in [40], where it is shown that CTL is in full agreement with the non-probabilistic branching bisimulation of [62] by extending transition systems to doubly-labeled Kripke structures. In the latter, relations defined on transition systems and on Kripke structures can be easily compared. We thus interpret the logic over a similar extension of a PTS, called an *EPTS* (Extended PTS). In fact, an EPTS is a PTS in which the states also have labels.

Definition 4.3.1 (Translating PTS to Extended PTS)**.** Let Φ be a PTS. EPTS(Φ), with a state labeling function $\mathcal{L}$, is constructed from Φ in the following way:

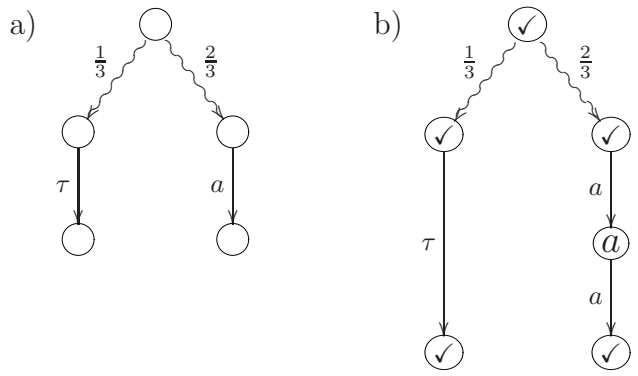

Figure 4.2: A PTS (a) and its extended version (b)

(i) every transition $s \xrightarrow{a} t$ where $a \in \mathcal{A}$ is erased, and instead a state (s, a, t) and the transitions $s \xrightarrow{a} (s, a, t)$ and $(s, a, t) \xrightarrow{a} t$ are created;

(ii) for every new state (s, a, t), $\mathcal{L}((s, a, t)) = a$, and for every old state s, $\mathcal{L}(s) = \checkmark$, where $\checkmark$ is a new constant such that $\checkmark \notin \mathcal{A}$.

Example 4.3.2. In Figure 4.2-a) a PTS is given, together with its extended version in Figure 4.2-b). In the extended version, every observable action transition is split into two and a label denoting the action itself is assigned to the newly created state; the old states are labeled by $\checkmark$. Note that the probabilistic transitions and the τ-transitions remain unchanged.

Definition 4.3.3. The syntax of pCTL is generated by the following grammar:

$$\psi := \ \checkmark \mid a \mid \neg\psi \mid \psi \wedge \psi' \mid \exists P_{\bowtie p}(\psi U \psi')$$

where $\bowtie \in \{<, >, \leq, \geq\}$, $p \in [0, 1]$, $a \in \mathcal{A}$.

Note that we have not included the "next" operator [16], since we consider branching bisimulation.

In the following definition we assume that a scheduler does not stop an execution if there is a step that can be executed. In other words, a scheduler can stop an execution only in a deadlock state. This is to stay inline with the generally accepted interpretation of (non-probabilistic) CTL formulas, based on *maximal* paths only, i.e. paths that are either infinite, or end with a deadlock state [40].

Definition 4.3.4. [pCTL Semantics] For a given EPTS with a labeling function $\mathcal{L}$, satisfaction of a formula ψ in a state s, $s \vDash \psi$, is defined inductively:

(i) $s \vDash \checkmark$ iff $\mathcal{L}(s) = \checkmark$;

(ii) $s \vDash a$ iff $\mathcal{L}(s) = a$;

(iii) $s \vDash \neg\psi$ iff $s \nvDash \psi$;

(iv) $s \vDash \psi \wedge \psi'$ iff $s \vDash \psi$ and $s \vDash \psi'$;

(v) $s \vDash \exists P_{\bowtie p}(\psi U \psi')$ iff there exists a scheduler σ such that the probability measure of the set of all paths scheduled by σ that start in s and satisfy formula $\psi U \psi'$ is $\bowtie p$, where a path $c = s_0 l_1 s_1 \ldots$ satisfies formula $\psi U \psi'$, denoted by $c \vDash \psi U \psi'$, iff there exists $n \geq 0$ such that $s_n \vDash \psi'$ and, for all $i < n$, $s_i \vDash \psi$.

4.3.2 Soundness of branching bisimilarity for pCTL

We proceed with proving that branching bisimilar states in an EPTS satisfy the same pCTL formulas. It is easy to show that two states in a PTS Φ are branching bisimilar if and only if they are branching bisimilar in EPTS(Φ) (see [40] for the proof outline). This justifies our approach.

We need to extend the definition of branching bisimilarity for paths.

Definition 4.3.5. Paths c_1 and c_2 are *branching bisimilar* iff, given the canonical colouring of the PTS they belong to, they induce equal coloured traces.

Lemma 4.3.6. *Let s and t be states such that $s \sim_b t$. Let σ be a scheduler that generates the set of paths* $\mathrm{Paths}(\sigma, s)$ *starting at s. There exists a scheduler σ' that generates a set of paths* $\mathrm{Paths}(\sigma', t)$ *starting at t, such that*

(i) the set of coloured traces defined by $\mathrm{Paths}(\sigma, s)$ *coincides with the set of coloured traces defined by* $\mathrm{Paths}(\sigma', t)$*, and*

(ii) for every finite path c, the probability measure of all paths in $\mathrm{Paths}(\sigma, s)$ *that are branching bisimilar to c is equal to the probability measure of all paths in* $\mathrm{Paths}(\sigma', t)$ *that are branching bisimilar to c.*

Proof. The first part follows from Theorem 4.2.7. For the second part, we calculate the probability measure of the set of σ-scheduled paths that are branching bisimilar to a given finite path c, by considering the blends in the coloured trace generated by c. Let $\mathcal{V}_1, \mathcal{V}_2, \ldots \mathcal{V}_k$ be the non-pure-colour blends that appear in the coloured trace of c (they all come from probabilistic

states). Let $\gamma_1, \gamma_2, \ldots \gamma_k$ be the (pure colour) blends that follow $\mathcal{V}_1, \mathcal{V}_2, \ldots \mathcal{V}_k$ in the coloured trace generated by c, resp. Let $\mathsf{c}_1, \mathsf{c}_2, \ldots \mathsf{c}_k$ be the colours of $\gamma_1, \gamma_2, \ldots \gamma_k$, resp. Then, the probability measure of the set of σ-scheduled paths that are branching bisimilar to c is equal to

$$\begin{cases} 1, & \text{if } \{\mathcal{V}_i\}_{i\in\{1,\ldots k\}} = \emptyset \\ \prod_{i=1}^{k} \mathcal{V}_i\left(\mathsf{c}_i\right), & \text{otherwise.} \end{cases}$$

□

Theorem 4.3.7. *Branching bisimilar states in an EPTS satisfy the same pCTL formulas.*

Proof. Let $s \sim_b t$. For the purpose of the proof, we extend the syntax of pCTL with path formulas $\psi U \psi'$, with semantics given in Def. 4.3.4. The proof is by induction on the structure of the formula. The nontrivial step is for formulas of type $\exists P_{\bowtie p}(\psi U \psi')$. By the semantics of the path formulas $\psi U \psi'$ and by the inductive assumption, two branching bisimilar paths satisfy the same formulas of type $\psi U \psi'$. Let σ be a scheduler that starts in s and generates the set of paths $\mathrm{Paths}\,(\sigma, s)$. By Lemma 4.3.6, there exists a scheduler σ' that generates a set of paths $\mathrm{Paths}\,(\sigma', t)$ starting from t, such that the set of coloured traces defined by the paths of $\mathrm{Paths}\,(\sigma, s)$ coincides with the one defined by the paths of $\mathrm{Paths}\,(\sigma', t)$. Let $\psi U \psi'$ be a path formula. Let $\mathrm{Paths}\,(\sigma, s)_{\psi U \psi'}$ be the subset of (finite) paths in $\mathrm{Paths}\,(\sigma, s)$ that satisfy formula $\psi U \psi'$. We partition the set $\mathrm{Paths}\,(\sigma, s)_{\psi U \psi'}$ into classes of branching bisimilar paths. By Lemma 4.3.6, and because branching bisimilar paths satisfy the same formulas of type $\psi U \psi'$, every class of $\mathrm{Paths}\,(\sigma, s)_{\psi U \psi'}$ has a correspondent in the analog partitioning of $\mathrm{Paths}\,(\sigma', t)_{\psi U \psi'}$, and their probability measures coincide. This completes the proof. □

Note that the opposite does not hold, i.e., if two states are not branching bisimilar they may still satisfy the same formulas, as for instance, states s and t in Figure 2.1. However, the purpose of the result presented here is to show that branching bisimilarity can be used for reduction of a system prior to model checking w.r.t. a straightforward probabilistic extension of CTL. We leave the problem of logical characterization of branching bisimilarity for the future.

4.4 A complete axiomatization: Process theory pTCP$_\tau$

In this section we define the process theory pTCP$_\tau$, which extends the theory TCP$_\tau$ [11] with a probabilistic choice operator. Besides the latter, it also includes the operators sequential composition, alternative composition, parallel composition, hiding and encapsulation, and the constants 0 (deadlock) and 1 (termination). The semantical equivalence for pTCP$_\tau$ is branching bisimilarity, slightly adapted to become compatible for the rest of the operators.

4.4.1 Process language pTCP$_\tau$

We present the process language pTCP$_\tau$. The underlying semantics of pTCP$_\tau$ is defined by a set of operational rules by means of which each process expression can be interpreted as a process graph. The latter is used to model probabilistic processes and is defined next.

Definition 4.4.1 (Probabilistic process graph)**.** Given a PTS, a *probabilistic process graph* or simply a *process graph* is the connected graph induced by the states reachable from a particular state s, called *the root* of the process graph.

A process graph is usually named by its root. We assume that a certain subset of the nondeterministic states, denoted by $\downarrow$, is a set of *terminating* states, and we write $s\downarrow$ rather than $s \in \downarrow$. For the sake of convenience, in the rest of this section we assume that the process graphs are *root-unwound*, i.e. that they have no incoming transitions at the root. Since each strong bisimulation class by [68] contains a root-unwound process graph, this restriction does not cause any loss of generality.

Let $\gamma\colon \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ be a partial commutative and associative communication function [11]. The syntax of the pTCP$_\tau$ language is defined by the following grammar:

$$\begin{aligned} E ::= 0 \ \mid \ 1 \mid a.E \mid E{+}E \mid E{\cdot}E \mid E \oplus_\pi E \mid \\ E \| E \mid E \between E \mid E|E \mid \partial_H(E) \mid \tau_I(E) \mid x \end{aligned}$$

where $a \in \mathcal{A}_\tau$, $\pi \in (0,1)$, $I, H \subseteq \mathcal{A}$, and $x \in V$, where V is a set of recursive variables. The constants 0 and 1 stand for the deadlock, resp. termination process. $a.E$ is the action prefix, $E{+}E$ stands for alternative composition, $E{\cdot}E$ for sequential composition, $E \oplus_\pi E$ for probabilistic choice, $E\|E$ for

parallel composition, $E \between E$, resp. $E|E$ are the left merge, resp. communication operator, and $\partial_H(E)$, resp. $\tau_I(E)$ are the encapsulation, resp. hiding operator. We assume the following binding strengths: $. > \cdot > \between > \| > +$, i.e. "." binds strongest. Operators $\between$ and $|$ bind equally as $\|$, and $\oplus$ binds equally as $+$. pTCP_τ process expressions that do not contain any variables are called *closed* process expressions. To model infinite processes we allow *guarded recursive specifications* [11]. A *guarded recursive specification* is a finite set of equations of the form $x = p_x(V)$, where $x \in V$ and p is a pTCP_τ expression in which all occurrences of variables from V are prefixed by an action from $\mathcal{A}$ [11]. Moreover, the only operators allowed in p are: 0, 1, action prefix $a.$, for $a \in \mathcal{A}_\tau$, $+$ and $\oplus_\pi$. In this way we restrict ourselves to finitely definable processes, i.e. processes that can be represented by finite-state graphs. We refer the reader to [15,41] for more details.

$$1\!\downarrow \qquad a.x \xrightarrow{a} x \qquad \frac{x \xrightarrow{a} x',\ y\not\rightarrow}{x+y \xrightarrow{a} x',\ y+x \xrightarrow{a} x'} \qquad \frac{x\!\downarrow,\ y\not\rightarrow}{(x+y)\!\downarrow,\ (y+x)\!\downarrow}$$

$$\frac{x \overset{\pi}{\rightsquigarrow} x',\ y\not\rightarrow}{x+y \overset{\pi}{\rightsquigarrow} x'+y,\ y+x \overset{\pi}{\rightsquigarrow} y+x'} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ y \overset{\rho}{\rightsquigarrow} y'}{x+y \overset{\pi\rho}{\rightsquigarrow} x'+y'}$$

$$\frac{x \overset{\rho}{\rightsquigarrow} x'}{x \oplus_\pi y \overset{\pi\rho}{\rightsquigarrow} x',\ y \oplus_\pi x \overset{(1-\pi)\rho}{\rightsquigarrow} x'} \qquad \frac{x\not\rightarrow}{x \oplus_\pi y \overset{\pi}{\rightsquigarrow} x,\ y \oplus_\pi x \overset{1-\pi}{\rightsquigarrow} x}$$

$$\frac{x\!\downarrow, y\!\downarrow}{(x\cdot y)\!\downarrow} \qquad \frac{x \xrightarrow{a} x'}{x\cdot y \xrightarrow{a} x'\cdot y} \qquad \frac{x\!\downarrow,\ y \xrightarrow{a} y'}{x\cdot y \xrightarrow{a} y'}$$

$$\frac{x\!\downarrow,\ y \overset{\pi}{\rightsquigarrow} y'}{x\cdot y \overset{\pi}{\rightsquigarrow} x\cdot y'} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ x'\not\downarrow}{x\cdot y \overset{\pi}{\rightsquigarrow} x'\cdot y} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ y\not\rightarrow}{x\cdot y \overset{\pi}{\rightsquigarrow} x'\cdot y} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ x'\!\downarrow,\ y \overset{\rho}{\rightsquigarrow} y'}{x\cdot y \overset{\pi\rho}{\rightsquigarrow} x'\cdot y'}$$

Table 4.1: Operational semantics for the pTCP_τ expressions: rules for prefix, choice operators and sequential composition

Tables 4.1, 4.2 and 4.3 represent the operational semantics for pTCP_τ. The rules can be applied to pTCP_τ process expressions, and the generated process graph is counted as an interpretation of the process expression. Next, we discuss briefly the semantics of the pTCP_τ operators. The negative premise $x\not\rightarrow$ that appears in several rules denotes that x does not start with a probabilistic transition. The deadlock process 0 cannot execute any activity, while process 1 can only terminate (Table 4.1). Process $a.x$ per-

$$\frac{x \xrightarrow{a} x',\ y \not\rightarrow}{x \parallel y \xrightarrow{a} x' \parallel y,\ y \parallel x \xrightarrow{a} y \parallel x'} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y',\ \gamma(a,b) = c}{x \parallel y \xrightarrow{c} x' \parallel y'}$$

$$\frac{x \overset{\pi}{\rightsquigarrow} x',\ y \not\rightarrow}{x \parallel y \overset{\pi}{\rightsquigarrow} x' \parallel y,\ y \parallel x \overset{\pi}{\rightsquigarrow} y \parallel x'} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ y \overset{\rho}{\rightsquigarrow} y'}{x \parallel y \overset{\pi\rho}{\rightsquigarrow} x' \parallel y'} \qquad \frac{x\!\downarrow,\ y\!\downarrow}{(x \parallel y)\!\downarrow}$$

$$\frac{x\!\downarrow,\ y\!\downarrow}{(x \mid y)\!\downarrow} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y',\ \gamma(a,b) = c}{x \mid y \xrightarrow{c} x' \parallel y'} \qquad \frac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \parallel y}$$

$$\frac{x \overset{\pi}{\rightsquigarrow} x',\ y \not\rightarrow}{x \between y \overset{\pi}{\rightsquigarrow} x' \between y,\ y \between x \overset{\pi}{\rightsquigarrow} y \between x'} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ y \overset{\rho}{\rightsquigarrow} y'}{x \between y \overset{\pi\rho}{\rightsquigarrow} x' \between y'}$$

$$\frac{x \overset{\pi}{\rightsquigarrow} x',\ y \not\rightarrow}{x \mid y \overset{\pi}{\rightsquigarrow} x' \mid y,\ y \mid x \overset{\pi}{\rightsquigarrow} y \mid x'} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x',\ y \overset{\rho}{\rightsquigarrow} y'}{x \mid y \overset{\pi\rho}{\rightsquigarrow} x' \mid y'}$$

Table 4.2: Operational semantics for the pTCP_τ expressions: rules for merge and communication operators

forms action a and proceeds as process x. The probability distribution that yields probability transitions of process $x \oplus_\pi y$ is a linear combination of the distributions of x and y. If one of the operands, say x, is a nondeterministic process, then, as expressed by the second rule for the probabilistic choice operator (Table 4.1), process $x \oplus_\pi y$ behaves as x with probability π. The probability distribution of process $x{+}y$ is obtained by joining the distributions of x and y, namely, it is a product of their individual distributions (see also [3]). The intuition behind this is that, since process $x{+}y$ is either x or y, then if process x behaves as process x' with probability π and

$$\frac{x \xrightarrow{a} x',\ a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x'}{\partial_H(x) \overset{\pi}{\rightsquigarrow} \partial_H(x')} \qquad \frac{x\!\downarrow}{\partial_H(x)\!\downarrow}$$

$$\frac{x \xrightarrow{a} x',\ a \notin I}{\tau_I(x) \xrightarrow{a} \tau_I(x')} \qquad \frac{x \xrightarrow{a} x',\ a \in I}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')} \qquad \frac{x \overset{\pi}{\rightsquigarrow} x'}{\tau_I(x) \overset{\pi}{\rightsquigarrow} \tau_I(x')} \qquad \frac{x\!\downarrow}{\tau_I(x)\!\downarrow}$$

Table 4.3: Operational semantics for the pTCP_τ expressions: rules for encapsulation and hiding

process y behaves as process y' with probability ρ, then $x+y$ would behave as either x' or y' with probability $\pi\rho$. A similar approach is taken for the sequential composition $x \cdot y$. Namely, if process x terminates and process y behaves as process y' with probability π, then, as the bottom-left-most rule in Table 4.1 expresses, process $x \cdot y$ behaves as process $x \cdot y'$ with probability π. The bottom-right-most rule in Table 4.1 expresses the same idea for a more complicated case, when the terminating process is preceded by a probabilistic transition. This rule imposes combining consecutive probabilistic transitions. This definition of sequential composition, for a process language containing the constant 1, is introduced here for the first time and extends the definition of [11] to the probabilistic setting.

The probability distribution of the parallel composition of x and y, $x \parallel y$, is a product of the individual distributions of x and y (Table 4.2) (see also [68]). Namely, if process x behaves as process x' with probability π and y behaves as y' with probability ρ, then the parallel composition, $x \parallel y$, behaves as $x' \parallel y'$ with probability $\pi\rho$; process $x' \parallel y'$ is a nondeterministic process that can perform an action from x' or y', or can perform an action that is a result of communication. The operators $\between$ and $|$ are the standard left merge and communication operator [11]. $x \between y$ is the parallel composition of x and y, with the restriction that the first action comes from x. In $x \mid y$ processes x and y are forced to communicate first and then to proceed in parallel. The semantics of $\between$ and $|$ has been extended for our model in the same lines as for parallel composition. Note that these operators are, however, not needed to define the operational semantics of parallel composition; rather they are needed later in order to obtain a finite axiomatization of parallel composition.

For process $\partial_H(x)$ the initial probability distribution is inherited from process x (Table 4.3), and similarly for $\tau_I(x)$. Applied on a nondeterministic process, the encapsulation and hiding operators work as usual: $\partial_H(x)$ blocks execution of any action from set H, while $\tau_I(x)$ renames all actions from set I that x can perform into τ.

Remark Tables 4.1, 4.2, and 4.3 contain rules with negative premises, e.g. $y \not\rightarrow$. Not always specifications containing such rules are well-defined. For example, from the rule

$$\frac{x \overset{a}{\not\rightarrow}}{x \overset{a}{\rightarrow} x'}$$

it cannot be concluded whether a transition $x \overset{a}{\rightarrow} x'$ exists or not. These issues have been studied in e.g. [24, 59, 65], and several techniques have been proposed there to check whether a set of operational semantics rules is well-defined. In [65] it has been shown that a sufficient condition for a set of

operational rules to be well-defined is for it to be *stratifiable*. Stratification ensures that no transition depends negatively on itself. It can be checked by this method that the operational semantics of pTCP_τ is well-defined; the details of the proof are, however, beyond the scope of the current text.

4.4.2 Branching bisimilarity and pTCP_τ operators

We present a congruence result for branching bisimilarity with respect to the operators in pTCP_τ . However, as we have included termination property of the states, we need to adapt branching bisimilarity appropriately. Namely, in order to achieve congruence for sequential composition, the states need to be able to simulate each other's termination property. While in the non-probabilistic setting this condition suffices [62], in the probabilistic setting it has to be strengthened. First, note that process $\tau.1$, that can perform only a τ-action that leads to a terminating state, cannot be related to 1. Otherwise, when $\tau.1$ and 1 are each composed with a non-trivial probabilistic process, say $a.0 \oplus_{0.5} b.0$ sequentially, the resulting composed processes are not branching bisimilar and thus the congruence property for sequential composition is compromised. Thus, in the terms of PTS, a terminating state cannot be related to a non-terminating state, even if the latter has a τ-transition to a terminating state. Second, note that even process $1 + \tau.1$ cannot be related to 1, for the same reasons as above. Thus, we must add a requirement saying that when two terminating states are related, they must be able to simulate each other's τ-transitions.

We incorporate the previous discussion in the following definition.

Definition 4.4.2 (Termination-sensitive branching bisimulation)**.** Let R be a branching bisimulation defined on the set of states of a PTS. R is *termination-sensitive* if, whenever $(s,t) \in R$ and $s{\downarrow}$, then either

- $t \in S_p$, or
- $t{\downarrow}$ and, moreover, if $s \xrightarrow{\tau} s'$ then $t \xrightarrow{\tau} t'$ and $(s',t') \in R$.

As usual, a rooted version [5, 62] of branching bisimilarity is needed for compatibility with alternative composition.

Definition 4.4.3 (Root condition)**.** Process graphs s and t are *rooted termination-sensitive branching bisimilar* (RTSB bisimilar for short), denoted by $s \sim_b^{rt} t$, if and only if there exists a termination-sensitive branching bisimulation R with $(s,t) \in R$, such that:

1. if $s \rightsquigarrow s'$ and $s' \xrightarrow{a} s''$ for some s', $s'' \in S$ and $a \in \mathcal{A}_\tau$, then there exist $t', t'' \in S$ such that $t \rightsquigarrow t'$, $t' \xrightarrow{a} t''$, $(s',t') \in R$, and $(s'',t'') \in R$, and

2. if $s \xrightarrow{a} s'$ for some $s' \in S$ and $a \in \mathcal{A}_\tau$, then there exist $t' \in S$ such that $t \xrightarrow{a} t'$, $(s', t') \in R$.

R is called *RTSB bisimulation* for process graphs s and t.

Essentially, the conditions for rooted bisimilarity require that the related processes should be able to simulate exactly their first actions, i.e. the first τ-action is not considered inert. Thus, processes $\tau.a.0$ and $a.0$ are not related. Otherwise, the alternative compositions of each of them with $b.0$ are not branching bisimilar and the relation would not be a congruence for alternative composition.

Proposition 4.4.4. *$\sim_b^{rt}$ is an equivalence relation.*

Proof. See proof of Theorem 3.2.5. □

Theorem 4.4.5 (Congruence theorem)**.** *RTSB bisimilarity $\sim_b^{rt}$ is congruence with respect to the operators in $pTCP_\tau$.*

Proof. Follows the lines of the proof of Theorem 3.3.4 for the merge operator. □

4.4.3 Axiomatization

We give a ground-complete axiomatization of RTSB bisimilarity, i.e. axiomatization for closed process expressions, for the operators in pTCP_τ .

The following predicate function, defined on a subset of the process expressions, will be needed for the axiom that is typical for RTSB bisimilarity.

Definition 4.4.6. Let X be the set of pTCP_τ process expressions that contain only the operators $\oplus$, $+$ and $a.\_$ for $a \in \mathcal{A}_\tau$, and the constants 1 and 0. The boolean function $\mathrm{ter} : X \mapsto \{\mathsf{true}, \mathsf{false}\}$ is defined inductively in the following way:

- $\mathrm{ter}\,(1) = \mathsf{true}$;
- $\mathrm{ter}\,(0) = \mathsf{false}$;
- $\mathrm{ter}\,(a.x) = \mathsf{false}$;
- $\mathrm{ter}\,(x \oplus_\pi y) = \mathrm{ter}\,(x)$ or $\mathrm{ter}\,(y)$;
- $\mathrm{ter}\,(x + y) = \mathrm{ter}\,(x)$ or $\mathrm{ter}\,(y)$.

Basically, ter (x) is true if x with a positive probability can terminate immediately, without performing any actions from $\mathcal{A}_\tau$.

The ground-complete axiomatization of rooted branching bisimilarity is given in Table 4.4. Note that a condition of type $x = x + x$ appears several times in the axioms. It describes processes that are not initially probabilistic in nature (see also [4]). For example, it does not hold for $x \equiv a.0 \oplus_{1/2} b.0$, but it applies if $x \equiv a.0 + b.0$ or $x \equiv a.0 \oplus_{1/2} a.0$. It can be checked that, indeed, if $x \sim_b^{rt} x+x$, then for every x', x'' such that $x \rightsquigarrow x'$ and $x \rightsquigarrow x''$, it holds $x' \sim_b x''$.

We briefly discuss the most notable axioms of Table 4.4. As already noticed, axiom $x+x = x$ is not valid in general (see also [3,5,11]). However, this axiom still holds for processes that start with action transitions or for $x \equiv 1$, captured by the axioms AA3 and EA3. Axioms P1-P5 are inherited from the strong probabilistic bisimulation setting [3, 11]. There are three main laws that axiomatize the parallel composition operator: PM1 and PM2, which describe the interplay of $\|$ and $\oplus$, and the conditional axiom M, which states on which processes the interleaving can be performed. PM1 and PM2 express that $\|$ distributes (left and right) over $\oplus$. As long as at least one of the two parallel processes starts with a probability distribution, these laws will be applied. Axiom PrB characterizes RTSB bisimilarity. It is a counterpart of the branching axiom in the nonprobabilistic setting [62]: $a.\,(y + \tau.\,(y + z)) = a.\,(y + z)$. Due to the conditions $y = y + y$ and $z = z + z$, it removes only a τ step that is followed by a process with a trivial (Dirac) outermost distribution, namely $y + z$. On the other hand, the condition ter $(y + z) = \mathsf{false}$ requires that $y + z$ does not terminate. Thus, a τ step that precedes a process that terminates cannot be eliminated, for the reasons explained in Subsection 4.4.2. In order to apply axiom PrB, process expression $y + z$ should contain only the operators $\oplus$, $+$ and $a._-$ for $a \in \mathcal{A}_\tau$, and the constants 1 and 0. The following theorem guarantees that every process expression can be rewritten only in terms of those operators. The theorem will be also useful later, in the proof of the completeness of the axiomatization.

Theorem 4.4.7 (Elimination theorem)**.** *Let p be a $pTCP_\tau$ closed expression. Then there is a $pTCP_\tau$ expression q without the operators $\|$, $\between$, $|$, $\cdot$, τ_I, and ∂_H, such that $pTCP_\tau \vdash p = q$.*

Proof. By structural induction on the structure of p (see also [11]). □

The following two theorems guarantee the soundness and the completeness of the axiomatization in Table 4.4 with respect to RTSB bisimilarity.

Theorem 4.4.8 (Soundness)**.** *Let p and q be closed expressions represented by process graphs x and y, respectively. Then, $pTCP_\tau \vdash p = q$ implies $x \sim_b^{rt} y$.*

Table 4.4: Axioms of pTCP_τ. $a,\ b \in \mathcal{A}_\tau\ I, H \subseteq \mathcal{A},\ w, x, y, z \in V$

A1	$x + y = y + x$	P1	$x \oplus_\pi y = y \oplus_{1-\pi} x$
A2	$(x + y) + z = x + (y + z)$	P2	$x \oplus_\pi (y \oplus_\rho z) = \left(x \oplus_{\frac{\pi}{\pi+\rho-\pi\rho}} y\right) \oplus_{\pi+\rho-\pi\rho} z$
AA3	$a.x + a.x = a.x$	P3	$x \oplus_\pi x = x$
EA3	$1 + 1 = 1$	P4	$(x \oplus_\pi y) \cdot z = x \cdot z \oplus_\pi y \cdot z$
A4	$x + 0 = x$	P5	$(x \oplus_\pi y) + z = (x + z) \oplus_\pi (y + z)$
A5	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	A8	$x \cdot 1 = x$
A6	$0 \cdot x = 0$	A9	$1 \cdot x = x$
A7	$(x + y) \cdot z = x \cdot z + y \cdot z$	A10	$a.x \cdot y = a.(x \cdot y)$

TI1	$\tau_I(a.x) = a.\tau_I(x)$ if $a \notin I$	D1	$\partial_H(a.x) = a.\partial_H(x)$ if $a \notin H$
TI1'	$\tau_I(a.x) = \tau.\tau_I(x)$ if $a \in I$	D2	$\partial_H(a.x) = 0$ if $a \in H$
TI2	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	D3	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$
TI3	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$	D4	$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$
TI4	$\tau_I(x \oplus_\pi y) = \tau_I(x) \oplus_\pi \tau_I(y)$	D5	$\partial_H(x \oplus_\pi y) = \partial_H(x) \oplus_\pi \partial_H(y)$
TI5	$\tau_I(1) = 1$	D6	$\partial_H(1) = 1$
TI6	$\tau_I(0) = 0$	D7	$\partial_H(0) = 0$

M	$x \parallel y = x \between y + y \between x + x \mid y$	if	$x = x + x$ and $y = y + y$
PM1	$x \parallel (y \oplus_\pi z) = (x \parallel y) \oplus_\pi (x \parallel z)$	PC1	$x \mid (y \oplus_\pi z) = (x \mid y) \oplus_\pi (x \mid z)$
PM2	$(x \oplus_\pi y) \parallel z = (x \parallel z) \oplus_\pi (y \parallel z)$	PC2	$(x \oplus_\pi y) \mid z = (x \mid z) \oplus_\pi (y \mid z)$
PL1	$x \between (y \oplus_\pi z) = (x \between y) \oplus_\pi (x \between z)$	C1	$x \mid (y + z) = x \mid y + x \mid z$
PL2	$(x \oplus_\pi y) \between z = (x \between z) \oplus_\pi (y \between z)$	C2	$(x + y) \mid z = x \mid z + y \mid z$
LM1	$1 \between x = 0$	C3	$1 \mid 1 = 1$
LM2	$a.x \between y = a.(x \parallel y)$ if $y = y + y$	C4	$0 \mid x = 0$
LM3	$(x + y) \between z = x \between z + y \between z$	C5	$a.x \mid 1 = 0$
LM4	$0 \between x = 0$	C6	$a.x \mid b.y = c.(x \parallel y)$ if $\gamma(a, b) = c$
		C7	$a.x \mid b.y = 0$ if $\gamma(a, b)$ undefined
		C8	$x \mid 0 = 0$
		C9	$1 \mid a.x = 0$

PrB	$a.((y + \tau.(y + z)) \oplus_\pi w) = a.((y + z) \oplus_\pi w)$ if $y = y + y$, $z = z + z$, and $\mathrm{ter}(y + z) = \mathsf{false}$.

Proof. Straightforward (see [3, 4, 11]). □

The proof of the following theorem is given in Subsection 4.4.3.1.

Theorem 4.4.9 (Ground-completeness)**.** *Let p and q be closed $pTCP_\tau$ expressions represented by process graphs x and y, respectively. Then, $x \sim_b^{rt} y$ implies $pTCP_\tau \vdash p = q$.*

4.4.3.1 Proof of Theorem 4.4.9

The proof of Theorem 4.4.9 is based on the method of process graph transformations [21] and builds on the proof of the corresponding theorem in [62]. We define a confluent and terminating rewriting system on finite process graphs (i.e. process graphs without infinite paths), such that every rewriting step corresponds to an equation in the algebra. In this way, every process graph reduces to a normal form. We prove that if two process graphs are RTSB bisimilar, then they have the same normal forms; thus, their corresponding process expressions can be proved equal in pTCP_τ.

More precisely, we use the colouring definition of branching bisimilarity of Section 4.2, adapted for RTSB bisimilarity. When a process graph is in a normal form, all its states have different blends, and, given two arbitrary states, there is at most one probabilistic transition between them.

A finite process graph is reduced to its normal form by repeating the following steps, as long as it is possible:

- Find a pair of nondeterministic states that have equal action and terminating potentials, and remove one of the states.
- Find a pair of probabilistic states that lead to the same classes with the same probabilities and remove one of the states.
- Find a pair of different transitions $s \overset{\pi}{\rightsquigarrow} s'$ and $s \overset{\rho}{\rightsquigarrow} s'$ and merge them.
- Find a transition $s \overset{1}{\rightsquigarrow} t$ and eliminate it.
- Find a manifestly inert transition and eliminate it (a transition $s \xrightarrow{\tau} t$, where s is not preceded by the root or is not the root itself, and neither s nor t terminates, is *manifestly inert* if all of the action transitions originating from s, that are different from $s \xrightarrow{\tau} t$, can also be performed by t.)

While the first four steps correspond to axioms for strong bisimilarity, the fifth step abstracts away from the unobservable transitions and corresponds to axiom PrB.

We proceed with a detailed presentation of the proof. In the completeness proof, we are going to consider only *finite* process graphs.

Definition 4.4.10 (Finite process graphs)**.** A process graph is *finite* if it does not have infinite paths.

In Section 4.2 we have defined canonical colouring of a PTS. Here we adapt the definition for process graphs. The following two definitions facilitate that.

Definition 4.4.11 (Rooted colouring)**.** Let x be a finite process graph. If x is a nondeterministic state, then a colouring of the states of x is *rooted* iff x does not have the same colour as a non-root state. If x is a probabilistic state, then a colouring of the states of x is *rooted* when, if a state x', such that $x \rightsquigarrow x'$, has the same colour as a state x'', then $x \rightsquigarrow x''$.

Definition 4.4.12 (Termination-sensitive colouring)**.** A colouring of a process graph is *termination-sensitive* iff a terminating state does not have the same colour as a nonterminating state, and, moreover, given two terminating states s and t with equal blends, if $s \xrightarrow{\tau} s'$ then $t \xrightarrow{\tau} t'$ such that s' and t' have the same blends.

A colouring of a process graph that is both rooted and termination-sensitive is called *RTS colouring.*

The following proposition recasts Corollary 4.2.8 in the new setting.

Proposition 4.4.13. *Given a finite process graph x, there exists a proper, consistent, RTS colouring for x, called* canonical colouring*, such that two states have the same blend if and only if they are related by an RTSB autobisimulation for x.*

We define the subset of process graphs that are in normal form.

Definition 4.4.14 (Normal form)**.** A finite process graph is in *normal form* if and only if, given its canonical colouring, all the states have different blends, and, given two arbitrary states, there is at most one probabilistic transition between them.

Next, we show that every finite process graph is RTSB bisimilar to exactly one normal form, up to isomorphism. Given a process graph x, by S^x we denote the set of states that belong to x, and similarly for S^x_n and S^x_p.

Definition 4.4.15. For a given finite process graph x, graph $\mathrm{NF}(x)$ is defined in the following way:

– The nondeterministic states are defined by the set of pure colours in the canonical colouring χ of x, i.e. every colour c that appears in

the latter defines a nondeterministic state in $\mathrm{NF}(x)$. $\mathrm{NF}(x)$ has a transition $\mathsf{c} \xrightarrow{a} \mathsf{c}'$ $(a \neq \tau)$ iff x has a transition $r \xrightarrow{a} r'$, where $\chi(r) = \mathsf{c}$ and $\chi(r') = \mathsf{c}'$. $\mathrm{NF}(x)$ has a transition $\mathsf{c} \xrightarrow{\tau} \mathsf{c}'$ iff $\mathsf{c} \neq \mathsf{c}'$ and x has a transition $r \xrightarrow{\tau} r'$ where $\chi(r) = \mathsf{c}$ and $\chi(r') = \mathsf{c}'$. A state c in $\mathrm{NF}(x)$ is terminating iff x has a terminating state r such that $\chi(r) = \mathsf{c}$.

– The probabilistic states are defined by the (non-pure colour) blends in the canonical colouring of x. If $\mathsf{b} \in S^{\mathrm{NF}(x)}$ is a probabilistic state, then $\mathrm{NF}(x)$ has a transition $\mathsf{b} \overset{\pi}{\rightsquigarrow} \mathsf{c}$ iff $\pi \neq 0$ and $\mathsf{b}(\mathsf{c}) = \pi$.

It is easy to check that $\mathrm{NF}(x)$ of the above definition is a process graph in a normal form according to Def. 4.4.14.

Proposition 4.4.16. *A finite process graph x is RTSB bisimilar to* $\mathrm{NF}(x)$.

Proof. Consider the canonical colouring of x and the trivial colouring of $\mathrm{NF}(x)$ induced by its construction. Then x and $\mathrm{NF}(x)$ have the same coloured traces. □

Thus, each class of RTSB bisimilar process graphs contains a process graph in normal form. It is left to show that there is only one such graph, up to isomorphism. Isomorphism of process graphs is defined next.

Definition 4.4.17 (Isomorphism)**.** Let x and y be process graphs such that there is at most one probabilistic transition between any two states. Process graphs x and y are *isomorphic* iff there exists a bijection $f\colon S^x \mapsto S^y$ such that $f(x) = y$, and for $s, s' \in S^x$ and $a \in \mathcal{A}_\tau$

– $s \xrightarrow{a} s'$ iff $f(s) \xrightarrow{a} f(s')$,

– $s{\downarrow}$ iff $f(s){\downarrow}$, and

– $s \overset{\pi}{\rightsquigarrow} s'$ iff $f(s) \overset{\pi}{\rightsquigarrow} f(s')$.

Proposition 4.4.18. *Two finite process graphs in normal form are RTSB bisimilar iff they are isomorphic.*

Proof. ($\Longrightarrow$) Let x and y be process graphs in normal form such that $x \sim_b^{rt} y$. Let χ be the canonical colouring of the PTS induced by process graphs x and y, and, moreover, let χ be rooted for x and for y and termination-sensitive. Then χ is canonical colouring for process graph x and also for process graph y. Define $f : S^x \mapsto S^y$, such that $f(x) = y$ iff $\chi(x) = \chi(y)$. From the fact that x and y are in normal form and from $x \sim_b^{rt} y$ it follows that this mapping is well-defined and, moreover, a bijection. It is now not hard to check that f satisfies the conditions of Definition 4.4.17 (see also [62]).

□

From Proposition 4.4.18 and Proposition 4.4.16 it follows that every process graph has a RTSB bisimilar normal form, unique up to isomorphism. We can thus refer to $\mathrm{NF}(x)$ as *the normal form* of x.

Next, we define double states and manifestly inert transitions.

Definition 4.4.19. Given a finite process graph x such that there is at most one probabilistic transitions between two states,

1. A pair $(s, s') \in S_n^x \times S_n^x$, where $s \neq s'$, $x \neq s$, $x \neq s'$, $x \not\rightsquigarrow s$, and $x \not\rightsquigarrow s'$, is a pair of *nondeterministic double states* iff, for all $a \in \mathcal{A}$ and $t \in S$,
   - $s \xrightarrow{a} t$ iff $s' \xrightarrow{a} t$, and
   - $s\downarrow$ iff $s'\downarrow$, and in case $s\downarrow$, $s \xrightarrow{\tau} t$ iff $s' \xrightarrow{\tau} t$.
2. A pair $(s, s') \in S_p^x \times S_p^x$, where $s \neq s'$, is a pair of *probabilistic double states* iff, for all $t \in S_n$ and $\pi \in (0, 1]$, $s \overset{\pi}{\rightsquigarrow} t$ iff $s' \overset{\pi}{\rightsquigarrow} t$.
3. A transition $s \xrightarrow{\tau} s'$ such that $s' \in S_n$, $x \not\rightsquigarrow s$, $x \neq s$, $s \not\downarrow$ and $s' \not\downarrow$, is *manifestly inert* iff, for all actions a and states t with $(a, t) \neq (\tau, s')$, it holds that $s \xrightarrow{a} t$ implies $s' \xrightarrow{a} t$.

Proposition 4.4.20. *Let x be a finite process graph without nondeterministic or probabilistic double states, without $\overset{1}{\rightsquigarrow}$ or manifestly inert transitions, and such that given any two probabilistic states, there is at most one probabilistic transition between them. Then, x is in normal form.*

Proof. Suppose that x is not in normal form. Then, if there is no more than one probabilistic transitions between two states, given the canonical colouring χ of x, there are two different states with equal blends.

Suppose first that x has two probabilistic states s and r with equal blends and there are no nondeterministic states with equal pure colours. Then it easily follows that if, for some t, $s \overset{\pi}{\rightsquigarrow} t$ then $r \overset{\pi}{\rightsquigarrow} t$, and vice versa, if $r \overset{\pi}{\rightsquigarrow} t$ then $s \overset{\pi}{\rightsquigarrow} t$. Since there is at most one probabilistic transition between two states, it must hold that s and r are probabilistic double states. Thus, we obtain a contradiction.

Suppose now that there exist nondeterministic states with equal colours. If there exist probabilistic double states then the proof is finished. Therefore, assume that there are no probabilistic double states. We define the *depth* $d(s)$ of a nondeterministic state s to be the number of transitions in the longest path starting from s, and the *total depth* of two states to be the sum of their individual depths. Let s and r be two different nondeterministic states with

equal colours, such that their total depth is minimal, i.e. if r' and s' are nondeterministic states with equal colours such that $d(r') + d(s') < d(r) + d(s)$, then $r' = s'$. Without loss of generality, we assume that $d(s) \leq d(r)$. We prove that

(i) if $r \xrightarrow{a} t$ and $(a, t) \neq (\tau, s)$, then $s \xrightarrow{a} t$, and

(ii) if $s \xrightarrow{a} t$, then either $r \xrightarrow{\tau} s$ or $r \xrightarrow{a} t$.

(iii) $s\downarrow$ if and only if $r\downarrow$, and if $s\downarrow$, then $s \xrightarrow{\tau} t$ iff $r \xrightarrow{\tau} t$.

From (i), (ii) and (iii) it would follow that, if there exists a transition $r \xrightarrow{\tau} s$, then it is manifestly inert; otherwise (r, s) is a pair of double nondeterministic states. Either way, the proof would be completed. (Note that r and s are both different from the root, since they are equally coloured.)

Proof of (i): Let $r \xrightarrow{a} t$ and $(a, t) \neq (\tau, s)$. Since r and s have equal coloured traces, either $a = \tau$ and t has the same blend as r, or s has a coloured trace starting with $(\chi(r), a, \chi(t))$, where $\chi(y)$ is the colour of state y with respect to the canonical colouring χ of x. In the first case, from $d(t) + d(s) < d(r) + d(s)$ it follows that $t = s$, which contradicts our assumption. So, s has a coloured trace $(\chi(r), a, \chi(t))$. Suppose that $s \xrightarrow{\tau} u$ for a state u with the same colour as s. Then, from $d(s) + d(u) < d(s) + d(r)$, we obtain that $s = u$, which is not possible since x is a finite graph. Thus, we obtain a contradiction. So, there must exist a state u such that $s \xrightarrow{a} u$ and the blends of u and t are equal. If $u = t$, then the proof is complete. Suppose $u \neq t$. We can assume that u and t are not double nondeterministic states; otherwise, from $d(u) + d(t) < d(s) + d(r)$ it follows that $u = t$ and the proof is complete. Let t' and u' be states with the same colour such that $t \rightsquigarrow t'$ and $u \rightsquigarrow u'$. Since $d(t') + d(u') < d(r) + d(s)$, it follows that $t' = u'$. The last, together with the fact that u and t have equal blends, implies that u and t are double probabilistic states, thus contradicting the assumption that there don't exist probabilistic double states.

A similar line of reasoning is applied to the proof of (ii).

Proof of (iii): Since r and s have the same colour, $s\downarrow$ iff $r\downarrow$. Assume now that $s\downarrow$. Assume first that $s \xrightarrow{\tau} t$. We need to prove that $r \xrightarrow{\tau} t$. If t has the same colour as s, then from $d(t) + d(r) < d(s) + d(r)$ it follows that $t = r$. But then $d(s) > d(r)$, and we obtain a contradiction. Therefore, $\chi(t) \neq \chi(s)$. Since s and r have equal coloured traces, $r \xrightarrow{\tau} t'$ and $\chi(t) = \chi(t')$. From $d(t) + d(t') < d(s) + d(r)$ it follows that $t = t'$, which we were supposed to show. Assume now that $r \xrightarrow{\tau} t$. If $\chi(t) = \chi(r)$, then from $d(s) + d(t) < d(s) + d(r)$ we have that $s = t$. But then $s \xrightarrow{\tau} s'$ such that $\chi(s) = \chi(s')$

and $s \neq s'$. Then, from $d(s) + d(s') < d(r) + d(s)$ we obtain that $s = s'$, which is not possible in a finite graph. So, $\chi(t) \neq \chi(r)$. Then, $s \xrightarrow{\tau} t'$ and $\chi(t) = \chi(t')$. From $d(t) + d(t') < d(s) + d(r)$ we obtain $t = t'$, which completes the proof. □

From Proposition 4.4.20 it follows that, in order to obtain the normal form of a finite graph x, we need to unify the pairs of double states, to contract the manifestly inert transitions, to delete the $\overset{1}{\rightsquigarrow}$ transitions, and to merge multiple probabilistic transitions between any pair of states. We now define formally these rewriting steps that reduce a process graph to its normal form.

Definition 4.4.21. The process graph rewriting relation $\rightarrowtail_{\mathrm{NF}}$ is defined by the following one-step reductions:

1. Unifying a pair of nondeterministic double states (s, s'): replace every transition $t \overset{\pi}{\rightsquigarrow} s$ by $t \overset{\pi}{\rightsquigarrow} s'$ and replace every transition $t \xrightarrow{a} s$ by $t \xrightarrow{a} s'$.

2. Unifying a pair of probabilistic double states (t, t'): replace every transition $s \xrightarrow{a} t$ by $s \xrightarrow{a} t'$.

3. Contracting a manifestly inert transition $s \xrightarrow{\tau} t$: replace every transition $r \xrightarrow{a} s$ by $r \xrightarrow{a} t$ and replace every transition $r \overset{\pi}{\rightsquigarrow} s$ by $r \overset{\pi}{\rightsquigarrow} t$.

4. Erasing a transition $s \overset{1}{\rightsquigarrow} t$: replace every transition $r \xrightarrow{a} s$ by $r \xrightarrow{a} t$.

5. Merging all probabilistic transitions between s and t: replace all transitions $s \rightsquigarrow t$ by a single transition $s \overset{\pi}{\rightsquigarrow} t$, where $\pi = P(s, t)$.

Proposition 4.4.22. *For the set of finite process graphs, the rewriting relation $\rightarrowtail_{\mathrm{NF}}$ satisfies the following properties:*

(ii) If $x \rightarrowtail_{\mathrm{NF}} y$ then $x \sim_b^{rt} y$.

(iii) The relation $\rightarrowtail_{\mathrm{NF}}$ is confluent and terminating.

Proof. (ii) The relation $\rightarrowtail_{\mathrm{NF}}$ is terminating since in every step it decreases the number of states or the number of transitions that belong to the process graph. For confluence, assume that process graphs x' and x'' can be obtained from x by applying $\rightarrowtail_{\mathrm{NF}}$ as long as it is possible. By Proposition 4.4.20, x' and x'' are both in normal form. By (ii) they are branching bisimilar. So, by Proposition 4.4.18, they are isomorphic. □

Next, we establish a relation between the finite process graphs and the closed expressions. First we introduce some notation. Let $\{a_i\}_{i=1}^n$ be a sequence of actions and $\{x_i\}_{i=1}^n$ be a sequence of process expressions. By $\sum_{i=1}^n a_i.x_i$ we denote a process expression that has the form $a_1.x_1 + a_2.x_2 + \ldots + a_n.x_n$, where the brackets "(" and ")" are omitted. Let x be a probabilistic state and $\{x \stackrel{\pi_i}{\rightsquigarrow} x_i\}_{i=1}^n$ be a sequence of all the probabilistic transitions originating from x. Process graph $[x]$ is constructed such that a new state $[x]$ is created and for every transition $x \stackrel{\pi_i}{\rightsquigarrow} x_i$, $i > 1$, a transition $[x] \stackrel{\pi_i/(1-\pi_1)}{\rightsquigarrow} x_i$ is created.

Definition 4.4.23. Let x be a finite process graph. A process expression $\langle x \rangle$ is associated to x in the following way.

$$\langle x \rangle = \begin{cases} 0 & \text{if } x \text{ is a deadlock state} \\ \sum_{i=1}^n a_i.\langle x_i \rangle & \text{if } \exists \{x \xrightarrow{a_i} x_i\}_{i=1}^n,\ x \not\downarrow \\ 1 & \text{if } x \not\rightarrow,\ x\downarrow \\ \sum_{i=1}^n a_i.\langle x_i \rangle + 1 & \text{if } \exists \{x \xrightarrow{a_i} x_i\}_{i=1}^n,\ x\downarrow \\ \langle x' \rangle & \text{if } x \stackrel{1}{\rightsquigarrow} x' \\ x_1 \oplus_{\pi_1} \langle [x] \rangle & \text{if } \exists \{x \stackrel{\pi_i}{\rightsquigarrow} x_i\}_{i=1}^n \end{cases}$$

Note that the process expression $\langle x \rangle$ is determined modulo the axioms for commutativity and associativity, i.e. axioms A1, A2, P1 and P2 from Table 4.4.

Proposition 4.4.24. *If x and y are isomorphic graphs, then*

$$\mathrm{A1, A2, P1, P2} \vdash \langle x \rangle = \langle y \rangle.$$

Proof. Straightforward (see [3]). □

The following lemmas show that every process graph rewriting step can be mimicked in the process theory pTCP_τ.

Lemma 4.4.25. *Suppose (s,t) is a pair of double states, either probabilistic or nondeterministic, and $a \in \mathcal{A}_\tau$. Then,*

$$pTCP_\tau \vdash a.\langle s \rangle = a.\langle t \rangle \quad \textit{and}$$
$$pTCP_\tau \vdash \langle s \rangle \oplus_\pi w = \langle t \rangle \oplus_\pi w.$$

Proof. See [5, 62]. □

Lemma 4.4.26. $pTCP_\tau \vdash (x \oplus_\pi x) \oplus_\rho w = x \oplus_\rho w$.

Proof. Trivial. □

Lemma 4.4.27. *Let $s \xrightarrow{\tau} s'$ be a manifestly inert transition and $a \in \mathcal{A}_\tau$.*

$$pTCP_\tau \vdash a.\langle s\rangle = a.\langle s'\rangle \quad \textit{and}$$
$$pTCP_\tau \vdash a.\left(\langle s\rangle \oplus_\pi w\right) = a.\left(\langle s'\rangle \oplus_\pi w\right).$$

Proof. Since $s \xrightarrow{\tau} s'$ is a manifestly inert transition, we know that

$$\mathrm{A1, A2, P1, P2} \vdash \langle s\rangle = y + \tau\cdot\langle s'\rangle \quad \text{and} \quad \mathrm{A1, A2, P1, P2} \vdash \langle s'\rangle = y + z$$

for some closed expressions y and z. Moreover, $\langle s'\rangle = \langle s'\rangle + \langle s'\rangle$ and $\mathrm{ter}\left(\langle s\rangle\right) = \mathrm{ter}\left(\langle s'\rangle\right) = \mathsf{false}$. Then,

$$\begin{aligned}\mathrm{pTCP}_\tau \vdash a.\langle s\rangle =& a.(y + \tau\cdot\langle s'\rangle)\\ =& a.(y + \tau\cdot(y + z))\\ =& a.((y + \tau\cdot(y + z)) \oplus_\pi (y + \tau\cdot(y + z)))\\ =& \text{ (twice PrB and P3) } a.(y + z)\\ =& a.\langle s'\rangle.\end{aligned}$$

$$\begin{aligned}\mathrm{pTCP}_\tau \vdash a.\left(\langle s\rangle \oplus_\pi w\right) =& a.\left((y + \tau\cdot\langle s'\rangle) \oplus_\pi w\right)\\ =& a.\left((y + \tau\cdot(y + z)) \oplus_\pi w\right)\\ =& a.\left((y + z) \oplus_\pi w\right)\\ =& a.\left(\langle s'\rangle \oplus_\pi w\right).\end{aligned}$$

□

Lemma 4.4.28. *If $x \rightarrowtail_{\mathrm{NF}} y$, then $pTCP_\tau \vdash \langle x\rangle = \langle y\rangle$.*

Proof. By Lemmas 4.4.25, 4.4.26, and 4.4.27. □

Next, we restate (the ground-completeness) Theorem 4.4.9 and give a proof. Because of the Elimination Theorem 4.4.7, the ground-completeness theorem can take the following form.

Theorem 4.4.9 (Ground-completeness)**.** Let x, y be finite process graphs such that $x \sim_b^{rt} y$. Then $\mathrm{pTCP}_\tau \vdash \langle x\rangle = \langle y\rangle$.

Proof. Since $x \sim_b^{rt} y$, if $\mathrm{NF}(x)$ and $\mathrm{NF}(y)$ are the normal forms of x and y, respectively, then from Proposition 4.4.16 it follows that $\mathrm{NF}(x) \sim_b^{rt} x$ and $y \sim_b^{rt} \mathrm{NF}(y)$. Thus, $\mathrm{NF}(x) \sim_b^{rt} \mathrm{NF}(y)$, which, according to Proposition 4.4.18, implies that $\mathrm{NF}(x)$ and $\mathrm{NF}(y)$ are isomorphic. Therefore,

$$\mathrm{pTCP}_\tau \vdash \langle \mathrm{NF}(x) \rangle = \langle \mathrm{NF}(y) \rangle.$$

On the other hand, since x can be rewritten to $\mathrm{NF}(x)$, we have

$$\mathrm{pTCP}_\tau \vdash \langle x \rangle = \langle \mathrm{NF}(x) \rangle.$$

Similarly for y and $\mathrm{NF}(y)$. Finally,

$$\mathrm{pTCP}_\tau \vdash \langle x \rangle = \langle \mathrm{NF}(x) \rangle = \langle \mathrm{NF}(y) \rangle = \langle y \rangle.$$

□

Chapter 5

Concluding remarks to part I

We discuss related work of the results presented in this part, and end with concluding remarks.

5.1 Related work

As closely related to the results presented in Part I we consider those concerning weak types of bisimulations for systems exhibiting both probabilistic and nondeterministic behaviour. Weak and branching bisimulations for the alternating model of probabilistic systems have been defined in [91] and [8] respectively, and the latter equivalence has been axiomatized in [5], for a language with sequential composition, alternative composition, probabilistic choice, and hiding. However, as shown in [4], none of these bisimulations is congruence for the standard parallel composition of [68]. As explained in the introduction, this was our motivation to define a new branching bisimulation.

Recently, a branching bisimulation equivalence relation has been defined in a model, which is inspired by the alternating model, and where the probabilistic steps are timed [36] (this model can be also seen as a discrete-time variant of the interactive Markov chains [70]). Both action and timed probabilistic transitions are possible from the same state. This allows flexibility of the parallel composition and, thus, relating e.g. states s and t in Figure 2.1 does not break the congruence property of the relation. On the other hand, an action transition cannot be simulated by a path containing probabilistic transitions, e.g. processes s and u in Figure 3.1a are not equivalent by the relation of [36].

The alternating model of probabilistic systems that we consider here allows full nondeterminism and probabilistic choice. A closely related model is the non-alternating model [97], which also has the same modeling capabili-

ties. In the latter, each probabilistic choice is guarded by an action. A process makes a nondeterministic choice among several, possibly equal, actions, and the execution of an action leads to a probability distribution over the next states. Because of the similarities between the non-alternating and the alternating model, possible translating and embedding functions have been studied [17, 100, 102]. One can map a process in the alternating model to a process in the non-alternating model by introducing a special, invisible action for the probabilistic choice [100]. Alternatively, a translation can be made by "erasing" each probabilistic state that comes between an action transition and a probabilistic transition [17, 100, 102]. However, differences occur when composing processes. For example, the first mapping is not compositional with respect to parallel composition: the image of the parallel composition of $a.\,(b.0 \oplus_\pi c.0)$ and $d.0$ is not equal to the parallel composition of the images of $a.\,(b.0 \oplus_\pi c.0)$ and $d.0$. The second mapping, as it can be anticipated, makes differences when composing in parallel, or with a probabilistic choice, two initially probabilistic processes (e.g. $a.0 \oplus_{1/3} b.0$ and $c.0 \oplus_{1/2} d.0$). The reason for the differences is the fact that in the non-alternating model each probabilistic choice requires at least a τ-action to guard it.

Weak and branching bisimulations for the non-alternating model have been defined in [99]. Weak bisimulation has been axiomatized in [41]. For the weak versions of bisimulation in this model to be transitive, it should be allowed for a single transition to be simulated by a convex combination of several transitions. A side-effect of this property is that these relations are not decidable in polynomial time [29]. Interestingly however, processes $a.\tau.\,(b.0 \oplus_\pi c.0)$ and $a.\,(b.0 \oplus_\pi c.0)$ are neither related by our bisimulation, nor by [99], when the second (compositional) mapping is used.

In [45] the semantics of pCTL* of [23] is extended to treat internal steps. Behaviors, on which path formulas are interpreted, ignore states with τ steps, together with the τ steps. With this semantics it is shown in [45] that the notion of weak bisimulation of [91] is sound and complete for the presented logic. However, this semantics implies that two non-probabilistic systems that satisfy the same formulas are not necessary branching bisimilar by [62]. On the contrary, we extend the semantics in such a way, that if two non-probabilistic systems satisfy the same formulas, then they are branching bisimilar in the sense of [62] (recall that CTL characterizes non-probabilistic branching bisimilarity [40]). For these reasons, we built on [40], rather than on [45] or [99].

We close the discussion on related work by returning once again to the basic problem with compositionality for the weak types of bisimulations in the alternating model. Namely, while we have taken the conservative approach of strengthening the bisimulation in question to become compatible

with the standard parallel composition, there is still the more radical option of adapting the operator to support compositionality of the bisimulation. Indeed, reference [4] notes that parallel composition can be defined by giving priority to the internal actions [4] over the visible actions, so that the (weak) bisimulations defined in [91] and [8] can be compositional. Note that this approach opposes those in the bisimulation-based process algebras as ACP [11] and CCS [86], where the internal actions and the observable actions are treated equally when composing processes and the accent is on preserving the branching structure of processes. In fact, the parallel composition in [4] is very similar to the CSP parallel composition [71], where the underlying semantics is trace-style, i.e. linear-time, and thus the exact moment in which the internal activity occurs cannot be determined. Part II of the present text is dedicated to probabilistic trace-style semantics and a process algebra in the style of CSP.

5.2 Concluding remarks

In this part, we have defined a branching bisimulation for the alternating model of probabilistic systems that is a congruence for parallel composition, as well as for the rest of the standard operators. For the congruence result, an internal transition that is immediately followed by a non-trivial probabilistic distribution is not considered inert, and, thus, is not eliminated by a branching bisimulation reduction. We have shown that our branching bisimulation is the coarsest congruence for parallel composition that is included in the relation of [8], which, together with [91], are the intuitive notions for branching, respectively, weak bisimulation for the alternating model. In addition, we have shown that branching bisimulation preserves probabilistic CTL formulas, and have given an algorithm that decides branching bisimilarity in polynomial time in the number of states. As intermediate results, two alternative definitions of branching bisimilarity are presented: one based on coloured traces, and another involving schedulers. Finally, a complete axiomatization of branching bisimilarity has been given, that includes the operators of TCP [11] and probabilistic choice. Similarly to the non-probabilistic case [62], only one axiom characterizes branching bisimilarity, in addition to the axioms that are already valid in the strong bisimilarity setting.

In [6] the Concurrent Alternating Bit Protocol has been verified using the equational theory presented in this part, and several sound recursive verification rules. In the future it would be interesting to study whether other, more complex protocols, can be verified using the same techniques. Also, it would be interesting to exploit whether, indeed, branching bisimulation minimiza-

tion improves the efficiency of the verification of large and complex systems. Results in this direction have already been shown for strong probabilistic bisimulation minimization [76].

Part II

Testing semantics

Chapter 6

Introduction

A central question in the theory of concurrent processes is when two processes should be considered equivalent. Various attempts to answer this question have led to the concepts of observational equivalence, bisimulation, testing equivalence, failure equivalence, etc. The underlying semantics of the may/must testing theory for concurrent processes [39, 69] considers two processes equivalent if and only if they cannot be distinguished when interacting with their environment, which is an arbitrary process itself. With this reasoning, this theory takes the view that the exact moment an internal (nondeterministic) choice in a process is resolved is *unobservable* (contrary to bisimulation-based process theories where this moment is considered observable, e.g. [11, 85]). The same aspect appears as well in the failures semantics of the language CSP [25, 71, 96], and in fact, it has been shown in [89] that testing equivalence [39, 69] and failures equivalence [25] coincide for a broad class of processes.

In order to capture quantitative aspects of the system behaviour, some of the internal choices in a process may be refined by probabilistic choices. In this case, the probability with which a certain event happens depends on the way the remaining nondeterminism is resolved. This nondeterminism is usually resolved by means of a *scheduler*, a function that, each time a process has to make an internal choice, selects one of the alternatives. For example, consider the process $s \| u$ in Fig. 6.1, which has one probabilistic choice and two internal choices (*left* and *right*, depending on the result of the probabilistic choice). There are four possible schedulers for this process: one scheduler selects the left τ-transition in the left internal choice and the left τ-transition in the right internal choice; another scheduler selects the left τ-transition in the left internal choice and the right τ-transition in the right internal choice, etc. Application of a scheduler on a process specification without action choices, as the above one, yields a fully probabilistic process,

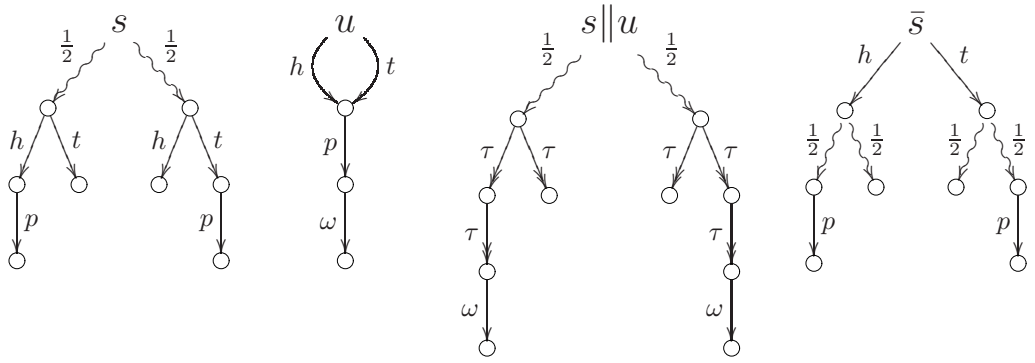

Figure 6.1: The coin-flipping machine and the guessing user

which can be further analyzed using, for instance, standard Markov chain analysis techniques. It is essential, however, that the probability of an event to occur is relative to the particular scheduler.

Such schedulers, that resolve the nondeterminism in all possible ways, were also employed in an extension of the may/must testing theory for the probabilistic case [108]. However, it was soon observed that this semantics yields unrealistic overestimation of the probabilistic behaviour of the process/test synchronization, and the problem was detected exactly in the freedom that the schedulers have [83, 87, 97]. As a result, the internal probabilistic choice in this semantics is *observable*. We explain this phenomenon by the following examples.

Example 6.0.1 (A gambling machine)**.** Consider a system consisting of a machine and a user. The machine is equipped with a menu of two buttons *head* and *tail* via which the user interacts with the machine. The machine, modeled by process graph s in Fig. 6.1, first makes a fair choice (i.e. flips a fair coin), based on which it decides the *winning* button: in one half of the machine runs a prize is given if *head* is pressed by the user, and in the other half of the runs a prize is given if *tail* is pressed. The user, modeled by process graph u in Fig. 6.1, can press any of the two buttons *head* or *tail*. If the right button is pressed, she wins a prize, and is happy (denoted by action ω) afterwards. Note that by no means the user is able to detect the machine's choice beforehand.

Let the user and the machine interact, by synchronizing on their common actions, except on the "happy" action ω. In terms of testing theory [39], process s is tested with test u. Considered as a simple game of chance, by means of probability theory it can be calculated that the user wins a prize with probability $\frac{1}{2}$. However, most of the existing approaches for probabilistic testing, in particular probabilistic may/must testing [42, 44, 74, 90, 98, 108], do not

yield probability $\frac{1}{2}$ for action ω being reported. Let us consider the result of synchronization of the two processes s and u, represented by process $s\|u$ in Fig. 6.1, where actions are hidden after synchronization, labeled by τ. By applying the above mentioned four schedulers to $s\|u$, one obtains the set of probabilities $\{0, \frac{1}{2}, 1\}$ with which action ω can be reported. Therefore, since the power of the schedulers is unrestricted, unrealistic bounds of 0 and 1 for the probability to pass the test are obtained. In fact, when resolving the nondeterminism in $s\|u$, the above mentioned may/must testing approaches allow schedulers that correspond to strategies the user can define and deploy only if she *knows* the result of the coin-flipping *before* guessing. More concretely, such schedulers ignore the fact that both internal choices in $s\|u$ are resolved in the same way, regardless of the outcome of the probabilistic choice.

Consider now process $\bar{s}$ in Fig. 6.1. Process $\bar{s}$ can as well represent the behaviour of the coin-flipping machine from the view point of the user: she cannot observe the exact moment the machine flips the coin, *before* or *after* a button is pressed. According to the user, the machine acts as specified as long as she is able to guess the result in half of the cases. Nevertheless, the two schedulers defined by the probabilistic may/must testing, when applied to process $\bar{s}\|u$, yield exactly probability $\frac{1}{2}$ of reporting action ω. Consequently, none of the approaches in [42, 44, 74, 90, 98, 108] considers processes s and $\bar{s}$ testing-equivalent: they produce different probability bounds for reporting action ω. Note that if the probabilities are ignored, and each probabilistic choice is treated as an internal choice, then processes s and $\bar{s}$ are testing equivalent in the (non-probabilistic) may/must testing theory [39]. □

Example 6.0.2 (The coin flipper – result guesser game)**.** Consider the following game. Player x tosses a fair coin without revealing the outcome and waits. Player y waits while the coin is being tossed, and then writes down his guess about the outcome of the flipping without showing it to x. Then, both players agree to reveal their outcomes, i.e. x to uncover the coin and y to show what he has written. Players x and y are modeled in Fig. 6.2. Obviously, the probability that player y guesses correctly equals $\frac{1}{2}$. However, the schedulers applied to the synchronization of processes x and y give the set of probabilities $\{0, \frac{1}{2}, 1\}$ for reporting ω. This, thus, suggests that there is a strategy for player y by which he can always guess the result correctly. On the other hand, if process $\bar{x}$ in Fig. 6.2 is synchronized with y, both schedulers applied to the resulting process graph yield exactly probability $\frac{1}{2}$ for action ω to happen. Hence, in probabilistic may/must testing theory, processes x and $\bar{x}$ cannot be equated. As a consequence, in this theory prefix

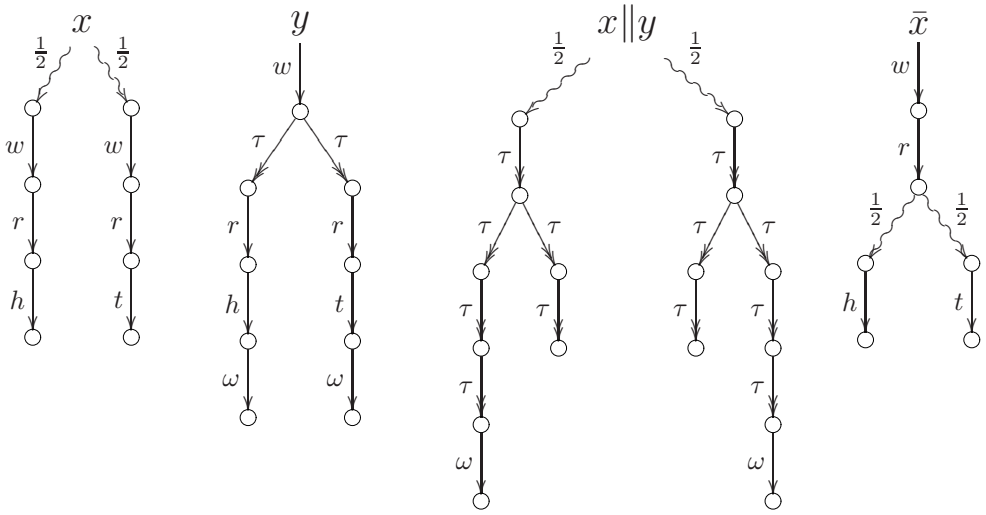

Figure 6.2: The coin-flipper and result-guesser game

does not distribute over probabilistic choice, thus making the moment when an internal probabilistic choice is resolved *observable.*[1]

Note that specifying the *waiting* action of process x with the same action name, regardless of the outcome of the flipping, is essential for player x to keep the outcome unknown to player y. Namely, if the coin flipper after the flip, depending on the outcome, offers different actions for synchronization, a and b, as shown in Fig. 6.3, then the outcome of the flip is revealed to the other player. In this case, the guesser can surely guess the result: he makes his guess depending on the action on which both players previously synchronized. □

It is worth noting that equating processes x and $\bar{x}$ from Fig. 6.2, i.e. allowing distribution of *prefix* over internal probabilistic choice [71], is closely related to equating processes s and $\bar{s}$ in Fig. 6.1, i.e. allowing distribution of *external* action choice over internal probabilistic choice [71], from a point of view of compositionality. Namely, if distribution of external choice over internal probabilistic choice is not allowed under a semantical equivalence, then allowing distribution of action prefix over internal probabilistic choice breaks the congruence property for *interleaving* [71] (or merge) of that equivalence. This can be concluded from Fig. 6.4: relating processes p and q requires relating processes $p|||d$ and $q|||d$, where process d can perform only action d.[2]

[1] Variants of this example were initially discussed in [83, 87, 97].

[2] We refrain from giving a formal definition of interleaving at the moment, but we emphasize that it is inherited from [71].

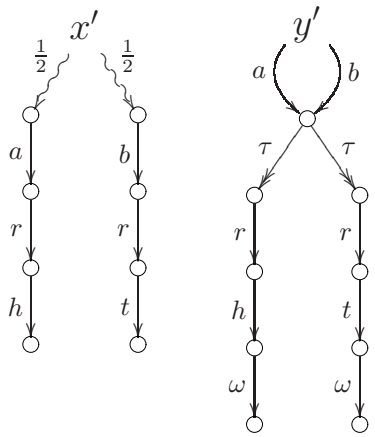

Figure 6.3: The guessing game: unfair play

If we are not able to relate processes that differ only in the moment an internal probabilistic choice is resolved, before or after an action execution, then, for the verification of processes exhibiting external, classical internal and internal probabilistic choice, we can rely only on equivalences that inspect the internal structure of processes, as bisimulations and simulations [58]. Indeed, it has been shown in [44, 74, 84] that the testing preorders

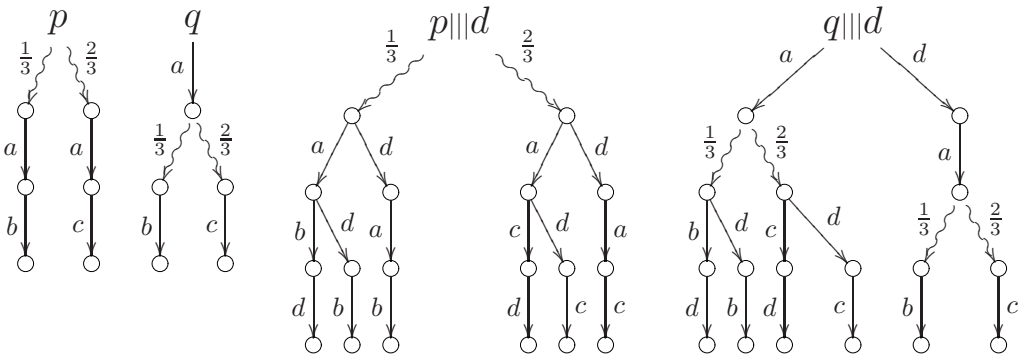

Figure 6.4: Processes p and q, interleaved with action d

defined in [98, 108] are branching time, simulation-like relations. The most discouraging fact is that in a parallel composition probability information might be lost, as the previous examples show. This questions the reasons for adding probabilities in the model to start with.

In this part we propose a testing semantics in the style of [39] for processes exhibiting external, internal and probabilistic choices that solves the problem with overestimation of the probabilities with which a process passes a

test, and achieves unobservability of the internal choice as originally in [39]. The unobservability of the internal choice results from a ready-trace-style characterization of the testing preorder relation, where a ready-trace is an alternating sequence of action menus and performed actions [13, 58, 92]. The testing semantics itself is based on a novel method for labeling the internal transitions that appear in the synchronization of a process and a test, and thus "remembering" the same internal choices that appear in different futures, as those in the above examples. The core idea is that the labels on the internal transitions contain all information based on which the internal choice is resolved. For example, both internal choices in process $s \| u$ in Fig. 6.1 are resolved based on the menu $\{h, t\}$ of action-candidates for synchronization, and thus their labels contain exactly this information, together with the synchronized action itself. In this way, internal choices using the same information are resolved in the same way, regardless of the considered future. In fact, when resolving nondeterminism in order to obtain the probabilities with which a process passes a test, we first resolve the internal nondeterminism of the process, as it is resolved independent of the rest of the nondeterminism, and then we resolve the nondeterminism that results from synchronization, as in process $s \| u$ from Example 6.0.1, and the nondeterminism of the test, as in Example 6.0.2.

Subsequently, we develop an algebraic probabilistic process theory based on the induced ready-trace equivalence that also includes a general parallel composition. This parallel composition allows processes to synchronize on a set of actions and to interleave on the rest of the actions, as in CSP [96], and also allows action hiding after synchronization, as in CCS [85]. The CSP-style axiomatization of the ready-trace equivalence shows that all the distributivity axioms for internal choice [71,96] are preserved, and no new axioms regarding the interplay between external and internal choice are added. Thus, with this approach we solve a problem which was open throughout the years, namely to find a satisfactory extension of CSP with probabilistic choice w.r.t. the verification power of the axioms.

To stay concise while presenting the ideas and the results, in this part we assume that the processes are divergence-free, i.e. no infinite sequences of internal and/or probabilistic transitions exist. In the conclusion to this part we discuss the divergence of processes.

Structure of Part II In Chapter 7 we define our semantical model, the testing preorder relation and the ready-trace preorder relation, and we prove that both preorders coincide. More concretely, the model is defined in Section 7.1, and in Section 7.2 we define how the internal nondeterminism in a

process is resolved, by first performing unfolding and appropriate relabeling. In Section 7.3 we define the synchronization of a process and a test and the testing preorder relation. In Section 7.4 we define the ready-trace preorder relation, and in Section 7.5 we prove that the two preorder relations coincide. In Chapter 8 we define our process language and give a complete axiomatization of the ready-trace equivalence. More concretely, in Section 8.1 we define the choice and priority operators, in Section 8.2 we define the parallel composition operator, in Section 8.3 we define normal forms of processes, in Section 8.4 we give the congruence result, and finally in Section 8.5 we present the axiomatization of the ready-trace equivalence.

Chapter 7

Probabilistic testing theory: Retaining the probabilities

We define a testing preorder relation, in the style of De Nicola & Hennessy's may/must testing preorders, for processes exhibiting external, internal and probabilistic choices. The aim of our testing semantics is to yield realistic estimates of the probability to pass a test, and in this respect to improve the standard probabilistic may/must testing theory. To achieve this, in our model the transitions representing an internal choice are suitably labeled, such that the labels indicate the information based on which the choice is resolved. We also give an alternative characterization of our testing preorder relation in terms of ready-traces. From this characterization it follows that the resulting equivalence relation is insensitive to the exact moments in which the probabilistic and the internal choices occur – a property which is inherent to the original De Nicola & Hennessy's equivalence, but which fails to be preserved by the standard probabilistic may/must testing equivalence.

7.1 Process graphs

In this section we define our model of process graphs and operations on them. Three kinds of choices can be modeled by the process graphs: choice between several different actions (external choice), choice between several internal transitions (internal choice), and probabilistic choice, as a refinement of the internal choice. However, the internal transitions in our model have been assigned labels, local to the process to which they belong. The labels are meant to contain information for the schedulers that resolve the internal

nondeterminism, and after an appropriate unfolding and relabeling function (defined in the next section) is applied, they represent unknown probability distributions. Our model can be also seen as an orthogonal combination of reactive probabilistic processes [80] (or Markov decision processes [94]) and parametric discrete-time Markov chains [46], where the transition probabilities are parameters, without a time component.

Given a directed graph r, by $s \xrightarrow{l} t$ we denote that there exists an edge in r originating from a node s and ending in a node t, labeled with l; we may omit s, t, or l from the notation to denote that they are arbitrary. For a finite index set I, by $[\{s \xrightarrow{l_i} s_i\}_{i\in I}]$ we denote that there exist edges $\{s \xrightarrow{l_i} s_i\}_{i\in I}$ and s has no other outgoing edges.

We presuppose a finite set of actions $\mathcal{A}$ and a countable set of *internal labels* $\mathcal{L}$ such that $\mathcal{A} \cap \mathcal{L} = \emptyset$.

Definition 7.1.1 (Process graph)**.** A *process graph* r, or simply *process* r, is a directed, finite-state and finite-edge graph with root r, such that

- there exist three types of edges, or *transitions*: *action* ($\rightarrow$), *internal* ($\twoheadrightarrow$), and *probabilistic* ($\rightsquigarrow$);
- there exist three types of nodes, or *states*: *action*, *nondeterministic*, and *probabilistic*; from an action (resp. nondeterministic, probabilistic) state there can originate only action (resp. internal, probabilistic) transitions;
- the action transitions are labeled with actions from $\mathcal{A}$ such that no two action transitions with the same state of origin are labeled the same;
- the internal transitions are labeled with labels from $\mathcal{L}$ such that
  - no two internal transitions with the same state of origin are labeled the same, and
  - if $s \xrightarrow{\tau_1} \twoheadrightarrow$ and $t \xrightarrow{\tau_1} \twoheadrightarrow$, then $[\{s \xrightarrow{\tau_i} \twoheadrightarrow s_i\}_{i\in I}]$ iff $[\{t \xrightarrow{\tau_i} \twoheadrightarrow t_i\}_{i\in I}]$, i.e. if two states share a label on their outgoing internal transitions, then they have the same sets of labels on all of their outgoing transitions;
- the probabilistic transitions are labeled with scalars from $(0, 1]$, such that
  - given two states, there is at most one probabilistic transition connecting them, and
  - for each probabilistic state s, if $[\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}]$ then $\sum_{i\in I} \pi_i = 1$, i.e. the sum of the labels on the outgoing transitions equals one;

– all states are reachable from r.

The set of all process graphs is denoted by $\mathcal{G}$. A state without outgoing transitions, i.e. a *deadlock state*, is considered an action state. The deadlock process is denoted by the constant 0. Given an action state s, by s_a we denote the state (if it exists) for which $s \xrightarrow{a} s_a$; by $\mathcal{I}(s)$ we denote the set of actions $\{a_i\}_{i\in I}$ such that $[\{s \xrightarrow{a_i} s_i\}_{i\in I}]$, $\emptyset$ if s is a deadlock state. $\mathcal{I}(s)$ is called the *menu* of s. Intuitively, $\mathcal{I}(s)$ is the set of actions that process s can perform initially.

Example 7.1.2. The graphs s, u, and $\bar{s}$ in Fig. 6.1, and the graphs z, v, and r in Fig. 7.1 are process graphs.

Each process s defines a set of equations, called constraints for s. Similarly as in parametric Markov chains [46], they restrict the values that the internal labels in s can take, such that the values of the labels assigned to an internal choice form a probability distribution, i.e. they are in the interval $[0, 1]$ and sum up to 1. The constraints are defined next and will play an essential role throughout this part.

Definition 7.1.3 (Constraints)**.** Let s be a process. $\mathcal{C}(s)$, the set of *constraints* for s, is a set of linear equations over labels in $\mathcal{L}$, such that an equation is in $\mathcal{C}(s)$ if and only if it has the form $\sum_{i\in I} \tau_i = 1$ with $[\{t \xrightarrow{\tau_i} t_i\}_{i\in I}]$ for some nondeterministic state t in s. A *resolution* of $\mathcal{C}(s)$ is a function assigning values from $[0, 1]$ to the variables in $\mathcal{C}(s)$, respecting the constraints $\mathcal{C}(s)$.

We define the parametric operators $\sqcap$, $\Box$, and $\oplus$ on process graphs. The operators $\sqcap$ and $\Box$ have the same intuitive meaning as the corresponding operators in [71], i.e. $\sqcap_{i\in I}\tau_i s_i$ is (in our case labeled) internal (nondeterministic) choice among processes $\{s_i\}_{i\in I}$, while $\Box_{i\in I}\ a_i s_i$ is a choice among actions $\{a_i\}_{i\in I}$, each a_i prefixing process s_i. $\oplus_{i\in I}\pi_i s_i$ is a probabilistic choice among processes $\{s_i\}_{i\in I}$, each process s_i being chosen with probability π_i.

Definition 7.1.4. Let I be a finite index set and $\{s_i\}_{i\in I}$ be a set of process graphs. The parametric operators $\oplus$, $\sqcap$, $\Box$ on process graphs are defined as follows:

– For $\{\tau_i\}_{i\in I} \subset \mathcal{L}$ such that $\{\tau_i\}_{i\in I}$ do not appear in the process graphs $\{s_i\}_{i\in I}$, the process graph $\sqcap_{i\in I}\tau_i s_i$ is constructed by creating a new state s, a set $\{s'_i\}_{i\in I}$ of disjoint copies of the process graphs $\{s_i\}_{i\in I}$, and transitions $\{s \xrightarrow{\tau_i} s'_i\}_{i\in I}$.

– For $\{a_i\}_{i\in I} \subseteq \mathcal{A}$, the process graph $\Box_{i\in I}\, a_i s_i$ is constructed by creating a new state s, a set $\{s'_i\}_{i\in I}$ of disjoint copies of the process graphs $\{s_i\}_{i\in I}$, and transitions $\{s \xrightarrow{a_i} s'_i\}_{i\in I}$.

– For $\{\pi_i\}_{i\in I} \subset (0,1]$ such that $\sum_{i\in I} \pi_i = 1$, the process graph $\oplus_{i\in I} \pi_i s_i$ is constructed by creating a new state s, a set $\{s'_i\}_{i\in I}$ of disjoint copies of the process graphs $\{s_i\}_{i\in I}$, and transitions $\{s \overset{\pi_i}{\rightsquigarrow} s'_i\}_{i\in I}$.

We omit the operator sign $\oplus, \Box$ or $\sqcap$ if there is only one operand. We also write a rather than $a0$.

Definition 7.1.5. A *finite* process graph is a process graph in which only finite paths exist. A *finite process tree* is a finite process graph that has a tree form. A *divergence-free* process graph is a process graph without infinite sequences of probabilistic or internal transitions.

To simplify the technical framework, throughout this part we assume that processes are divergence-free.

7.2 Unfolding and coherent labeling

In this section we define how to unfold a process graph up to a finite length and relabel its internal transitions, such that two nondeterministic states have the same label sets only if they represent the same instance of an internal choice. Then, we define in a straightforward way how the internal nondeterminism in a process is resolved, by assigning probability distributions to the internal choices.

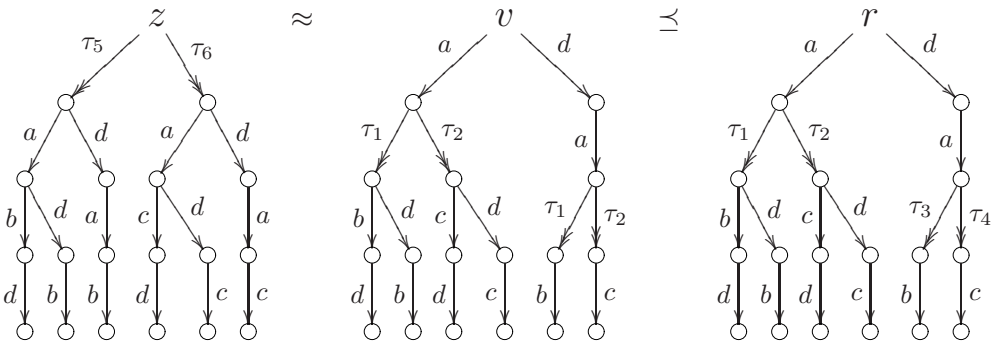

Figure 7.1: Several processes and relations between them; processes z and v are obtained by interleaving $\tau_5(ab) \sqcap \tau_6(ac)$ and $a(\tau_1 b \sqcap \tau_2 c)$, respectively, with action d.

The idea behind labeling the internal transitions is to be able to identify them in a parallel context. In this way, if the same instance of an internal choice appears in different futures, we ensure that it will be resolved in the same way in every future. For example, as already discussed, the two internal choices in $x\|y$ actually represent the same choice and thus in our model they shall be labeled the same (we defer the discussion for the exact labeling to the next section). An even more trivial example is process v in Fig. 7.1. Namely, it can be seen as the interleaving of process $p \equiv a(\tau_1 b \sqcap \tau_2 c)$ and process $q \equiv d$. Since there is no communication between p and q, the internal choice in p cannot be influenced by action d. Consequently, both internal choices that appear in process graph v must be resolved in the same manner, and this is ensured by the same label sets. However, consider graph s in Fig. 7.2. According to Definition 8.3.1, it is a process graph; however, note that there

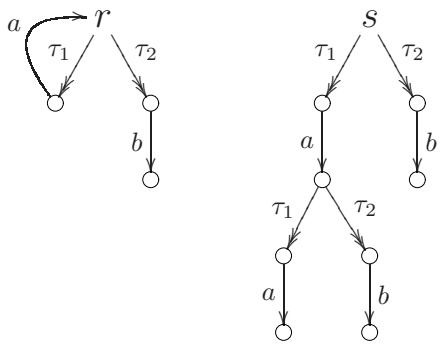

Figure 7.2: A cyclic process and a finite process

is a path with two nondeterministic states that use the label set $\{\tau_1, \tau_2\}$. Reasonably, the same instance of an internal choice cannot happen twice in the same future, and this is why we have to relabel the second internal choice with new labels. Typically, graphs like s in Fig. 7.2 result when a cyclic process graphs is unfolded: s can be seen as an unfolding of process graph r in the same figure. Clearly, in one run, all the internal choices in process r are resolved pairwise independently, due to their nondeterministic nature.

We integrate the above discussion into the following recursive function that unfolds a process up to a certain depth and relabels the internal transitions appropriately. Besides a process graph s, the function also has as arguments a set of labels L in $\mathcal{L}$ and a nonnegative integer m. The set of labels L collects the labels used in the unfolding up to the present moment and serves to ease the definition, while the integer m is the *unfolding depth*, i.e. the maximal number of observable actions that appear in a path of the

unfolded process graph. The function returns a process graph in the form of a finite tree.

Definition 7.2.1. The partial function $\mathrm{U}\colon \mathcal{G}\times 2^{\mathcal{L}}\times\mathbb{N}\mapsto\mathcal{G}$, called *unfolding*, is defined as

$$\mathrm{U}\,(s,L,m)\equiv\begin{cases}0, & \text{if } m=0 \text{ or } s\equiv 0\\ \oplus_{i\in I}\pi_i\mathrm{U}\,(s_i,L,m), & \text{if } m>0, [\{s\overset{\pi_i}{\rightsquigarrow}s_i\}_{i\in I}]\\ \Box_{i\in I}\ a_i\mathrm{U}\,(s_i,L,m-1), & \text{if } m>0, [\{s\overset{a_i}{\longrightarrow}s_i\}_{i\in I}]\\ \sqcap_{i\in I}\tau_i^{n+1}\mathrm{U}\,\big(s_i,L\cup\{\tau_i^{n+1}\}_{i\in I},m\big), & \text{if } m>0, [\{s\overset{\tau_i}{\twoheadrightarrow}s_i\}_{i\in I}],\\ & n=\max\{k|\tau_i^k\in L, i\in I\}\end{cases}$$

where each label τ_i^{n+1} is *fresh*, i.e. it hasn't been used for labeling internal transitions yet,[1] and for a given set of non-negative numbers I, by $\max I$ we denote the maximum of the numbers in I, if $I\neq\emptyset$, or 0, if $I=\emptyset$.

By unfolding a process graph, a finite process tree is obtained. The labels on the internal transitions in the obtained tree are new with respect to the labels in the original process graph. Every time a state, having outgoing internal transitions labeled with the set $\{\tau_i\}_{i\in I}$, is to be unfolded, a fresh set of labels $\{\tau_i^{n+1}\}_{i\in I}$ is used for labeling the corresponding internal transitions in the unfolded tree. This set of labels, in fact, contains implicit information, n, about the number of times a state in the original graph with the label set $\{\tau_i\}_{i\in I}$ has been unfolded up to that moment.

Example 7.2.2. The result of function $\mathrm{U}(v,\emptyset,3)$, where v is as in Fig. 7.1, is process $a(\tau_1^0(bd\ \Box\ db)\sqcap\tau_2^0(cd\ \Box\ dc))\ \Box\ da(\tau_1^0b\sqcap\tau_2^0c)$. Thus, when v is unfolded, both internal choices remain labeled with the same sets of labels.

Example 7.2.3. The result of function $\mathrm{U}(r,\emptyset,2)$, where r is as in Fig. 7.2, is process $\tau_1^0(a(\tau_1^1a\sqcap\tau_2^1b))\sqcap\tau_2^0b$. Thus, when r is unfolded up to depth 2, the two internal choices are relabeled with different sets of labels. The same holds for the unfolding of process s in Fig. 7.2.

The following example reveals that assigning labels to cyclic processes requires delicate reasoning.

Example 7.2.4. Consider the second left-most process in Fig. 7.3. By unfolding it up to depth 2, we obtain the process $\tau_1^0a(\tau_1^1a\sqcap\tau_2^1b)\sqcap\tau_2^0b(\tau_1^1a\sqcap\tau_2^1b)$. Consider now the right-most process in the same figure. By unfolding it up to depth 2, we obtain the process $\tau_1^0a(\tau_3^0a\sqcap\tau_4^0b)\sqcap\tau_2^0b(\tau_5^0a\sqcap\tau_6^0b)$.

[1] The index n in τ_i^n is not to be confused with the n^{th} power of τ_i; we would denote the latter with $(\tau_i)^n$.

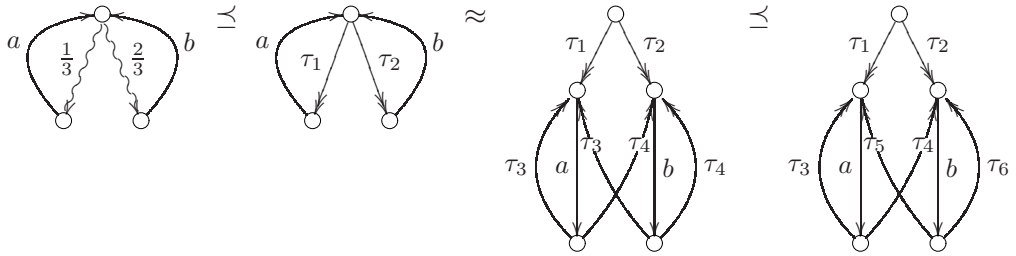

Figure 7.3: Several cyclic process graphs and relations between them.

Note that, in order to express full nondeterminism in general cyclic processes, i.e. to express that all the internal choices are resolved independently from each other, one has to switch to infinite-state graphs. However, recall that our present focus is on defining a testing semantics that gives reasonable probabilities for a process to pass a test, i.e. the present focus is on finite behaviour, where this expressivity does not play a role.

Having labeled the internal choices in an unfolded process tree in a coherent way, the internal nondeterminism can be resolved by assigning probabilities to the internal choices. Recall from Def. 7.1.3 that $\mathcal{C}(s)$ denotes the set of constraints for the labels used in process s.

Definition 7.2.5. [Resolution of a process] Let s be a process graph, $m \geq 0$, and $\bar{s} = \mathrm{U}(s, \emptyset, m)$. An *m-resolution* of s is the process graph s^m obtained when, for an arbitrary resolution λ of $\mathcal{C}(\bar{s})$, every transition $t \stackrel{\tau_i}{\twoheadrightarrow} t'$ in graph $\bar{s}$ is replaced by $t \stackrel{\lambda(\tau_i)}{\rightsquigarrow} t'$, if $\lambda(\tau_i) \neq 0$, or erased, otherwise.

Thus, an m-resolution of a process s is the process that results when the labels on the internal transitions in the unfolding of s up to depth m have been replaced by probabilities. Note that, speaking in terms of schedulers, we employ randomized schedulers [23, 94, 97], which assign arbitrary probability distributions to the internal transitions, rather than the deterministic schedulers considered in the introduction, that assign only the trivial probability distributions. This is because the former schedulers are more powerful, and with them, probabilistic choice can be treated as a special type of internal choice.

Remark The unfolding function of Def. 7.2.1 takes into consideration that a cyclic process may be a parallel composition of simpler processes; with this reasoning, as we saw, infinite-state graphs are necessary in order to model infinite processes where all the choices are resolved pairwise independently.

Another way of defining the unfolding function is to assign different values to all the internal labels of the process. This way, finite state graphs suffice. Clearly, a compromise has to be made in this case, too: we have to assume that the process-argument of the unfolding function is not obtained as a parallel composition. We leave the details of this solution for the future.

7.3 Testing semantics

In this section we define our testing semantics, and a testing preorder relation on processes based on it. More precisely, we define how labels are assigned to the internal transitions that result when a process is tested, or in other words, a synchronization operator for a process and a test. Then, we define how the parametric probability with which a process passes a test is calculated, simply by treating the internal choices in the synchronization as parametric probabilistic choices. Given a value-assignment of the parametric probabilities, we obtain a probability with which a process can pass a test. Based on the parametric probabilities, we define the testing preorder relation on processes.

7.3.1 Synchronization

We define the synchronization of a process and a test in our model. A *test* T, as usual, is a finite process tree, such that for a symbol $\omega \notin \mathcal{A}$, there may exist transitions $s \xrightarrow{\omega}$ for some states s in T, denoting success.[2] We assume that all the labels on internal transitions in a test belong to the set $\{\tau_i\}_{i\in I}$ for some index set I. Additionally, we assume that all the labels on the internal transitions in a test are distinct (we justify this assumption below).

In the previous section we defined how the internal nondeterminism in a process is labeled in a coherent way and resolved. Recall that submitting a process to a test means synchronizing both of them on all common actions, except on the success-reporting action ω. Therefore, additional nondeterminism arises when a process is being tested. First, there is the nondeterminism with respect to the action on which the process and the test synchronize, if there are multiple candidate-actions for synchronization at one moment, i.e. *synchronization nondeterminism*, and, second, there is the internal nondeterminism of the test. In order to determine how the synchronized transitions should be labeled, let us recall Example 6.0.1 and the synchronization $s\|u$ of the coin-flipping machine s and user u. Clearly, in order to avoid the

[2]It has been shown that infinite tests do not increase the distinguishing power [77, 98], because infinite paths cannot report success.

problem with overestimation of probabilities, we must ensure that both internal choices in $s \| u$ are resolved in the same way. Thus, the resolution of the synchronization nondeterminism should not depend on the probabilistic (or, for that matter, the internal) transitions of the process or the test. Moreover, note that the state in which the process or the test, or for that matter their synchronization, resides at the moment, should not play a role in the resolution of synchronization nondeterminism. In fact, in order to keep the probability with which the user guesses the outcome of coin-flipping to $\frac{1}{2}$, the resolution of synchronization nondeterminism should be based at most on the menu that the machine offers, or, in other words, on the actions that are candidates for synchronization. As the process and the test proceed interacting, the entity that resolves synchronization nondeterminism (in our example the user) may actually remember the history of synchronization and take it into account when making the current choice. To conclude, speaking in terms of labels, each label on a synchronized transition should represent exactly the set of actions-candidates for synchronization at that moment, the actual action on which the current synchronization happens, and the history of synchronization.

Regarding the test, recall that we assume that all the labels on internal transitions that it contains are distinct. In other words, for simplicity, we restrict to the subset of tests that are most powerful, as they have "full control" over their internal choices. In fact, if two processes are not distinguished by this subset of tests, then they should not be distinguished by the rest of the (less powerful) tests, too. Now, to avoid underestimation of the probabilities with which success is reported, we capture the special case when the test itself resolves the synchronization nondeterminism (as in Example 6.0.1). Therefore, we bear in mind that the test can also take into account the history of resolving the synchronization nondeterminism when resolving its internal nondeterminism. For example, consider the process $\frac{1}{2}(ad \,\Box\, b) \oplus \frac{1}{2}(ac)$ and the test $a(\tau_1 d\omega \sqcap \tau_2 c\omega) \,\Box\, b\omega$. When testing the process, the test can choose transition τ_1 if action a was synchronized out of the candidate-actions a and b, and the transition τ_2 if a was the only candidate for synchronization. In this way, the test can resolve its internal choices such that success is always reported. Thus, when an internal transition originating from the test is labeled in the synchronization of the process and the test, the label includes the history of synchronization.

We now formalize the above discussions, i.e. we define the synchronization operator for a process and a test. Let $\mathcal{G}_{\not\rightarrow} \subset \mathcal{G}$ be the set of all process graphs that do not contain internal transitions, i.e. the set of *deterministic processes*. In the following definition we assume that the process is deterministic, and later we make a remark about the general case. We presuppose a special

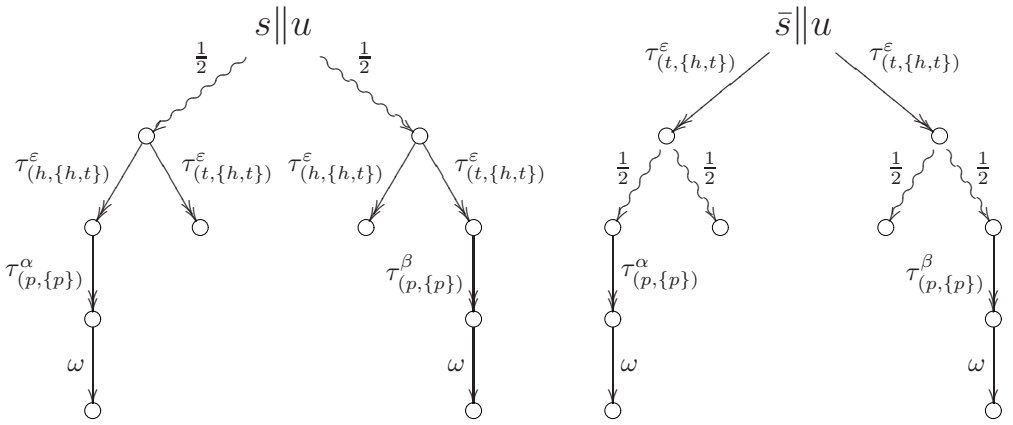

Figure 7.4: The synchronization of the coin-flipping machine s and the guessing user u in Fig. 6.1. $\alpha = \tau^{\varepsilon}_{(h,\{h,t\})}$, $\beta = \tau^{\varepsilon}_{(t,\{h,t\})}$.

label ε that cannot appear in any process or test. The synchronous parallel composition $\|_l$ is parametrized by a label $l \in \mathcal{L}$, equal to the last label created for an internal transition resulting from synchronization nondeterminism, up to the present moment. Thus, in order to pass the history of synchronization to a newly created label l', it is enough to incorporate l in l' as a superscript.

Definition 7.3.1 (Synchronization)**.** The synchronization $s\|_l T$ of a deterministic process s and a test T is defined as

$$s\|_l T = \begin{cases} \omega, & \text{if } T \xrightarrow{\omega} \\ \oplus_{i\in I}\pi_i(s_i\|_l T), & \text{if } [\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}],\ T \not\xrightarrow{\omega} \\ \oplus_{i\in I}\pi_i\,(s\|_l T_i)\,, & \text{if } [\{T \overset{\pi_i}{\rightsquigarrow} T_i\}_{i\in I}],\ s \not\rightsquigarrow \\ \sqcap_{i\in I}\tau^l_i\,(s\|_l T_i), & \text{if } [\{T \xrightarrow{\tau_i} T_i\}_{i\in I}],\ s \not\rightsquigarrow \\ \sqcap_{a\in K}\tau^l_{(a,K)}\left(s_a\|_{\tau^l_{(a,K)}} T_a\right), & \text{for } K = \mathcal{I}\,(s) \cap \mathcal{I}\,(T) \neq \emptyset,\ T \not\xrightarrow{\omega} \\ 0, & \text{otherwise.} \end{cases}$$

If $l = \varepsilon$, then we write $s\|T$ rather than $s\|_\varepsilon\, T$.

Example 7.3.2. Figure 7.4 gives the synchronization $s\|u$ of process s and test u, and the synchronization $\bar{s}\|u$ of process $\bar{s}$ and test u from Fig. 6.1. Note how both internal choices in $s\|u$, that result from the options to synchronize on h or on t, are labeled with the same set of labels. Moreover, this set of labels is also used for labeling the corresponding internal choice in $\bar{s}\|u$.

Example 7.3.3. Figure 7.5 gives the synchronization $x\|y$ of process x and test y and the synchronization $\bar{x}\|y$ of process $\bar{x}$ and test y from Fig. 6.2,

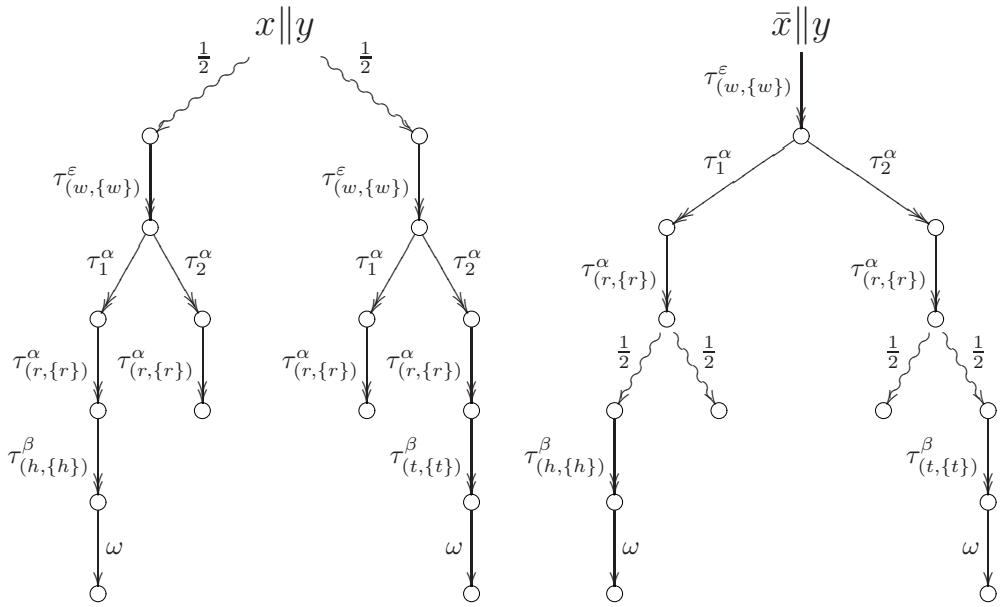

Figure 7.5: Synchronized players x and y and players $\bar{x}$ and y in Fig. 6.2. $\alpha = \tau^{\varepsilon}_{(w,\{w\})}$, $\beta = \tau^{\alpha}_{(r,\{r\})}$.

where test y in our model has been modeled as $w(\tau_1 rh\omega \sqcap \tau_2 rt\omega)$. Note how both internal choices in $x \| y$ are labeled with the same sets of labels, and the same set of labels has been used for the internal choice in $\bar{x} \| y$. In Figure 7.6 the synchronization $x' \| y'$ of process x' and test y' from Fig. 6.3 is given, where test y' in our model has been modeled as $a((\tau_1 rh\omega) \sqcap (\tau_2 rt\omega)) \,\Box\, b((\tau_3 rh\omega) \sqcap (\tau_4 rt\omega))$ (thus satisfying our requirements that the test is a process tree with all the labels distinct).

Remark Note that the operators defined in definitions 7.2.1 and 7.3.1 can be merged into one operator, yielding the synchronization of an arbitrary process with a test. We separated the definitions for clarity and because Def. 7.2.1 is also needed in Sec. 7.4.

7.3.2 The result of testing

Having labeled all the internal transitions in the synchronization of a deterministic process and a test, we can define the parametric probability with which a deterministic process s passes a test T, i.e. with which action ω is reported in $s \| T$. Namely, by treating every label set as a parametric probability distribution, the probability to report ω in the synchronization is a

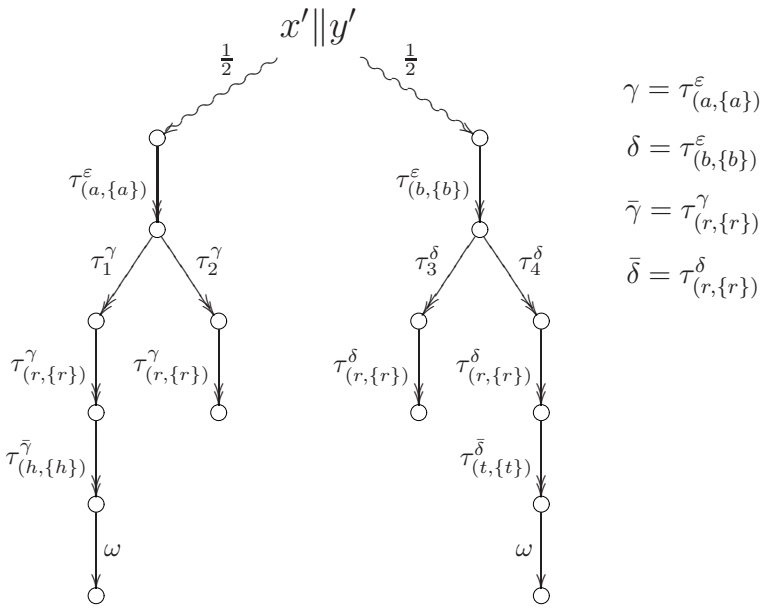

Figure 7.6: Synchronized players x' and y' in Fig. 6.3.

polynomial over the variables in $\mathcal{L}$. For our convenience, we bypass the creation of the process graph representing the synchronization, and we calculate the resulting polynomial directly. Thus, the computation of the parametric probability with which $s\|T$ performs action ω mimics the creation of $s\|T$ itself.

Denote by $\mathcal{T}$ the set of all tests and by $\mathbb{P}$ the set of all polynomials with variable names in the label set $\mathcal{L}$.

Definition 7.3.4 (Result of testing)**.** The partial function $\mathcal{R}\colon \mathcal{G}_{\not\to}\times\mathcal{T}\times\mathcal{L}\mapsto \mathbb{P}$ is defined as

$$\mathcal{R}\,(s,T,l)=\begin{cases}1, & \text{if } T\xrightarrow{\omega}\\ \sum_{i\in I}\pi_i\cdot\mathcal{R}\,(s_i,T,l), & \text{if } [\{s\overset{\pi_i}{\rightsquigarrow}s_i\}_{i\in I}],\ T\not\xrightarrow{\omega}\\ \sum_{i\in I}\pi_i\cdot\mathcal{R}\,(s,T_i,l), & \text{if } [\{T\overset{\pi_i}{\rightsquigarrow}T_i\}_{i\in I}],\ s\not\xrightarrow{\tau}\\ \sum_{i\in I}\tau_i^l\cdot\mathcal{R}\,(s,T_i,l), & \text{if } [\{T\overset{\tau_i}{\twoheadrightarrow}T_i\}_{i\in I}],\ s\not\xrightarrow{\tau}\\ \sum_{a\in K}\tau^l_{(a,K)}\mathcal{R}\left(s_a,T_a,\tau^l_{(a,K)}\right) & \text{for } K{=}\mathcal{I}\,(s)\cap\mathcal{I}\,(T)\neq\emptyset,\ T\not\xrightarrow{\omega}\\ 0, & \text{otherwise.}\end{cases}$$

If $l=\varepsilon$, then we shall write $\mathcal{R}(s,T)$ rather than $\mathcal{R}(s,T,\varepsilon)$, and we call $\mathcal{R}(s,T)$ the *result* of testing s with T. Given a resolution λ of $\mathcal{C}(s\|T)$ for a deterministic process s and a test T, the value of the polynomial $\mathcal{R}(s,T)$ for

the values of its variables given by λ is denoted by $\Pr(s, T, \lambda)$. Intuitively, $\Pr(s, T, \lambda)$ is the probability with which the deterministic process s passes test T, given that the synchronization nondeterminism and the internal nondeterminism of T when testing process s are resolved by λ.

The following proposition says that if two processes yield the same result when tested by a certain test, then they also yield the same constraints when synchronized with the test. In this case, for every resolution of the constraints, the two processes pass the test with the same probability.

Proposition 7.3.5. *Let s and t be two deterministic processes and T be a test, such that $\mathcal{R}(s, T) = \mathcal{R}(t, T)$. Then, $\mathcal{C}(s\|T) = \mathcal{C}(t\|T)$, and given an arbitrary resolution λ of $\mathcal{C}(s\|T)$, $\Pr(s, T, \lambda) = \Pr(t, T, \lambda)$.*

Proof. (Sketch) From $\mathcal{R}(s, T) = \mathcal{R}(t, T)$, we have that $s\|T$ and $t\|T$ use the same set of labels L for their internal transitions. Moreover, by Def. 7.3.4, L can be partitioned into sets $\{L_i\}_{i\in J}$, such that for every internal choice in $s\|T$ there is a set L_j containing the labels assigned to that internal choice, and the same for the internal choices in $t\|T$. Thus, by Def. 7.1.3, we obtain that $\mathcal{C}(s\|T) = \bigcup_{i\in J} \left\{\sum_{l\in L_i} l = 1\right\} = \mathcal{C}(t\|T)$. The proof of the second part is trivial. □

Example 7.3.6. Consider processes s and $\bar{s}$, test u in Fig. 6.1, and the synchronizations $s\|u$ and $\bar{s}\|\bar{u}$ in Fig. 7.4. It is easy to derive that $\mathcal{R}(s, u) = \mathcal{R}(\bar{s}, u)$, and therefore, by Proposition 7.3.5, $\mathcal{C}(s\|u) = \mathcal{C}(\bar{s}\|u)$. Moreover, for every resolution λ of $\mathcal{C}(s\|u)$, we have $\Pr(s, u, \lambda) = \Pr(\bar{s}, u, \lambda) = \frac{1}{2}$. That is, no matter how the internal nondeterminism in the synchronization of process s and test u is resolved, the probability with which s passes u is $\frac{1}{2}$, and the same holds for process $\bar{s}$ and test u. In other words, the user guesses the outcome of the coin-flipping in either machine with probability $\frac{1}{2}$.

Example 7.3.7. Consider processes x and $\bar{x}$ and test y in Fig. 6.2 and the synchronizations $x\|y$ and $\bar{x}\|y$ in Fig. 7.5. We have $\mathcal{R}(x, y) = \mathcal{R}(\bar{x}, y)$ and therefore, by Proposition 7.3.5, $\mathcal{C}(x\|y) = \mathcal{C}(\bar{x}\|y)$. Moreover, for every resolution λ of $\mathcal{C}(x\|y)$, we have $\Pr(x, y, \lambda) = \Pr(\bar{x}, y, \lambda) = \frac{1}{2}$. That is, no matter how y resolves its internal nondeterminism when synchronized with x, the probability with which success is reported is $\frac{1}{2}$. In other words, no matter how player y makes his guess, it will coincide with the outcome of coin-flipping with probability $\frac{1}{2}$. The same holds for process $\bar{x}$ and test y. On the other hand, consider processes x' and y' in Fig. 6.3, and their synchronization $x'\|y'$ in Fig. 7.6. There exists a resolution λ of $\mathcal{C}(x'\|y')$, assigning value 0 to τ_2^γ and τ_3^δ and value 1 to all the other labels, such that $\Pr(x', y', \lambda) = 1$. In other words, player y' can always guess the right outcome of the coin-flipping.

7.3.3 Testing preorder

In the classical sense of probabilistic testing [42, 44, 74, 90, 98, 108], in order for process s to implement process t, it is required that for every test T and every resolution of the nondeterminism in $s\|T$, there exists a resolution of the nondeterminism in $t\|T$ that can mimic the probability to report success in a certain way. In our setting, for the reasons explained in Chapter 6, we move from the "compose-and-schedule" approach to "schedule-and-compose" (see also [32] for this term). That is, in order for process s to implement process t, we require that for every test T, no matter how s resolves its internal nondeterminism, t can resolve its internal nondeterminism such that T cannot distinguish between s and t.

So far, we can anticipate that a good ground on which to compare two deterministic processes s and t when tested with test T is the function $\mathcal{R}(x, T)$. Namely, assume that $\mathcal{R}(s, T) = \mathcal{R}(t, T)$. Then, by Proposition 7.3.5, $\mathcal{C}(s\|T) = \mathcal{C}(t\|T)$, i.e. the same internal label sets appear in both $s\|T$ and $t\|T$. This means that the same options for resolving the synchronization nondeterminism and the internal nondeterminism arise when either s or t is tested by T, that is, the test cannot distinguish between s and t based on the actions that s or t offer or have offered for synchronization. By Proposition 7.3.5, from $\mathcal{R}(s, T) = \mathcal{R}(t, T)$ we also have that for an arbitrary resolution λ of $\mathcal{C}(s\|T)$ (and therefore of $\mathcal{C}(t\|T)$) it holds that $\Pr(s, T, \lambda) = \Pr(t, T, \lambda)$. That is, for every resolution of the nondeterminism resulting from testing, the probabilities to report success coincide for both s and t. In other words, even if the test repeatedly tests s and t, it cannot make a difference between s and t based on the observed frequency with which success is reported for a particular resolution of the synchronization nondeterminism and the internal nondeterminism in the test. Finally, note that the polynomial $\mathcal{R}(x, T)$ is "insensitive" to the exact moments in time at which x makes its probabilistic choices, i.e. $\mathcal{R}(s, T)$ does not reveal to T the internal structure of x. To summarize, the polynomial $\mathcal{R}(x, T)$ contains the exact information that test T can exploit in order to make a difference between two processes.

The above discussion is formalized in the following definitions, presenting the testing preorder relation and the induced equivalence. Given a test T, by length(T) we denote the maximal number of observable actions which T can perform before performing the success action ω. Recall from Def. 7.2.5 that an m-resolution of a process is obtained when it has been unfolded up to length m and the internal nondeterminism has been resolved.

Definition 7.3.8 (Testing preorder)**.** Let s and t be two processes. s *implements* t, denoted $s \preceq_{\mathsf{T}} t$, iff for every natural $m \geq 0$, for every test T

with length$(T) = m$ and for every m-resolution s^m of s, there exists an m-resolution t^m of t such that $\mathcal{R}(s^m, T) = \mathcal{R}(t^m, T)$.

Definition 7.3.9 (Testing equivalence)**.** Processes s and t are *testing-equivalent*, denoted $s \approx_{\mathsf{T}} t$, iff $s \preceq_{\mathsf{T}} t$ and $t \preceq_{\mathsf{T}} s$.

Remark Recall that our testing semantics allows for scenarios where the tested process is a machine with menus and the test is a user which resolves the synchronization nondeterminism. In this case the test has the power to see the history of actions-candidates for synchronization, rather than only the history of performed actions (i.e. its local history). In other scenarios, though (considered in the next chapter), a test would resolve its internal nondeterminism the same way the process does, by looking only into its local history. In this case, there is no need to change the labels of the internal transitions in a test while testing as it is done in Def. 7.3.1. It is important, however, that the way a test resolves its internal nondeterminism has no influence on the testing preorder relation, as we will see in Section 7.5. It rather only influences the obtained maximal/minimal probabilities to report success: the more power a test has, the higher/lower the obtained maximal/minimal probability (over all resolutions of the nondeterminism).

7.4 Probabilistic ready-trace preorder

In this section we define our probabilistic ready-trace preorder relation on processes. The relation is based on the ability of a process to mimic the probabilistic behaviour of another process under an arbitrary resolution of the internal nondeterminism of the latter. By mimicking the probabilistic behaviour, we mean matching the probability of an arbitrary ready-trace, i.e. a sequence of action-menus and performed actions. As the probability to observe a ready-trace is conditioned upon the actions that are actually performed, we employ conditional probabilities of the ready-traces. For these reasons, we exploit the Bayesian definition of probability, defined next, in which the probability is naturally conditioned, rather than the measure-theoretic definition.

7.4.1 Bayesian probability

The following definitions are taken from [81]. We consider a sample space, Ω, consisting of points called *elementary events*. Selection of a particular $a \in \Omega$ is referred to as "a has occurred". An *event* $A \subseteq \Omega$ is a set of elementary events. $A, B, C, \ldots$ range over events. An event A *has occurred* iff a has

occurred for some $a \in A$. Let $A_1, A_2, \ldots$ be a sequence of events and C be an event. The members of the sequence are *exclusive given* C, if, whenever C has occurred, no two of them can occur together, that is, if $A_i \cap A_j \cap C = \emptyset$ whenever $i \neq j$. C is called a *conditioning* event. If the conditioning event is Ω, then "given Ω" is omitted.

For certain pairs of events A and B, a real number $P(A|B)$ is defined and called the *probability* of A given B. These numbers satisfy the following axioms:

Ax1: $0 \leq P(A|B) \leq 1$ and $P(A|A) = 1$.

Ax2: If the events in $\{A_i\}_{i=1}^{\infty}$ are exclusive given B, then

$$P\left(\bigcup_{i=1}^{\infty} A_i \mid B\right) = \sum_{i=1}^{\infty} P(A_i|B).$$

Ax3: $P(C|A \cap B) \cdot P(A|B) = P(A \cap C|B)$.

For $P(A|\Omega)$ we simply write $P(A)$.

7.4.2 The preorder relation $\preceq_{\mathsf{RT}}$

We define a ready trace, the conditional probability of a ready trace, and the preorder relation on processes based on the conditional probabilities.

Definition 7.4.1 (Ready-trace)**.** A *ready-trace of length* n is a sequence $(M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$, where $M_i \subseteq \mathcal{A}$ for all $i \in \{1, 2, \ldots, n\}$ and $a_i \in M_i$ for all $i \in \{1, 2, \ldots, n-1\}$.

We assume that an observer is able to see the actions that the process performs, together with the menus out of which actions are chosen. Intuitively, a ready-trace $(M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$ can be observed if the initial menu is M_1, then action $a_1 \in M_1$ is performed, then the next menu is M_2, then action $a_2 \in M_2$ is performed and so on, until the observation ends at a point when the menu is M_n.

Next, given a deterministic finite process s, we define process $s_{(M,a)}$, that is the process that s becomes, assuming that menu M was offered to s and action a was performed. For example, for process s in Fig. 6.1, $s_{(\{h,t\},h)} = \frac{1}{2}p \oplus \frac{1}{2}0$. For a state s, we write shortly $s \overset{\pi}{\curvearrowright} s_n$ with s_n being an action state, rather than $s \overset{\pi_1}{\rightsquigarrow} s_1 \overset{\pi_2}{\rightsquigarrow} s_2 \ldots \overset{\pi_n}{\rightsquigarrow} s_n$ for $\pi = \pi_1\pi_2 \cdots \pi_n$.

Definition 7.4.2. Let s be a finite deterministic process. Let $M \subseteq \mathcal{A}$, $a \in M$. The process graph $s_{(M,a)}$ is obtained from s in the following way:

– if $\mathcal{I}(s) = M$ then $s_{(M,a)} \equiv s_a$;

– if $\{s_i\}_{i\in I} \neq \emptyset$ are all the process graphs such that $\mathcal{I}(s_i) = M$ and $s \overset{\pi_i}{\curvearrowright} s_i$ for $i \in I$, then

$$s_{(M,a)} \equiv \oplus_{i\in I} \frac{\pi_i}{\pi} s_{ia}, \text{ for } \pi = \sum_{i\in I} \pi_i.$$

– in any other case, $s_{(M,a)}$ is undefined.

Next, for a finite deterministic process s and a ready-trace $(M_1, a_1, \ldots, M_{n-1}, a_{n-1}, M_n)$, we define the conditional probability to observe menu M_n in s, given that previously the sequence $M_1, a_1, \ldots, M_{n-1}, a_{n-1}$ was observed. These conditional probabilities are essential for the definition of the ready-trace preorder relation, presented afterwards.

Definition 7.4.3. Let $(M_1, a_1, \ldots, M_{n-1}, a_{n-1}, M_n)$ be a ready-trace of length n and s be a finite deterministic process. Functions $P^1_s(M)$ and $P^n_s(M_n|M_1, a_1, \ldots M_{n-1}, a_{n-1})$ (for $n > 1$) are defined in the following way:

$$P^1_s(M) = \begin{cases} \sum_{i\in I} \pi_i P^1_{s_i}(M) & \text{if } [\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}], \\ 1 & \text{if } \mathcal{I}(s) = M, \\ 0 & \text{otherwise.} \end{cases}$$

$$P^2_s(M_2|M_1, a_1) = \begin{cases} P^1_{s_{(M_1,a_1)}}(M_2) & \text{if } P^1_s(M_1) > 0, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$P^n_s(M_n|M_1,a_1,...,M_{n-1},a_{n-1}) = \begin{cases} P^{n-1}_{s_{(M_1,a_1)}}(M_n|M_2,a_2,\ldots,a_{n-1}) & \text{if } P^1_s(M_1)>0, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Let the sample space consist of all the subsets of $\mathcal{A}$ and let s be a finite deterministic process. Function $P^1_s(M)$ can be interpreted as the probability that menu M is observed when process s starts executing. Let the sample space consist of all the ready-traces of length n. Function $P^n_s(M_n|M_1, a_1, \ldots M_{n-1}, a_{n-1})$ can be interpreted as the probability of the event $\{(M_1, a_1, \ldots, M_{n-1}, a_{n-1}, M_n)\}$, given the event $\{(M_1, a_1, \ldots M_{n-1}, a_{n-1}, X) \mid X \subseteq \mathcal{A}\}$, when observing ready-traces of process s. It can be shown that these probabilities are well defined, i.e. they satisfy axioms Ax1-Ax3 from Subsection 7.4.1.

Example 7.4.4. Consider process $p|||d$ in Fig. 6.4. We have:

$$P^1_{p|||d}(\{a,d\}) = 1,$$
$$P^2_{p|||d}(\{b,d\}|\{a,d\},a) = \frac{1}{3},$$
$$P^3_{p|||d}(\{d\}|\{a,d\},a,\{b,d\},b) = 1.$$

Definition 7.4.5 (Ready-trace preorder)**.** Let s and t be two processes. We say s *implements* t *w.r.t. ready-traces* (notation $s \preceq_{\mathsf{RT}} t$) if and only if for every $m \geq 0$ and every m-resolution $\bar{s}$ of s, there exists an m-resolution $\bar{t}$ of t such that for all $k \leq m$ and for all ready-traces $(M_1, a_1, \ldots, M_k)$,

- $P^1_{\bar{s}}(M_1) = P^1_{\bar{t}}(M_1)$ and
- if $k > 1$, then $P^k_{\bar{s}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ is defined if and only if $P^k_{\bar{t}}(M_k|M_1, a_1, \ldots, M_{k-1}, a_{k-1})$ is defined, and, in case they are both defined, they are equal.

Informally, a process s implements a process t if and only if for every m-resolution $\bar{s}$ of the nondeterminism in s, there is an m-resolution $\bar{t}$ of the nondeterminism in t, such that for every ready-trace $(M_1, a_1, \ldots, M_k)$ of length $k \leq m$, the probability to observe M_k, given that previously the sequence $M_1, a_1, \ldots M_{k-1}, a_{k-1}$ was observed, is defined at the same time for both $\bar{s}$ and $\bar{t}$, and, moreover, in case both probabilities are defined, they coincide. In general, process s implements process t iff s contains "less" internal nondeterminism than process t.

Definition 7.4.6 (Ready-trace equivalence)**.** Let s and t be two processes. s and t are ready-trace-equivalent, denoted by $s \approx_{\mathsf{RT}} t$, iff $s \preceq_{\mathsf{RT}} t$ and $t \preceq_{\mathsf{RT}} s$.

Examples Processes s and $\bar{s}$ in Fig.6.1 are ready-trace equivalent, and the same holds for the process pairs x and $\bar{x}$ in Fig. 6.2, z and v in Fig. 6.4, and $s\|u$ and $\bar{s}\|u$ in Fig 7.4. Processes z and v in Fig. 7.1 are ready-trace equivalent and they both implement process r in the same figure. Recall that processes z and v can be actually seen as an interleaving of processes $\tau_5 ab \sqcap \tau_6 ac$, resp. $a(\tau_1 b \sqcap \tau_2 c)$, none of which can recognize action d, with action d, while process r has "full control" over its nondeterminism. Fig. 7.3 presents relations between several cyclic processes.

7.5 The two preorders coincide

We show that the testing preorder relation $\preceq_{\mathsf{T}}$ and the ready-trace preorder relation $\preceq_{\mathsf{RT}}$ coincide. In addition, we show that deterministic tests, i.e. tests without internal transitions, suffice for comparison of processes.

Theorem 7.5.1. *Let s and t be two processes. If $s \preceq_{\mathsf{RT}} t$, then $s \preceq_{\mathsf{T}} t$.*

Proof. Assume that $s \not\preceq_{\mathsf{T}} t$. Then, by Def. 7.3.8 there exists a test T with $\text{length}(T) = m$, for some $m \geq 0$, such that there exists an m-resolution $\bar{s}$ of s, such that for every m-resolution $\bar{t}$ of t, it holds that $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. It is enough to show that for $\bar{s}$ and for every m-resolution $\bar{t}$ of t,

- there exists a menu $M \subseteq \mathcal{A}$, such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, or
- for some $k \leq m$ there exists a ready trace $(M_1, a_1, \ldots M_k)$, such that $P^k_{\bar{s}}(M_k | M_1, a_1, \ldots M_{k-1}, a_{k-1})$ and $P^k_{\bar{t}}(M_k | M_1, a_1, \ldots M_{k-1}, a_{k-1})$ are not defined at the same time, or they are both defined, but different.

We show this by induction on the structure of T. We assume that in the states in which T can perform an ω action, no other actions can be performed. Formally, if for some state u it holds that $u \xrightarrow{\omega}$, then $\mathcal{I}(u) = \{\omega\}$. We can assume this without loss of generality because the other actions, if they exist, do not change the power of the test (see Def. 7.3.4), i.e. the test immediately reports ω each time it is able to.

Suppose that the test can perform at most one transition before performing ω. Suppose first that there exist transitions $[\{T \overset{\tau_i}{\twoheadrightarrow} T_i\}_{i \in I}]$ such that every T_i for $i \in I$ can perform ω or deadlock. From Def. 7.3.4 it follows that T is not able to make a difference between s and t. Similarly if $[\{T \overset{\pi_i}{\rightsquigarrow} T_i\}_{i \in I}]$. Therefore, suppose that $[\{T \xrightarrow{a_i} T_i\}_{i \in I}]$ and every T_i for $i \in I$ can perform ω or deadlock. Let $\bar{t}$ be an m-resolution of t. By assumption, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. Assume that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu $M \subseteq \mathcal{A}$. Let $\{M_i\}_{i \in I}$ be all the menus such that $P^1_{\bar{s}}(M_i) = P^1_{\bar{t}}(M_i) = \pi_i > 0$ and $\mathcal{I}(T) \cap M_i \neq \emptyset$. Denote $\mathcal{I}(T) \cap M_i$ by M_i^T. From Def. 7.3.4 we have

$$\mathcal{R}(\bar{s}, T) = \sum_{i \in I} \pi_i \sum_{a \in M_i^T} \tau^{\varepsilon}_{\left(a, M_i^T\right)} \mathcal{R}\left(\bar{s}_{(M_i, a)},\ T_a,\ \tau^{\varepsilon}_{\left(a, M_i^T\right)}\right), \tag{7.1}$$

$$\mathcal{R}(\bar{t}, T) = \sum_{i \in I} \pi_i \sum_{a \in M_i^T} \tau^{\varepsilon}_{\left(a, M_i^T\right)} \mathcal{R}\left(\bar{t}_{(M_i, a)},\ T_a,\ \tau^{\varepsilon}_{\left(a, M_i^T\right)}\right). \tag{7.2}$$

Note that

$$\mathcal{R}\left(\bar{t}_{(M_i, a)},\ T_a,\ \tau^{\varepsilon}_{\left(a, M_i^T\right)}\right)$$

yields the polynomial 0 or 1 for every $i \in I$ and for every $a \in M_i^T$, depending only on whether T_a deadlocks or performs ω. Therefore, we obtain that $\mathcal{R}(\bar{s}, T) = \mathcal{R}(\bar{t}, T)$, which contradicts the assumption that $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$.

Therefore, there must exist a menu M such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$.

Suppose now that $[\{T \stackrel{\tau_i}{\twoheadrightarrow} T_i\}_{i\in I}]$, such that T_i for $i \in I$ are arbitrary tests. By assumption, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. By Def. 7.3.4, $\mathcal{R}(\bar{s}, T) = \sum_{i\in I} \tau_i^{\varepsilon} \mathcal{R}(\bar{s}, T_i)$ and $\mathcal{R}(\bar{t}, T) = \sum_{i\in I} \tau_i^{\varepsilon} \mathcal{R}(\bar{t}, T_i)$. Therefore, $\mathcal{R}(\bar{s}, T_i) \neq \mathcal{R}(\bar{t}, T_i)$ for some $i \in I$. The rest follows by the inductive assumption. The case when $[\{T \stackrel{\pi_i}{\twoheadrightarrow} T_i\}_{i\in I}]$, such that the T_i for $i \in I$ are arbitrary tests, is similar to the previous case.

Suppose now that there exist transitions $[\{T \stackrel{a_i}{\rightarrow} T_i\}_{i\in I}]$, such that each T_i is an arbitrary test. By assumption, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. If there exists a menu M such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, then we are done. Therefore, assume that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu $M \subseteq \mathcal{A}$. Let $\{M_i\}_{i\in I}$ be all the menus such that $P^1_{\bar{s}}(M_i) = P^1_{\bar{t}}(M_i) = \pi_i > 0$ and $\mathcal{I}(T) \cap M_i \neq \emptyset$. Denote $\mathcal{I}(T) \cap M_i$ by M_i^T. We obtain again that the equations (7.1) and (7.2) hold. Then, since $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$, it must be that

$$\mathcal{R}\left(\bar{s}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{\left(a,M_i^T\right)}\right) \neq \mathcal{R}\left(\bar{t}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{\left(a,M_i^T\right)}\right)$$

for some M_i and $a \in M_i$. From the last, it can be easily concluded that

$$\mathcal{R}\left(\bar{s}_{(M_i,a)},\ T_a\right) \neq \mathcal{R}\left(\bar{t}_{(M_i,a)},\ T_a\right).$$

By the inductive assumption, there exists a menu $M \subseteq \mathcal{A}$, such that

$$P^1_{\bar{s}_{(M_i,a)}}(M) \neq P^1_{\bar{t}_{(M_i,a)}}(M),$$

or there exists a ready trace $(M_1, a_1, \ldots M_k)$ for some $k < m$, such that

$$P^k_{\bar{s}_{(M_i,a)}}(M_k | M_1, a_1, \ldots M_{k-1}, a_{k-1}) \text{ and } P^k_{\bar{t}_{(M_i,a)}}(M_k | M_1, a_1, \ldots M_{k-1}, a_{k-1})$$

are not defined at the same time, or they are both defined, but different. From $P^1_{\bar{s}}(M_i) = P^1_{\bar{t}}(M_i)$ and from Definition 7.4.3 we have that in the first case $P^2_{\bar{s}}(M | M_i, a) \neq P^2_{\bar{t}}(M | M_i, a)$, while in the second case

$$P^{(k+1)}_{\bar{s}}(M_k | M_i, a, M_1, a_1, \ldots M_{k-1}, a_{k-1}) \text{ and}$$
$$P^{(k+1)}_{\bar{t}}(M_k | M_i, a, M_1, a_1, \ldots M_{k-1}, a_{k-1})$$

are not defined at the same time, or they are both defined, but different. Therefore, the proof of the theorem is complete.

□

Theorem 7.5.2. *Let s and t be two processes. If $s \preceq_{\mathsf{T}} t$, then $s \preceq_{\mathsf{RT}} t$.*

Proof. Assume that $s \not\preceq_{\mathsf{RT}} t$. Then, by Def. 7.4.5, for some $m \geq 0$ there exists an m-resolution $\bar{s}$ of s, such that for every m-resolution $\bar{t}$ of t, it holds that

- there exists $M \subseteq \mathcal{A}$ such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, or
- for some k, $1 < k \leq m$, there exists a ready trace $(M_1, a_1, \ldots M_k)$, such that $P^k_{\bar{s}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ and $P^k_{\bar{t}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ are not defined at the same time, or they are both defined, but different.

It is enough to show that there exists a test T with $\mathrm{length}(T) = m$ such that, for the given $\bar{s}$, and for an arbitrary m-resolution $\bar{t}$ of t, it holds that $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. The proof is by induction on the minimal value of m.

In the case $m = 0$ there is nothing to prove. Suppose $m = 1$. Then, $\bar{s}$ is a 1-resolution of s, such that for every 1-resolution $\bar{t}$ of t there exists a menu $M_{\bar{t}}$ with $P^1_{\bar{s}}(M_{\bar{t}}) \neq P^1_{\bar{t}}(M_{\bar{t}})$. Take the test $T = \Box_{a\in\mathcal{A}}\, a\omega$. Let $\bar{t}$ be an arbitrary 1-resolution of t. We have

$$\mathcal{R}(\bar{s}, T) = \sum_{M: P^1_{\bar{s}}(M) > 0} P^1_{\bar{s}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)},$$

$$\mathcal{R}(\bar{t}, T) = \sum_{M: P^1_{\bar{t}}(M) > 0} P^1_{\bar{t}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)}.$$

Assuming that $\mathcal{R}(\bar{s}, T) = \mathcal{R}(\bar{t}, T)$, we obtain that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu M, which contradicts our assumption that $P^1_{\bar{s}}(M_{\bar{t}}) \neq P^1_{\bar{t}}(M_{\bar{t}})$. Therefore, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$.

Assume now that $m > 1$. Let $\bar{t}$ be an arbitrary m-resolution of t. If there exists an $M \subseteq \mathcal{A}$ such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, then $T = \Box_{a\in\mathcal{A}}\, a\omega$ distinguishes between $\bar{s}$ and $\bar{t}$, similar as in the case $m = 1$. Assume that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu M. Take the test $T = T_m$, defined inductively by

$$T_1 = \Box_{a\in\mathcal{A}}\, a\omega,$$
$$T_n = \Box_{a\in\mathcal{A}}\, aT_{n-1} \text{ for every } n > 1.$$

We have

$$\mathcal{R}\left(\bar{s}, T\right) = \sum_{M: P^1_{\bar{s}}(M) > 0} P^1_{\bar{s}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)} \mathcal{R}\left(\bar{s}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right),$$

$$\mathcal{R}\left(\bar{t}, T\right) = \sum_{M: P^1_{\bar{t}}(M) > 0} P^1_{\bar{t}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)} \mathcal{R}\left(\bar{t}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right).$$

Assume that $\mathcal{R}(\bar{s}, T) = \mathcal{R}(\bar{t}, T)$. Then, for every menu M,

$$\mathcal{R}\left(\bar{s}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right) = \mathcal{R}\left(\bar{t}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right). \tag{7.3}$$

From the assumption that $P^k_{\bar{s}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ and $P^k_{\bar{t}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ are not defined at the same time, or they are both defined, but different, and from $P^1_{\bar{s}}(M_1) = P^1_{\bar{t}}(M_1)$, we have that

$$P^{k-1}_{\bar{s}_{(M_1,a_1)}}(M_k|M_2, a_2, \ldots M_{k-1}, a_{k-1}) \text{ and}$$
$$P^{k-1}_{\bar{t}_{(M_1,a_1)}}(M_k|M_2, a_2, \ldots M_{k-1}, a_{k-1})$$

are not defined at the same time, or they are both defined, but different. Then, by the inductive assumption, we have that

$$\mathcal{R}\left(\bar{s}_{(M_1,a_1)},\ T_{a_1},\ \tau^{\varepsilon}_{(a_1,M_1)}\right) \neq \mathcal{R}\left(\bar{t}_{(M_1,a_1)},\ T_{a_1},\ \tau^{\varepsilon}_{(a_1,M_1)}\right).$$

This contradicts (7.3). Therefore, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$ and the proof of the theorem is complete.

□

Corollary 7.5.3. *Let s and t be two processes. $s \preceq_{\mathsf{T}} t$ if and only if $s \preceq_{\mathsf{RT}} t$.*

The following proposition states that tests with internal transitions have no more distinguishing power than deterministic tests, i.e. tests without internal transitions. In other words, if two processes are not related by the testing preorder relation, then this can be concluded by using a certain deterministic test.

Proposition 7.5.4. *Let s and t be two processes. If $s \not\preceq_{\mathsf{T}} t$, then there exists a deterministic test T (without internal transitions) with length m, such that for some m-resolution $\bar{s}$ of s and for every m-resolution $\bar{t}$ of t, $\mathcal{R}(s, T) \neq \mathcal{R}(t, T)$.*

Proof. If $s \not\preceq_{\mathsf{T}} t$, then, by Corollary 7.5.3, $s \not\preceq_{\mathsf{RT}} t$. The rest of the proof follows the lines of the proof of Theorem 7.5.2. □

Chapter 8

A conservative probabilistic extension of CSP

In Chapter 7 we defined synchronization with hiding of two finite probabilistic processes by labeling the resulting internal transitions. The goal was to restrict the power of the schedulers that resolve the internal nondeterminism such that the probabilistic behaviour of the composed system is preserved. In order to be be able to model asynchronous systems, in this chapter we generalize the parallel composition to include an arbitrary number of processes, and to allow for action interleaving, in addition to synchronization with hiding. The labels assigned to the internal transitions resulting from parallelism again reflect the information that the components use in order to resolve nondeterminism, such that realistic estimates for the probabilistic behaviour of the system can be obtained. The parallel composition is incorporated in a probabilistic version of the process language CSP, that includes a general choice operator in addition to action, internal, and probabilistic choice, and that also includes an action priority operator. We show that the ready-trace preorder defined in Chapter 7 is a precongruence w.r.t. the operators of probabilistic CSP. We also give an axiomatic characterization of ready-trace equivalence, from which it follows that the distributivity laws for the internal choice are preserved from CSP, and no new laws regarding the interplay between the different choice operators are added.

8.1 Operators for choices and priority

In this section we present the choice and the action priority operators of our CSP-like probabilistic process language. The parallel composition is presented separately in the next section. We note that sequential composition can be included in a straightforward way in line with [71, 96].

In the rest of this part, to ease the notation, we assume that every time an index set I is used, it is finite and non-empty, and we agree that $\mathcal{A} = \{a_1, a_2, \ldots a_k\}$ every time an indexed set $\{a_i\}_{i\in I} \subseteq \mathcal{A}$ is assumed.

Syntax We call the simple probabilistic process language SP_p. The SP_p process terms are generated by the following grammar:

$$x ::= \ 0 \ \mid \ \Box_{i\in I}\ a_i x_i \ \mid \ x \,\Box\, x \ \mid \ \sqcap_{i\in I} \tau_i x_i \ \mid \oplus_{i\in I}\ \pi_i x_i \mid \ \Theta x$$

where $0 \notin \mathcal{A}$ is a new symbol, $\{a_i\}_{i\in I} \subseteq \mathcal{A}$, $\{\pi_i\}_{i\in I} \subset (0, 1]$ such that $\sum_{i\in I} \pi_i = 1$, and $\{\tau_i\}_{i\in I} \subset \mathcal{L}$ such that the labels $\{\tau_i\}_{i\in I}$ do not appear in the terms $\{x_i\}_{i\in I}$. We let $p, q, \ldots$ range over SP_p process terms.

We recall the operators 0, $\Box_{i\in I}\ a_i p_i$, $\sqcap_{i\in I} \tau_i p_i$, and $\oplus_{i\in I}\ \pi_i p_i$ from the previous chapter. The constant 0 stands for the *deadlock* process. The *external action choice* $\Box_{i\in I}\ a_i p_i$ stands for choice among the actions in $\{a_i\}_{i\in I}$ and proceeds as process p_j if action a_j is chosen and executed. We write ap (prefix) rather than $\Box\ ap$. We write a rather than $a0$. The *internal choice* $\sqcap_{i\in I} \tau_i p_i$ stands for labeled internal choice between processes $\{p_i\}_{i\in I}$. The *probabilistic choice* $\oplus_{i\in I}\ \pi_i p_i$ is process p_i with probability π_i for $i \in I$. The operator $p \,\Box\, q$ stands for a *general external choice* between processes p and q. The *priority* operator Θ [12, 13] assumes a partial order $>$ on $\mathcal{A}$. For actions a and b, we say a has higher priority than b iff $a > b$.

Semantics Table 8.1 represents the operational semantics of SP_p process terms. The rules can be applied to process terms, and the generated process graph is counted as an interpretation of the process term. The negative premises $x \not\stackrel{\tau}{\to}$ denote that x does not start with a probabilistic transition, and similar holds for the negative premises $x \not\stackrel{a}{\to}$ and $x \not\to$.[1] Rules R1 and R3 are standard. Rule R2 states that, when several processes are composed via an internal choice, newly introduced labels are assigned to the new internal transitions. This is because in our model the internal transitions are labeled, for the reasons discussed in the introduction.

[1] See the remark in Section 4.4 regarding rules with negative premises; a similar discussion applies for the meaningfulness of the rules in this part, too.

$$\text{(R1)}\ \frac{k \in I}{\Box_{i\in I}\, a_i p_i \xrightarrow{a_k} p_k} \qquad \text{(R2)}\ \frac{k \in I}{\sqcap_{i\in I}\tau_i p_i \overset{\tau_k}{\twoheadrightarrow} p_k} \qquad \text{(R3)}\ \frac{k \in I}{\oplus_{i\in I}\, \pi_i p_i \overset{\pi_k}{\rightsquigarrow} p_k}$$

$$\text{(R4)}\ \frac{p \xrightarrow{a} p',\ q \xrightarrow{a} q'}{p \Box q \xrightarrow{a} \tau_1^{\text{new}} p' \sqcap \tau_2^{\text{new}} q'} \qquad \text{(R5)}\ \frac{p \xrightarrow{a} p',\ q \not\twoheadrightarrow,\ q \not\rightsquigarrow,\ q \overset{a}{\not\twoheadrightarrow}}{p \Box q \xrightarrow{a} p',\ q \Box p \xrightarrow{a} p'}$$

$$\text{(R6)}\ \frac{p \overset{\pi}{\rightsquigarrow} p',\ q \not\twoheadrightarrow}{p \Box q \overset{\pi}{\rightsquigarrow} p' \Box q} \qquad \text{(R7)}\ \frac{p \overset{\pi}{\rightsquigarrow} p',\ q \not\twoheadrightarrow, q \not\rightsquigarrow}{q \Box p \overset{\pi}{\rightsquigarrow} q \Box p'}$$

$$\text{(R8)}\ \frac{p \overset{\tau_1}{\twoheadrightarrow} p'}{p \Box q \overset{\tau_1}{\twoheadrightarrow} p' \Box q} \qquad \text{(R9)}\ \frac{p \overset{\tau_1}{\twoheadrightarrow} p',\ q \not\twoheadrightarrow}{q \Box p \overset{\tau_1}{\twoheadrightarrow} q \Box p'}$$

$$\text{(R10)}\ \frac{p \xrightarrow{a} p',\ p \overset{b}{\not\twoheadrightarrow} \text{ for } a<b}{\Theta p \xrightarrow{a} \Theta p'} \qquad \text{(R11)}\ \frac{p \overset{\tau_1}{\twoheadrightarrow} p'}{\Theta p \overset{\tau_1}{\twoheadrightarrow} \Theta p'} \qquad \text{(R12)}\ \frac{p \overset{\pi}{\rightsquigarrow} p'}{\Theta p \overset{\pi}{\rightsquigarrow} \Theta p'}$$

Table 8.1: Operational semantics for the choice operators and the priority operator

Rule R4, similarly as in CSP [96], states that if two processes can initially perform action a, then the external choice between them can also perform action a; however, the choice on whether the first or the second process is executed afterwards is nondeterministic, i.e. internal. Note that the transitions of the internal choice are suitably labeled with newly introduced labels. Rule R5 demonstrates the priority of internal and probabilistic transitions over action transitions when bound in an external choice. This is because the environment (the external choice) is unable to prevent the internal transitions from occurring [96]. Rules R6 – R9 demonstrate the priority of an internal transition over a probabilistic transition in an external choice. This is an arbitrary technical solution, namely, we can also give priority to the probabilistic over the internal transitions. This freedom stems from the fact that the internal transitions are labeled with all the information they need in order to be resolved. In fact, since probabilistic choice is a special case of the internal choice, we argue that any ordering should lead to the same axiom set (which will become evident later). Rules R8 and R9 (resp. R6 and R7) express that, if both processes p and q can perform internal (resp. probabilistic) transitions initially, then the internal (resp. probabilistic) transitions of p happen first. This is also an arbitrary technical solution, and the freedom stems from the fact that in [96] internal choices, one per component bound in an external

choice, are all combined into one internal choice – e.g. $(a \sqcap b) \,\Box\, (c \sqcap d)$ has the same graph representation as $(a \,\Box\, c) \sqcap (a \,\Box\, d) \sqcap (b \,\Box\, c) \sqcap (b \,\Box\, d)$. Similarly, in probabilistic process algebras, probabilistic choices of the separate components are combined into one probabilistic choice in the external choice (see part I, and also [3, 68]). Actually, there is an alternative solution, namely to create a new internal choice that decides which process performs an internal or probabilistic action first when external choice between two processes is made. This solution is more complicated, since auxiliary operators would have to be introduced and would generate larger graphs. The bottom line is, the commutativity law for external choice remains valid with any solution.

Finally, the priority operator Θ forces the process to block actions that have lower priority than another action in the current menu (R10). This operator does not affect the internal and the probabilistic transitions (rules R11 and R12).

8.2 Parallel composition

In Chapter 7, for the purposes of preserving the probabilities to pass a test, we proposed a model with labeled internal transitions and we defined a synchronization operator for a process and a test, or a concurrency operator [71], with action hiding after synchronization. This operator sufficed for defining the testing preorder, for which it was shown that it can be characterized by a ready-trace preorder relation. Here, we propose an n-ary general parallel composition operator[2] that, in the fashion of CSP, allows processes to communicate on a set of actions and to interleave on the rest of the actions, and, in the fashion of CCS [85], allows action hiding after synchronization. We include action hiding after synchronization, since, as discussed in Chapter 1, it is an important modeling feature.[3] Two more types of internal nondeterminism arise when processes are composed in this way, in addition to the synchronization nondeterminism discussed in Chapter 7. First, there is the nondeterminism on whether the processes will synchronize or interleave, and second, if they interleave, then there is the nondeterminism on which process performs a particular action, if several processes can perform that action. In this section, we propose how to label the internal transitions of these new types of nondeterminism, such that the labels again reflect the information based on which the nondeterminism is resolved. In this way, we obtain realistic schedulers for the nondeterminism, and4 we also achieve compositionality for the ready-trace equivalence, as shown in Section 8.4.

[2] See [32, 48] for other n-ary parallel operators.

[3] In the conclusion to this part we discuss the general hiding operator.

In order to explain better the definition of the parallel composition, we are going to introduce it gradually, by first defining an n-ary concurrency operator, then an interleaving operator [71], and finally the general parallel composition operator.

8.2.1 Concurrency with hiding

The SP_p^c process language is obtained by extending SP_p with a parametrized, n-ary concurrency operator with action hiding after synchronization:

$$x ::= y \;\mid\; \|_n^l(x_1, x_2, \ldots x_n),$$

where y is a SP_p process term, $l \in \mathcal{L}$, $n \in \mathbb{N}^{\geq 2}$. The parameter l keeps information that is transferred to the labels of the newly created internal transitions. The operational rules for $\|_n^l$ are given in Fig. 8.1, where by

$$\text{(C1)}\quad \frac{p_i \not\stackrel{\tau}{\to} \text{ for } i \in \{1, \ldots k-1\},\ p_k \stackrel{\tau_k}{\twoheadrightarrow} p'_k}{\|_n^l(p_1, \ldots p_{k-1}, p_k, p_{k+1} \ldots p_n) \stackrel{\tau_k}{\twoheadrightarrow} \|_n^l(p_1, \ldots p_{k-1}, p'_k, p_{k+1} \ldots p_n)}$$

$$\text{(C2)}\quad \frac{p_i \not\stackrel{\tau}{\to} \not\stackrel{a}{\to} \text{ for } i \in \{1, \ldots k-1\},\ p_k \stackrel{\pi_k}{\rightsquigarrow} p'_k}{\|_n^l(p_1, \ldots p_{k-1}, p_k, p_{k+1} \ldots p_n) \stackrel{\pi_k}{\rightsquigarrow} \|_n^l(p_1, \ldots p_{k-1}, p'_k, p_{k+1} \ldots p_n)}$$

$$\text{(C3)}\quad \frac{p_i \stackrel{a}{\to} p_i^a \text{ for every } i \in \{1, \ldots n\},\ B = \cap_{i=1}^n \mathcal{I}(p_i)}{\|_n^l(p_1, \ldots p_n) \stackrel{\tau_{(a,B)}^l}{\twoheadrightarrow} \|_n^{\tau_{(a,B)}^l}(p_1^a, p_2^a, \ldots p_n^a)}$$

Figure 8.1: Operational semantics for concurrency with hiding.

$p \not\stackrel{\tau}{\to} \not\stackrel{a}{\to}$ we denote $p \not\stackrel{\tau}{\to}$ and $p \not\stackrel{a}{\to}$. Recall that $\mathcal{I}(p)$ denotes the initial menu of process p. Note that a reasoning, similar to the one that was employed in Chapter 7 and in Section 8.1, is incorporated in the definition of the present parallel operator. Namely, rules C1 and C2 demonstrate that internal transitions again have priority over probabilistic transitions, as in Table 8.1, and that which process first performs the internal (or probabilistic) transitions is irrelevant (no matter the asymmetry that is present in the rules). Note that we leave the labels on the internal transitions intact. On the one hand,

for simple processes, i.e. processes that are not obtained as compositions, this means that at most the local history of performed actions can have influence on the internal choices of the process. On the other hand, leaving the labels intact ensures that, when a complex process is synchronized with another process, the dependencies between the internal choices of the former one are preserved (for example, when process v in Fig. 7.1 is synchronized with process $ac \,\Box\, dab$).

Rule C3 generalizes the reasoning about the resolution of the synchronization nondeterminism in Chapter 7 (note that all n operands are necessary to participate in the synchronization in order for it to happen). Namely, each label on a (hidden) synchronized action carries as information the action on which synchronization happened, the set of action-candidates for synchronization at that moment, and the synchronization history. This means that the decision on which action to synchronize, when there is more than one action-candidate for synchronization, can depend at most on the set of the action-candidates for synchronization and on the synchronization history.

8.2.2 Interleaving

We extend the SP_p^c process language with a parametrized interleaving operator, such that there is no communication between the operands:

$$x ::= y \;\mid\; |||^{\psi}(x_1, x_2, \ldots x_n),$$

where y is a SP_p^c process term and ψ is a function $\psi\colon \mathcal{A} \times 2^{\{1,\ldots n\}} \mapsto \mathbb{N}$. The parameter ψ carries information that is passed to the labels of the newly created internal nondeterminism during the course of interleaving.

Note that when several process-operands can perform the same action at the same moment, the choice on which process offers the action to the environment next is nondeterministic (introducing thus *interleaving* nondeterminism). We can consider this as a race among the processes, since the processes do not communicate and thus no "intelligent" decision is being made. The question is what affects the outcome of the race, so that we can label the internal transitions with the appropriate information. Clearly, the set of processes that take part in the race is important. Then, the action itself may also play a role, as a process may have different speeds for different actions. Note that, in scenarios where each process has a certain probability to win the race on a particular action among a given set of processes, this information should suffice. However, in other scenarios, every time the same set of processes has a race on the same action, the probability for a particular process to win the race may take a different value, for example, the speed

$$\text{(I1)}\ \frac{p_i \not\stackrel{\tau}{\twoheadrightarrow} \text{ for } i \in \{1,\dots k{-}1\},\ p_k \stackrel{\tau_k}{\twoheadrightarrow} p'_k}{|||^{\psi}(p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \stackrel{\tau_k}{\twoheadrightarrow} |||^{\psi}(p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(I2)}\ \frac{p_i \not\stackrel{\tau}{\twoheadrightarrow}\not\stackrel{\tau}{\rightarrow} \text{ for } i \in \{1,\dots k{-}1\},\ p_k \stackrel{\pi_k}{\rightsquigarrow} p'_k}{|||^{\psi}(p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \stackrel{\pi_k}{\rightsquigarrow} |||^{\psi}(p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(I3)}\ \frac{p_i \stackrel{a}{\not\twoheadrightarrow}\not\stackrel{\tau}{\twoheadrightarrow}\not\stackrel{\tau}{\rightarrow} \text{ for } i \neq k,\ p_k \stackrel{a}{\rightarrow} p_k^a}{|||^{\psi}(p_1,\dots,p_{k-1},p_k,p_{k+1},\dots p_n) \stackrel{a}{\rightarrow} |||^{\psi}(p_1,\dots,p_{k-1},p_k^a,p_{k+1},\dots p_n)}$$

$$\text{(I4)}\ \frac{p_i \stackrel{a}{\rightarrow} p_i^a \text{ for all } i \in I \subseteq \{1,\dots n\},\ p_i \not\stackrel{\tau}{\twoheadrightarrow}\not\stackrel{\tau}{\rightarrow}\not\stackrel{a}{\rightarrow} \text{ for } i \notin I,\ \psi(a,I){=}m}{\begin{array}{c}|||^{\psi}(p_1,\dots p_n) \stackrel{a}{\rightarrow} \prod_{i\in I} \tau_{i,I,a}^{m+1}\,(|||^{\nu}(p_1,\dots,p_{i-1},p_i^a,p_{i+1},\dots p_n))\,,\\ \nu(a,I){=}m{+}1,\ \nu(b,J){=}\psi(b,J) \text{ for } b{\neq}a \text{ or } J{\neq}I\end{array}}$$

Figure 8.2: Operational semantics for interleaving

of an action may decrease by time. Thus, we can include the number of times that the given set of processes has raced on the particular action as information in the labels. This way we stay consistent with our reasoning in Sec. 7.2, that the consecutive internal choices are in general independent from each other. In general, note that the more information is included in the label, the more power the "scheduler" has. However, we conjecture that the exact labeling does not affect the congruence property of our preorder, as long as the labels do not include "local" information, i.e. information that is local to a process. We already discussed in the introduction that the local information should not affect the resolution of the "global" nondeterminism.

The operational rules for the interleaving operator, given in Fig. 8.2 encompass the above discussion. Rules I1 and I2 are similar to rules C1 and C2 from Fig. 8.1, and rule I3 says that if only one process can perform action a at the moment, then after performing a, the interleaving proceeds as usual. Rule R4 is more involved. It says that if several processes can perform action a at the moment, then after performing action a, there is an internal choice on which process has actually performed the action, or "won the race". Each internal transition label includes as information the actual process, the set

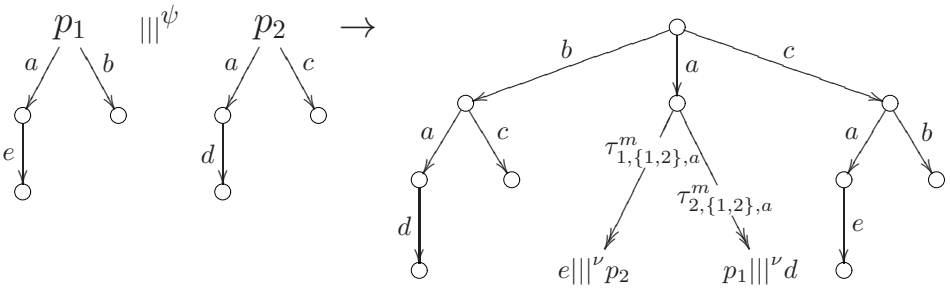

Figure 8.3: An example of interleaving. $m{=}\nu(a,\{1,2\}){=}\psi(a,\{1,2\}){+}1$, $\nu(x,I) = \psi(x,I)$ for $x{\neq}a$ or $I{\neq}\{1,2\}$.

of processes that participated in the race, the action a, and the number of times the same set of processes has raced on action a, incremented by one. The parameter ψ of the interleaving operator, which keeps track of the latter number for each pair of action and process set, is then appropriately updated.

Example 8.2.1. In Fig. 8.3 an example of interleaving of two processes is given. Note that in the result of interleaving, after action a is performed, there is an internal choice that decides from which process action a originated. The labels of the internal choice contain as information the index of the process that performed the action (1 or 2), the set $\{1,2\}$ of processes that participated in the race, the action a, and the number of times that processes p_1 and p_2 raced on action a, incremented by one, that is, m.

8.2.3 General parallel composition with hiding

The $\mathrm{CSP_p}$ process language is generated by the following grammar:

$$x ::= \Box_{i\in I}\, a_i x \;\mid\; x \,\Box\, x \;\mid\; \sqcap_{i\in I}\tau_i x \;\mid\; \oplus_{i\in I}\,\pi_i x \;\mid\; \|^l_n(x_1, x_2, \ldots x_n) \;\mid\; |||^\psi(x_1, \ldots x_n) \;\mid\; \|^{\omega,l}_{A,n}(x_1, \ldots x_k) \;\mid\; \downarrow^{\omega,l}_{A,n}(x_1, \ldots x_n) \;\mid\; \Downarrow^{\omega,l}_{A,n}(x_1, \ldots x_k)$$

where $l \in \mathcal{L}$, $A \subseteq \mathcal{A}$, $n \in \mathbb{N}^{\geq 2}$, $2 \leq k \leq n$, $\psi\colon \mathcal{A} \times 2^{\{1,\ldots n\}} \mapsto \mathbb{N}$, $\omega\colon \mathcal{A} \setminus A \times 2^{\{1,\ldots n\}} \mapsto \mathbb{N}$, and the rest is as for SP_p in Section 8.1.

As already mentioned, the n-ary parallel composition operator $\|^{\omega,l}_{A,n}(x_1, \ldots x_k)$ combines features from both the generalized parallel composition in CSP [96] and the parallel composition in CCS [85]. The operands synchronize on the actions from the set A, hiding the resulting action, while they interleave on the rest of the actions. The synchronization is n-party, i.e. n operands need to participate in order for it to happen (therefore, if $k < n$ then synchronization can not happen, but only interleaving). The operators $\downarrow^{\omega,l}_{A,n}$ and $\Downarrow^{\omega,l}_{A,n}$ are auxiliary operators that ease the definition of $\|^{\omega,l}_{A,n}$. In a

$$\text{(P1)}\ \frac{p_i \not\twoheadrightarrow \not\rightsquigarrow \ \text{ for } i \in \{1,\dots k{-}1\},\ p_k \overset{\pi_k}{\rightsquigarrow} p'_k}{\|^{\omega,l}_{A,n}(p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \overset{\pi_k}{\rightsquigarrow}\ \|^{\omega,l}_{A,n}(p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(P2)}\ \frac{p_i \not\twoheadrightarrow \ \text{ for } i \in \{1,\dots k{-}1\},\ p_k \overset{\tau_k}{\twoheadrightarrow} p'_k}{\|^{\omega,l}_{A,n}(p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \overset{\tau_k}{\twoheadrightarrow}\ \|^{\omega,l}_{A,n}(p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(P3)}\ \frac{B = \cap_{i=1}^{n}\mathcal{I}(p_i) \cap A \neq \emptyset,\ \ \cup_{i=1}^{n}\mathcal{I}(p_i) \not\subseteq A}{\|^{\omega,l}_{A,n}(p_1,\dots p_n) \overset{\tau^{l,\mathsf{syn}}_B}{\twoheadrightarrow} \downarrow^{\omega,l}_{A,n}(p_1,\dots p_n),\ \|^{\omega,l}_{A,n}(p_1,\dots p_n) \overset{\tau^{l,\mathsf{asyn}}_B}{\twoheadrightarrow} \downarrow^{\omega,l}_{A,n}(p_1,\dots p_n)}$$

Figure 8.4: Operational semantic rules for $\|^{\omega,l}_{A,n}$, part 1

composition $\downarrow^{\omega,l}_{A,n}$, the operands are forced to synchronize initially, proceeding afterwards in parallel, while in $\Downarrow^{\omega,l}_{A,n}$ the operands initially interleave and proceed in parallel afterwards. In all three operators $\|^{\omega,l}_{A,n}$, $\downarrow^{\omega,l}_{A,n}$, and $\Downarrow^{\omega,l}_{A,n}$, similarly as in $\|^{l}_{n}$ and $|||^{\psi}$, the parameters ω and l keep relevant information about the history of communication/interleaving, needed to infer labels for the internal choices that arise in the process composition.

Recall that, when processes are allowed to synchronize on a set of actions with hiding the synchronized action, and to interleave on the rest of the actions, several types of internal nondeterminism arise in the parallel composition. We have already discussed the synchronization nondeterminism and the interleaving nondeterminism in the previous subsections. However, hiding the synchronized action introduces a third type of nondeterminism, namely the processes have to "decide" whether to proceed synchronously by performing a hidden synchronized transition, or asynchronously, by performing a visible transition from one process only. The question is what affects the decision on whether to synchronize or not, so that plausible labels on the transitions of this nondeterministic choice can be assigned. Keeping in mind that all processes have to take part in the synchronization, we argue that the set of action-candidates for synchronization can influence the decision, as this is information available to all the processes at the moment of decision. With respect to the last, the history of synchronization, i.e. of action-candidates

$$\text{(P4)}\ \frac{B = \cap_{i=1}^{n} \mathcal{I}(p_i) \cap A = \emptyset}{\|_{A,n}^{\omega,l}(p_1, \ldots p_n) \overset{\tau^{\text{new}}}{\twoheadrightarrow} \Downarrow_{A,n}^{\omega,l} (p_1, \ldots p_n)}$$

$$\text{(P5)}\ \frac{B = \cap_{i=1}^{n} \mathcal{I}(p_i) \cap A \neq \emptyset, \quad \cup_{i=1}^{n} \mathcal{I}(p_i) \subseteq A}{\|_{A,n}^{\omega,l}(p_1, \ldots p_n) \overset{\tau^{\text{new}}}{\twoheadrightarrow} \downarrow_{A,n}^{\omega,l} (p_1, \ldots p_n)}$$

$$\text{(P6)}\ \frac{p_k \not\rightarrow \not\twoheadrightarrow \not\rightsquigarrow,\ p_i \rightarrow \text{ for every } i \neq k}{\|_{A,n}^{\omega,l}(p_1, \ldots p_n) \overset{\tau^{\text{new}}}{\twoheadrightarrow} \|_{A,n}^{\omega,l}(p_1, \ldots, p_{k-1}, p_{k+1}, \ldots p_n)}$$

$$\text{(P7)}\ \frac{j < n}{\|_{A,n}^{\omega,l}(p_1, \ldots p_j) \overset{\tau^{\text{new}}}{\twoheadrightarrow} \Downarrow_{A,n}^{\omega,l} (p_1, \ldots, p_j)}$$

Figure 8.5: Operational semantic rules for $\|_{A,n}^{\omega,l}$, part 2

for synchronization and the synchronized actions themselves must be also included in the labels.

The above discussion is incorporated in rule P3 in Fig. 8.4, which, together with Fig. 8.5, Fig. 8.6, and Fig. 8.7, gives the operational semantics for parallel composition and the auxiliary operators. Namely, rule P3 says that, if processes are able to synchronize (condition $B = \cap_{i=1}^{n} \mathcal{I}(p_i) \cap A \neq \emptyset$), but also to perform actions independently (condition $\cup_{i=1}^{n} \mathcal{I}(p_i) \not\subseteq A$), then there is an internal choice on whether to synchronize or not. The labels of the internal choice, $\tau_B^{l,\mathsf{syn}}$ and $\tau_B^{l,\mathsf{asyn}}$, include the set B of actions-candidates for synchronization, the history of synchronization l and the actual decision (syn or asyn). The rest of the rules in Fig. 8.4 and Fig. 8.5 are straightforward: rules P1 and P2 (Fig. 8.4), similarly as before, give priority to internal/probabilistic transitions over visible transitions, rule P4 (Fig. 8.5) says that if processes are not able to synchronize at the moment (condition $B = \cap_{i=1}^{n} \mathcal{I}(p_i) \cap A = \emptyset$), then they proceed by performing some visible action from one of the processes, while rule P5 (Fig. 8.5) says that, if processes can only synchronize at the moment, then this is what they do. Rules P6 and P7 (Fig. 8.5) say that, if one of the processes has reached deadlock, then the parallel composition can only continue asynchronously, i.e. no more

$$\text{(S1)}\ \frac{p_i \not\twoheadrightarrow\not\rightarrow \text{ for } i \in \{1,\dots k{-}1\},\ p_k \overset{\pi_k}{\rightsquigarrow} p'_k}{\downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \overset{\pi_k}{\rightsquigarrow} \downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(S2)}\ \frac{p_i \not\twoheadrightarrow \text{ for } i \in \{1,\dots k{-}1\},\ p_k \overset{\tau_k}{\twoheadrightarrow} p'_k}{\downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \overset{\tau_k}{\twoheadrightarrow} \downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(S3)}\ \frac{a \in A,\ p_i \overset{a}{\rightarrow} p_i^a \text{ for every } i \in \{1,\dots n\},\ B = \cap_{i=1}^n \mathcal{I}(p_i) \cap A}{\downarrow^{\omega,l}_{A,n} (p_1,\dots p_n) \overset{\tau^l_{(a,B)}}{\twoheadrightarrow} \|^{\omega,\tau^l_{(a,B)}}_{A,n} (p_1^a, p_2^a, \dots p_n^a)}$$

Figure 8.6: Operational semantics for parallel composition: rules for $\downarrow^{\omega,l}_{A,n}$

synchronization can happen, as the synchronization is n-ary.

The rules in Fig. 8.6, resp. Fig. 8.7 are adaptations of the rules in Fig. 8.1, resp. Fig. 8.2, such that the operands in $\downarrow^{\omega,l}_{A,n}$ synchronize on the first step and proceed in parallel ($\|^{\omega,l}_{A,n}$) afterwards, and the operands in $\Downarrow^{\omega,l}_{A,n}$ interleave on the first step and proceed in parallel ($\|^{\omega,l}_{A,n}$) afterwards. Note that, however, in rule AS4 (Fig. 8.7), that mimics rule I4 from Fig. 8.2, we have added the synchronization history as information to the labels of the resulting internal choice. That is, we have taken into account that which process wins the race may also depend on the synchronization history, which did not exist in the interleaving operator $|||^{\psi}$. The reason is that the processes may make decisions to let a certain process win based on their previous communication.

Remark Note that our definition of parallel composition with hiding, even when ignoring the labels on the internal transitions, is different than the generalized parallel composition of CSP, followed by hiding. For example, consider the processes $ac \,\Box\, b$ and a given in Fig. 8.8-a. Their CSP [96] parallel composition with synchronization on $\{a\}$ yields $ac \,\Box\, b$; hiding action a in the latter yields $c\sqcap(c \,\Box\, b)$. On the other hand, in our setting, the parallel composition $(ac \,\Box\, b) \,\|^{\widetilde{\omega},\varepsilon}_{\{a\},2}\, a$ yields an internal choice between actions c and b (see Fig. 8.8-b). This difference comes from an observation that parallel composition, followed by a hiding operator (Fig. 8.8-a), behaves differently than parallel composition with hiding after synchronization (Fig. 8.8-b). In

$$\text{(AS1)}\ \frac{p_i \not\twoheadrightarrow\not\rightarrow \text{ for } i \in \{1,\dots k{-}1\},\ p_k \overset{\pi_k}{\rightsquigarrow} p'_k}{\Downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \overset{\pi_k}{\rightsquigarrow} \Downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(AS2)}\ \frac{p_i \not\twoheadrightarrow \text{ for } i \in \{1,\dots k{-}1\},\ p_k \overset{\tau_k}{\twoheadrightarrow} p'_k}{\Downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p_k,p_{k+1}\dots p_n) \overset{\tau_k}{\twoheadrightarrow} \Downarrow^{\omega,l}_{A,n} (p_1,\dots p_{k-1},p'_k,p_{k+1}\dots p_n)}$$

$$\text{(AS3)}\ \frac{p_i \overset{a}{\not\twoheadrightarrow}\not\twoheadrightarrow\not\rightarrow \text{ for } i \neq k,\ p_k \overset{a}{\rightarrow} p^a_k, a \notin A}{\Downarrow^{\omega,l}_{A,n} (p_1,\dots,p_{k-1},p_k,p_{k+1},\dots p_j) \overset{a}{\rightarrow}\ \|^{\omega,l}_{A,n}(p_1,\dots,p_{k-1},p^a_k,p_{k+1},\dots p_j)}$$

$$\text{(AS4)}\ \frac{p_i \overset{a}{\rightarrow} p^a_i \text{ for all } i \in I \subseteq \{1,\dots j\},\ p_i \not\twoheadrightarrow\not\rightarrow\overset{a}{\not\twoheadrightarrow} \text{ for } i{\notin}I,\ a{\notin}A,\ \omega(a,I){=}m}{\begin{array}{c}\Downarrow^{\omega,l}_{A,n} (p_1,\dots p_j) \overset{a}{\rightarrow} \prod_{i\in I} \tau^{l,m+1}_{i,I,a} \left(\|^{\phi,l}_{A,n}(p_1,\dots,p_{i-1},p^a_i,p_{i+1},\dots p_j)\right), \\ \phi(a,I){=}m{+}1, \phi(b,J){=}\omega(b,J) \text{ for } b{\neq}a \text{ or } J{\neq}I\end{array}}$$

Figure 8.7: Operational semantics for parallel composition: rules for $\Downarrow^{\omega,l}_{A,n}$

the second case, if action b is in a menu of the composite, then it is not possible that action c is in the same menu. This is because the availability of action c requires that action a (out of a and b in the process $ac \,\Box\, b$) was performed. But the last means that b was excluded from further execution at the moment of choosing a between a and b.

8.3 Normal forms

In this section we generalize the model of process graphs introduced in the previous chapter, such that the labels on the internal transitions can be rational expressions over the label variables in the set $\mathcal{L}$. This generalization is needed in order to derive normal forms of the process graphs, that shall play an essential role in the proofs of the congruence theorem and the completeness of the axiomatization presented in the following sections. The need to generalize the model comes from an observation that processes such as $\tau_1 a \sqcap \tau_2 a \sqcap \tau_3 b$ and $\tau_4 a \sqcap \tau_5 b$ are ready-trace equivalent and should have com-

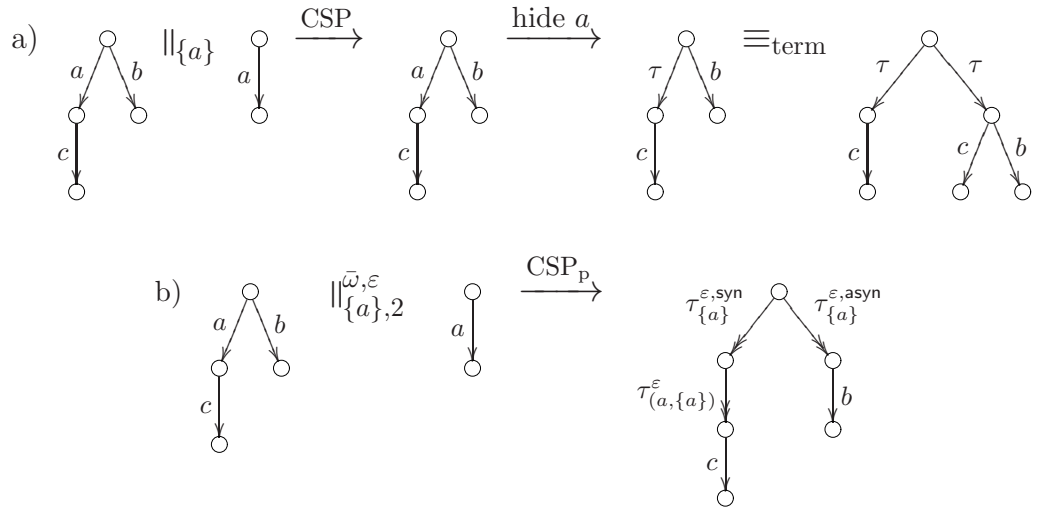

Figure 8.8: Differences between parallel composition followed by hiding (a) and parallel composition with hiding the synchronized actions (b)

parable normal forms. We obtain that the normal form of the former process is $(\tau_1+\tau_2)a \sqcap \tau_3 b$, while the normal form of the latter one is $\tau_4 a \sqcap \tau_5 b$. Then, we define two normal forms to be equivalent if they yield the same sets of deterministic processes when the internal nondeterminism is resolved.

The reduction to normal forms follows the lines of [13]; however, as pointed out above, the presence of probabilistic and labeled internal transitions in our setting introduces technical complications that do not exist in [13].

8.3.1 General process trees

In order to generalize the process graphs, we shall need a subset $\mathcal{Q}$ of the rational expressions over the elements of $\mathcal{L}$, generated by the following grammar:

$$\varphi ::= \alpha \mid l \mid \varphi_1 + \varphi_2 \mid \varphi_1 \cdot \varphi_2 \mid \frac{\varphi_1}{\varphi_2},$$

where $\alpha \in (0, 1]$, $l \in \mathcal{L}$, and "+", "$\cdot$", and "$\frac{\cdot}{\cdot}$" are ordinary algebraic addition, multiplication and fraction, respectively.

A *general-process tree* differs from a process tree in that the internal transitions are labeled with expressions in $\mathcal{Q}$, rather than only with labels in $\mathcal{L}$, and there are no restrictions on the expressions labeling the internal transitions.

Definition 8.3.1 (General-process tree)**.** A *general-process tree* r, or simply general process r, is a directed finite tree with root r, such that

- there exist three types of edges, or *transitions*: *action* ($\rightarrow$), *internal* ($\twoheadrightarrow$), and *probabilistic* ($\rightsquigarrow$);
- there exist three types of nodes, or *states*: *action*, *nondeterministic*, and *probabilistic*; from an action (resp. nondeterministic, probabilistic) state there can originate only action (resp. internal, probabilistic) transitions;
- the action transitions are labeled with actions from $\mathcal{A}$ such that no two action transitions with the same state of origin are labeled the same;
- the internal transitions are labeled with expressions from $\mathcal{Q}$;
- the probabilistic transitions are labeled with scalars from $(0, 1]$, such that (i) given two states, there is at most one probabilistic transition connecting them, and (ii) for each probabilistic state s, if $[\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}]$ then $\sum_{i\in I} \pi_i = 1$, i.e. the sum of the labels on the outgoing transitions equals one;
- all states are reachable from r.

Note that every process tree is also a general-process tree.

Next, we define transformations on general-process trees that shall lead to normal forms.

Definition 8.3.2 (General-process tree transformations)**.** Let p be a general-process tree. A transformation of p is called

(i) *substitution* if, for a state s in p, every transition $s \overset{\pi}{\rightsquigarrow} s'$ is replaced by $s \overset{\pi}{\twoheadrightarrow} s'$;

(ii) *erasing* if, given a transition $s \overset{\varphi}{\twoheadrightarrow} s'$ such that s has no other outgoing transitions, states s and s' are identified;

(iii) *compressing* if, for a state s such that $s \overset{\varphi}{\twoheadrightarrow} s'$ and $[\{s' \overset{\varphi_i}{\twoheadrightarrow} s'_i\}_{i\in I}]$, the transitions $\{s' \overset{\varphi_i}{\twoheadrightarrow} s'_i\}_{i\in I}$ are erased, and new transitions $\{s \overset{\varphi\varphi_i}{\twoheadrightarrow} s'_i\}_{i\in I}$ are created.

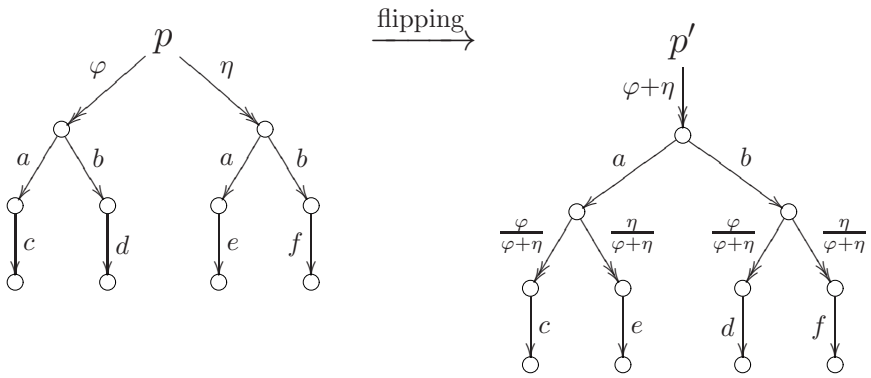

Figure 8.9: An example of "flipping"

(iv) *flipping* if, given a state s such that

$$s \overset{\varphi}{\twoheadrightarrow} s_1,\ s \overset{\eta}{\twoheadrightarrow} s_2,\ [\{s_1 \overset{a_i}{\longrightarrow} s_{1i}\}_{i\in I}],\ \text{and}\ [\{s_2 \overset{a_i}{\longrightarrow} s_{2i}\}_{i\in I}],$$

the transitions $s \overset{\varphi}{\twoheadrightarrow} s_1$ and $s \overset{\eta}{\twoheadrightarrow} s_2$ are erased, and new states s' and $\{s'_i\}_{i\in I}$ are created, together with the transitions

$$s \overset{\varphi+\eta}{\twoheadrightarrow} s',\ \{s' \overset{a_i}{\longrightarrow} s'_i\}_{i\in I},\ \{s'_i \overset{\frac{\varphi}{\varphi+\eta}}{\twoheadrightarrow} s_{1i}\}_{i\in I},\ \text{and}\ \{s'_i \overset{\frac{\eta}{\varphi+\eta}}{\twoheadrightarrow} s_{2i}\}_{i\in I};$$

(v) *deadlocks-joining* if, given a state s such that $s \overset{\varphi}{\twoheadrightarrow} s_1$, $s \overset{\eta}{\twoheadrightarrow} s_2$, and s_1, s_2 are both deadlock states, the transitions $s \overset{\varphi}{\twoheadrightarrow} s_1$ and $s \overset{\eta}{\twoheadrightarrow} s_2$ are erased, and a new transition $s \overset{\varphi+\eta}{\twoheadrightarrow} s_1$ is created.

Example 8.3.3. Fig. 8.9 gives an example of the transformation "flipping".

Next, we narrow down the set of general trees of interest to the set of those that are obtained from unfolded process trees by transformations.

Definition 8.3.4 (CG process tree)**.** A *coherent general-process tree* (CG process tree) is a general-process tree that can be obtained from a process tree, unfolded by Def. 7.2.1, by transforming it zero or more times by Def. 8.3.2.

Note that a transformation from Def. 8.3.2 does not essentially increase the set of constraints for a general-process tree, i.e. no new labels are added by a transformation step and no new restrictions to the old labels are imposed. We formalize this statement in the following proposition. For our convenience, we assume that a rational expression φ/φ in $\mathcal{Q}$ that is used for labeling a transition in a CG process tree always has value 1, even when φ evaluates to 0. Later, we will justify this assumption.

Proposition 8.3.5. *Let p be a* CG *process tree and p' be a* CG *process tree obtained from s via a transformation step. Then, the systems of equations $\mathcal{C}(p)$ and $\mathcal{C}(p) \cup \mathcal{C}(p')$ are equivalent.*

Proof. "Substitution" adds a constraint to $\mathcal{C}(p) \cup \mathcal{C}(p')$ that is trivially satisfied, while "erasing" adds no new constraints. "Compressing" adds a constraint

$$\psi_1 \sum_{i \in I} \varphi_i + \sum_{j \in J, j \neq 1} \psi_j = 1,$$

where $\psi_1 = \varphi$, $[\{s \overset{\psi_j}{\twoheadrightarrow} s_j\}_{j \in J}]$, and $s_1 = s'$. This constraint follows from the constraints

$$\sum_{i \in I} \varphi_i = 1 \text{ and } \sum_{j \in J} \psi_j = 1,$$

which are already in $\mathcal{C}(p)$. "Flipping" adds a constraint $\frac{\varphi}{\varphi+\eta} + \frac{\eta}{\varphi+\eta} = 1$, which is trivially satisfied. "Deadlock-joining" adds no constraints that are not already in $\mathcal{C}(p)$. □

Since a transformation does not increase the set of constraints defined by the original process tree, the CG process tree inherits the (linear) constraints from its original (unfolded) process tree. We introduce some notation to formalize this. Let p be a process tree, unfolded by Def. 7.2.1, and p' be a CG process tree obtained from p via zero or more transformation steps. By $\tilde{\mathcal{C}}(p')$ we denote $\mathcal{C}(p)$. Recall that a resolution of a set of constraints is a function that assigns values in $[0, 1]$ to the variables in the constraints, respecting the constraints. A *resolution* of p' is the process tree $\bar{p}'$ obtained when, for an arbitrary resolution λ of $\tilde{\mathcal{C}}(p')$, every transition

$$t \overset{\varphi}{\twoheadrightarrow} t'$$

in graph p' is replaced by

$$t \overset{\lambda(\varphi)}{\rightsquigarrow} t',$$

if $\lambda(\varphi) \neq 0$, or erased otherwise, where $\lambda(\varphi)$ for $\varphi \in \mathcal{Q} \setminus \mathcal{L}$ is defined inductively in the usual way, i.e.

$$\begin{aligned} \lambda(\alpha) &= \alpha \quad \text{for } \alpha \in (0, 1], \\ \lambda(\varphi_1 + \varphi_2) &= \lambda(\varphi_1) + \lambda(\varphi_2), \\ \lambda(\varphi_1 \cdot \varphi_2) &= \lambda(\varphi_1) \cdot \lambda(\varphi_2), \\ \lambda(\frac{\varphi_1}{\varphi_2}) &= \frac{\lambda(\varphi_1)}{\lambda(\varphi_2)}. \end{aligned}$$

From now on, unless stated otherwise, we assume that the (unfolded) process tree from which a CG process tree originates is implicitly given and we shall omit it.

Note that the assumption that φ/φ always evaluates to 1 can be now justified by the fact that a label φ/η on an internal transition in a CG process tree originates from "flipping". Namely, in this case there is a transition labeled with η, preceding the transition labeled with φ/η (see Fig. 8.9 and Def. 8.3.2). When $\lambda(\eta) = 0$, in the resolution of the CG process tree the transition labeled with η does not appear, and thus the transition labeled with φ/η does not appear, too. Because of this, our assumption that φ/φ always evaluates to 1 in Prop. 8.3.5 has no influence on the resolution of a CG process tree p, which is the only context in which the constraints $\mathcal{C}(p)$ are used.

Having defined resolutions of CG process trees, the definition of ready-trace equivalence for process trees (Def. 7.4.6) easily extends to CG process trees.

Definition 8.3.6 (Ready-trace equivalence)**.** Let s' and t' be CG process trees. We say s' *implements* t' *w.r.t. ready-traces*, denoted by $s' \preceq_{\mathsf{RT}} t'$ iff, for every resolution $\bar{s}'$ of s', there exists a resolution $\bar{t}'$ of t' such that for all ready-traces $(M_1, a_1, \dots M_k)$,

- $P^1_{\bar{s}'}(M_1) = P^1_{\bar{t}'}(M_1)$ and
- if $k > 1$, then $P^k_{\bar{s}'}(M_k | M_1, a_1, \dots M_{k-1}, a_{k-1})$ is defined if and only if $P^k_{\bar{t}'}(M_k | M_1, a_1, \dots M_{k-1}, a_{k-1})$ is defined, and, in case they are both defined, they are equal.

s' and t' are *ready-trace equivalent*, denoted by $s' \approx_{\mathsf{RT}} t'$, iff $s' \preceq_{\mathsf{RT}} t'$ and $t' \preceq_{\mathsf{RT}} s'$.

Proposition 8.3.7 (Soundness of the transformations)**.** *The transformations in Definition 8.3.2 are sound w.r.t.* $\approx_{\mathsf{RT}}$ *when applied to* CG *process trees, i.e. if* p' *is obtained from* p *via a transformation, then* $p' \approx_{\mathsf{RT}} p$.

Proof. We give a proof sketch for "flipping", while for the rest of the transformations the proof is straightforward. Suppose p' is obtained from p by flipping, as in Fig. 8.9 (note that p may have other outgoing internal transitions, omitted in the figure). First, note that the set of ready-traces remains unchanged. Let λ be resolution of $\mathcal{C}(p)$. Then, if $\lambda(\varphi + \eta) \neq 0$, we obtain that, for the resolutions $\bar{p}$ and $\bar{p}'$ of p, resp. p' with respect to λ, the conditional probabilities of any ready-trace of $\bar{p}$ and $\bar{p}'$ coincide. If $\lambda(\varphi + \eta) = 0$,

then $\bar{p}$ and $\bar{p}'$ are isomorphic, as the branches of p and p' given in Fig. 8.9 do not appear in $\bar{p}$, resp. $\bar{p}'$. □

8.3.2 Normal forms

We define normal forms of CG process trees and show that ready-trace equivalent CG process trees reduce to equivalent normal forms, i.e. to normal forms that yield the same sets of deterministic processes when the internal nondeterminism is resolved.

Definition 8.3.8 (Normal form)**.** A CG process tree is in *ready-trace normal form* (RT-normal form) if none of the transformations of Def. 8.3.2 can be applied.

Proposition 8.3.9. *Every general-process tree transformation sequence eventually ends in* RT*-normal form.*

Proof. A transformation of type "substitution" replaces the probabilistic transitions by internal transitions, which is clearly terminating. A transformation of type "flipping" reduces the number of action transitions. Therefore, there will be a moment when "flipping" will never again be applied, because the minimal possible number of action transitions is zero, and when "substitution" will never again be applied, because all the probabilistic transitions have been replaced by internal transitions. After that moment, only transformations of type "erasing", "compressing ", or "deadlock-joining" can be applied, each of which reduces the number of internal transitions by one and does not increase the number of the other types of transitions. Thus, the transformations terminate and the proof is complete. □

We call an internal transition $s \overset{\varphi}{\twoheadrightarrow} t$ in a CG process tree *trivial* if s has no other outgoing transitions.

Lemma 8.3.10. *A* CG *process tree p is in normal form iff it contains no probabilistic transitions, no consecutive internal transitions, no trivial internal transitions and, moreover, given an arbitrary nondeterministic state s of p, if $s \twoheadrightarrow s_1$, $s \twoheadrightarrow s_2$, and $\mathcal{I}(s_1) = \mathcal{I}(s_2)$ for some states s_1 and s_2, then $s_1 = s_2$.*

Proof. Directly from Definition 8.3.2 and Definition 8.3.8. □

Recall that a process tree is deterministic if it does not contain internal transitions.

Definition 8.3.11 (Isomorphism)**.** Two finite deterministic process trees p and q are *isomorphic* if there exists a bijection f, mapping the state set of p to the one of q, such that

- $f(p) = q$,
- if $s \xrightarrow{a} s'$ for s, s' in p, then $f(s) \xrightarrow{a} f(s')$, and
- if $s \overset{\pi}{\rightsquigarrow} s'$ for s, s' in p, then $f(s) \overset{\pi}{\rightsquigarrow} f(s')$.

We now define the relation *almost-equal* on CG process trees. Two CG process trees are almost-equal if they yield the same sets of deterministic processes when the internal nondeterminism is resolved.

Definition 8.3.12. Two CG process trees p and q are *almost-equal*, denoted $p \simeq q$, iff

- for every resolution $\bar{p}$ of p there exists a resolution $\bar{q}$ of q such that $\bar{p}$ and $\bar{q}$ are isomorphic, and viceversa,
- for every resolution $\bar{q}$ of q there exists a resolution $\bar{p}$ of p such that $\bar{p}$ and $\bar{q}$ are isomorphic.

The following lemma shall be needed in the proof of Prop. 8.3.14.

Lemma 8.3.13. *Let p be a finite process tree, unfolded by Def. 7.2.1. Every variable in $\mathcal{C}(p)$ appears in only one equation in $\mathcal{C}(p)$.*

Proof. From Def. 8.3.1, Def. 7.2.1, and Def. 7.1.3. □

Proposition 8.3.14. *Let p and q be* CG *process trees in* RT*-normal form. $p \approx_{\mathsf{RT}} q$ iff p and q are almost-equal.*

Proof. ($\Rightarrow$) The proof is by structural induction on p and using Lemma 8.3.10.

Assume first that $[\{p \xrightarrow{a_i} p_i\}_{i \in I}]$ for $\{p_i\}_{i \in I} = \{0\}$. Then, from the facts that $p \approx_{\mathsf{RT}} q$ and q is in normal form, and from Lemma 8.3.10, it must hold that $[\{q \xrightarrow{a_i} q_i\}_{i \in I}]$ and $\{q_i\}_{i \in I} = \{0\}$. Therefore, p and q are isomorphic and thus almost-equal. Note that the case $[\{p \overset{\varphi_i}{\twoheadrightarrow} p_i\}_{i \in I}]$ for $\{p_i\}_{i \in I} = \{0\}$ is not possible, since p is in normal form.

Assume that $[\{p \xrightarrow{a_i} p_i\}_{i \in I}]$, where $\{p_i\}_{i \in I}$ are arbitrary CG process trees. Then, because p is in normal form and because of Lemma 8.3.10, all CG process trees in $\{p_i\}_{i \in I}$ are in normal form. On the other hand, because

$p \approx_{\mathsf{RT}} q$ and q is in normal form, from Def. 8.3.6 and from Lemma 8.3.10 we have that p and q have the same initial menus, i.e. $[\{q \xrightarrow{a_i} q_i\}_{i\in I}]$ for some CG process trees $\{q_i\}_{i\in I}$, all of which are in normal form. From $p \approx_{\mathsf{RT}} q$ and from Def. 8.3.6 it easily follows that $p_i \approx_{\mathsf{RT}} q_i$ for all $i \in I$. From the inductive assumption, it follows that $p_i \simeq q_i$ for all $i \in I$. Therefore, $p \simeq q$.

Assume now that $[\{p \overset{\varphi_i}{\twoheadrightarrow} p_i\}_{i\in I}]$, where $\{p_i\}_{i\in I}$ are arbitrary CG process trees. Then, every CG process tree in $\{p_i\}_{i\in I}$ is in normal form. Because q is in normal form and $p \approx_{\mathsf{RT}} q$, from Def. 8.3.6 and from Lemma 8.3.10 it must be that $[\{q \overset{\rho_i}{\twoheadrightarrow} q_i\}_{i\in I}]$ for some $\{q_i\}_{i\in I}$ such that $\mathcal{I}(p_i) = \mathcal{I}(q_i)$ for every $i \in I$. Assume that $p_i \not\approx_{\mathsf{RT}} q_i$ for some $i \in I$. Without loss of generality, we have that there exists a resolution $\bar{p}_i$ of p_i such that for every resolution $\bar{q}_i$ of q_i there exists a ready-trace whose conditional probabilities for $\bar{p}_i$ and $\bar{q}_i$ do not match. From Def. 8.3.4, Prop. 8.3.5, and from Lemma 8.3.13, it is not hard to show that we can extend the resolution $\bar{p}_i$ of p_i to a resolution $\bar{p}$ of p (the equations in $\tilde{\mathcal{C}}(p)$ do not add restrictions to the variables in $\tilde{\mathcal{C}}(p_i)$ that are not already in $\tilde{\mathcal{C}}(p_i)$), such that for every resolution $\bar{q}$ of q there exists a ready-trace whose conditional probabilities for $\bar{p}$ and $\bar{q}$ do not match. Thus, we obtain that $p \not\approx_{\mathsf{RT}} q$, which contradicts our assumption. Therefore, $p_i \approx_{\mathsf{RT}} q_i$ for every $i \in I$. Then, by the inductive assumption, it follows that $p_i \simeq q_i$ for all $i \in I$. From the fact that $p \approx_{\mathsf{RT}} q$ and using the same argument as above, it can be concluded that $p \simeq q$. □

Proposition 8.3.15. *Two* CG *process trees p and q reduce to almost-equal* RT*-normal forms iff p and q are ready-trace equivalent.*

Proof. Follows from Propositions 8.3.7, 8.3.9, and 8.3.14. □

8.4 Congruence property for $\approx_{\mathsf{RT}}$

In this section we state one of our main results, namely that the probabilistic ready-trace equivalence on CG process trees is a congruence for the operators defined so far. We extend our grammar with the *general internal choice* operator, i.e. the language of $\mathrm{CSP}_{\mathrm{p}}^{\mathrm{g}}$ is generated by the grammar

$$x ::= y \;\mid\; \sqcap_{i\in I}\varphi_i x_i,$$

where y is a $\mathrm{CSP}_{\mathrm{p}}$ process term and φ_i are expressions in $\mathcal{Q}$. The semantic rules for the operators in $\mathrm{CSP}_{\mathrm{p}}^{\mathrm{g}}$ are inherited from the rules for $\mathrm{CSP}_{\mathrm{p}}$ given in Sections 8.1 and 8.2, by treating the internal transitions as *general* internal transitions, labeled with $\mathcal{Q}$-expressions, rather than only with labels in $\mathcal{L}$.

For the congruence theorem we need *compatibility* of the components. We elaborate more on compatibility afterwards.

Definition 8.4.1. Two CG process trees are *compatible* if they have disjoint sets of labels in $\mathcal{L}$.

Theorem 8.4.2 (Congruence)**.** *Let $\{p_i\}_{i\in I}$ and $\{q_i\}_{i\in I}$ be two sets of* CG *process trees, such that $\{p_i\}_{i\in I}$ are pairwise compatible, and the same holds for $\{q_i\}_{i\in I}$. Let $p_i \approx_{\mathsf{RT}} q_i$ for every $i \in I$. Then,*

$$\begin{aligned}
\Box_{i\in I}\, a_i p_i &\approx_{\mathsf{RT}} \Box_{i\in I}\, a_i q_i,\\
\sqcap_{i\in I}\tau_i p_i &\approx_{\mathsf{RT}} \sqcap_{i\in I}\tau_i q_i,\\
\oplus_{i\in I}\pi_i p_i &\approx_{\mathsf{RT}} \oplus_{i\in I}\pi_i q_i,\\
p_k \Box\, p_m &\approx_{\mathsf{RT}} q_k \Box\, q_m,\\
\Theta p_m &\approx_{\mathsf{RT}} \Theta q_m,\\
\|^l_n\ (p_1 \ldots p_n) &\approx_{\mathsf{RT}} \|^l_n\ (q_1 \ldots q_n),\\
|||^{\psi}\ (p_1 \ldots p_n) &\approx_{\mathsf{RT}} |||^{\psi}(q_1 \ldots q_n),\\
\|^{\omega,l}_{A,n}\ (p_1 \ldots p_k) &\approx_{\mathsf{RT}} \|^{\omega,l}_{A,n}(q_1 \ldots q_k),\\
\downarrow^{\omega,l}_{A,n}(p_1 \ldots p_n) &\approx_{\mathsf{RT}} \downarrow^{\omega,l}_{A,n}(q_1 \ldots q_n),\ \textit{and}\\
\Downarrow^{\omega,l}_{A,n}(p_1 \ldots p_k) &\approx_{\mathsf{RT}} \Downarrow^{\omega,l}_{A,n}(q_1 \ldots q_k)\ \textit{for } k \le n.
\end{aligned}$$

Proof. We prove that $\|^{\omega,l}_{A,n}\ (p_1 \ldots p_k) \approx_{\mathsf{RT}} \|^{\omega,l}_{A,n}(q_1 \ldots q_k)$, since this is the most involved case, and the proof for the rest of the parallel operators follows immediately. We present the proof for the case $n = 2$; the proof for an arbitrary n is based on the same ideas. Thus, we show that

$$\text{if } p \approx_{\mathsf{RT}} q, \text{ then } \|^{\omega,l}_{A,2}\ (p, r) \approx_{\mathsf{RT}} \|^{\omega,l}_{A,2}(q, r),$$

where p, q, and r are CG process trees.
Because of Proposition 8.3.15, it suffices to show that

(i) if q is obtained from p by a transformation step, then $\|^{\omega,l}_{A,2}(q, r)$ and $\|^{\omega,l}_{A,2}\ (p, r)$ are ready-trace equivalent, and

(ii) if p and q are almost-equal, then $\|^{\omega,l}_{A,2}(p, r)$ and $\|^{\omega,l}_{A,2}(q, r)$ are almost-equal, too (straightforward).

(i) The most involved case is when q is obtained from p by flipping. Without loss of generality, we can assume that state q is obtained from state p by flipping. The proof is by induction on the maximal length, length(r), of a

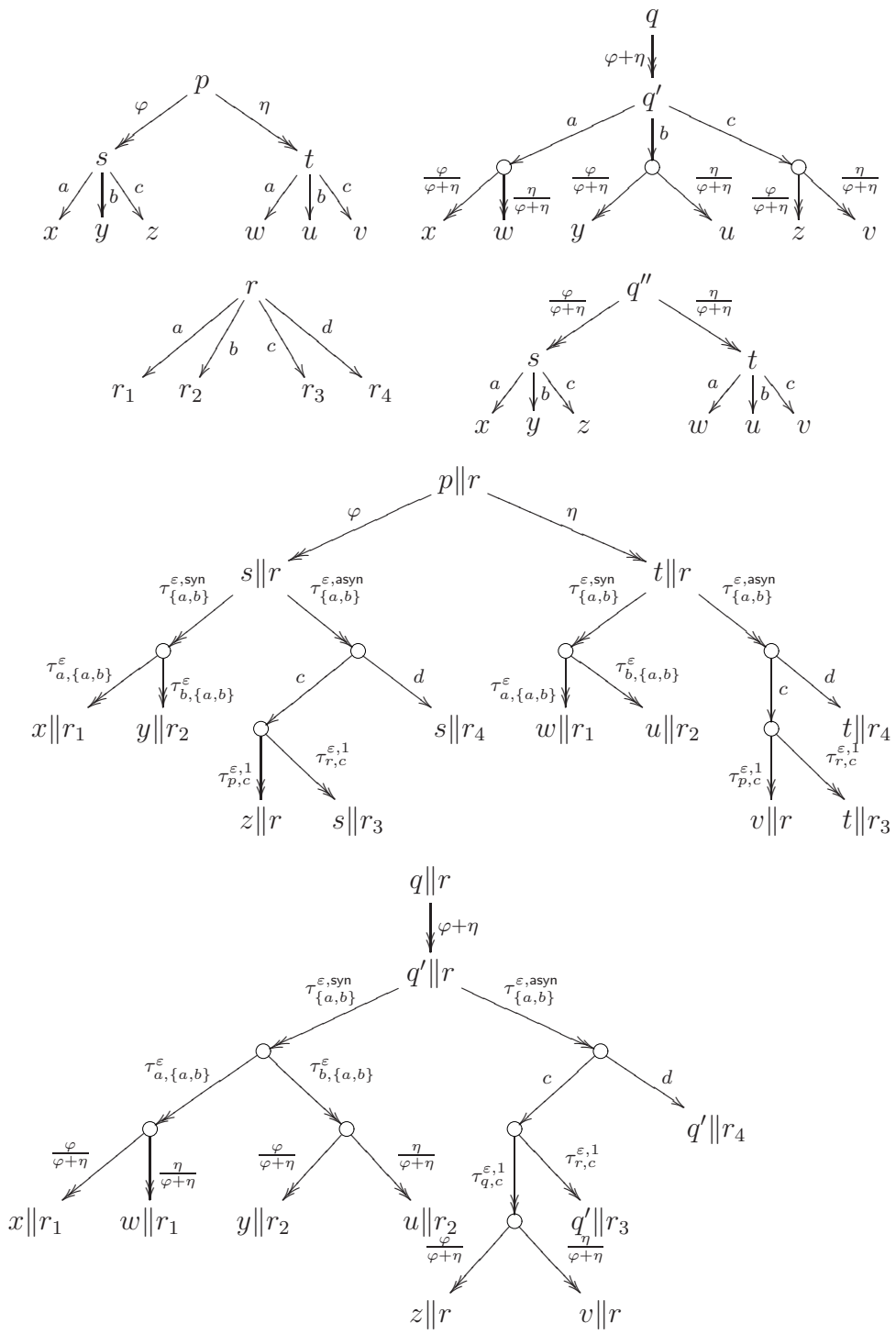

Figure 8.10: "Flipping" and composing in parallel. $A = \{a, b\}$.

sequence of observable actions that process r can perform. Since the rigorous proof uses heavy notation, but is actually not very difficult, it is best presented via an example, given in Fig. 8.10. In this example, the set A on which processes synchronize is equal to $\{a, b\}$. (For clarity, in the figure we have omitted the parameters of the operator $\|$ and we have shortened the notation for the labels on the internal transitions determining the winner of the race on action c.) Now, note that the "symbolic" probabilities to observe menu $\{c, d\}$ are the same for both $p\|r$ and $q\|r$, i.e. they are equal to $(\varphi + \eta)\tau^{\varepsilon,\mathsf{asyn}}_{\{a,b\}}$. Note also that the "symbolic" probabilities to reach any of the processes $x\|r_1$, $y\|r_2$, $w\|r_1$, and $u\|r_2$ are equal for both $p\|r$ and $q\|r$. For processes $z\|r$ and $v\|r$, the symbolic probabilities in $p\|r$ and $q\|r$, conditioned that the sequence $(\{c, d\}, c)$ was observed, are not exactly the same, however, the probabilities can be matched by considering all possible resolutions of the internal nondeterminism. Thus, the non-trivial case is to show that the (symbolic) conditional probabilities of any ready-trace following action d in $p\|r$ and $q\|r$ are the same (and similarly for the ready-traces following action c).

Assume first that length$(r) = 1$. Then, processes $s\|r_3$ and $s\|r_4$ are equivalent to s, processes $t\|r_3$ and $t\|r_4$ are equivalent to t, and processes $q'\|r_3$ and $q'\|r_4$ are equivalent to q'. Then, it is easy to check that the symbolic probabilities to observe menu $\{a, b, c\}$ after observing menu $\{c, d\}$ and action d are the same for both $p\|r$ and $q\|r$ (and equal to 1), and that the probabilities to observe menu $\{a, b, c\}$ after observing menu $\{c, d\}$ and action c can be made equal by resolutions for both $p\|r$ and $q\|r$. The symbolic probability to observe process x, given that the sequence $(\{c, d\}, d, \{a, b, c\}, a)$ was observed is equal to φ for both $p\|r$ and $q\|r$. Similarly for the rest of the symbolic probabilities. Note that if r starts with a probabilistic or internal transition, this transition takes precedence over the action transitions in p or q in parallel, and the proof reduces to the one for the case when r is an action state.

Assume now that length$(r) > 1$. From the inductive assumption it follows that process $q'\|r_3$ is ready-trace equivalent to process $q''\|r_3$ (process q'' is also given in Fig. 8.10), and similarly for processes $q'\|r_4$ and $q''\|r_4$. Thus, we can replace processes $q'\|r_3$ and $q'\|r_4$ in $q\|r$ with processes $q''\|r_3$, resp. $q''\|r_4$, and obtain a process q''' which is ready-trace equivalent to $q\|r$. Now, it is easy to check that the probabilities for processes $s\|r_3$, $s\|r_4$, $t\|r_3$, and $t\|r_4$ can be matched by resolutions (if they do not coincide symbolically) for both $p\|r$ and q'''. When r starts with a probabilistic or internal transition, the proof again reduces to one for the case when r is an action state. □

Remark Note that in Theorem 8.4.2 we have implicitly assumed that the

set of CG process trees is closed under the operators considered in the theorem. This property, however, can be easily concluded from the proof of the theorem; namely, it can be concluded that if q is obtained from p via transformation, then $q\|^{\omega,l}_{A,2}r$ can be obtained from $p\|^{\omega,l}_{A,2}r$ via transformations. Clearly, the general internal choice operator does not always yield CG process trees; however, as discussed in Sec. 8.3, this operator is introduced only for technical purposes and is not interesting on its own.

Remark The compatibility requirement in Theorem 8.4.2, that the operands must not have common labels for the internal transitions when they are composed, is essential. For example, take processes $p_1 \equiv \tau_1 c \sqcap \tau_2 d$, $p_2 \equiv \tau_3 c \sqcap \tau_4 d$, and $q \equiv \tau_1 a \sqcap \tau_2 b$. Then, although $p_1 \approx_{\mathsf{RT}} p_2$, we have that $ap_1 \,\Box\, bq \not\approx_{\mathsf{RT}} ap_2 \,\Box\, bq$. Namely, note that there is a resolution, say $\bar{u}$, for process $ap_2 \,\Box\, bq$, such that $P^2_{\bar{u}}(\{c\}|\{a,b\},a) = 0.5$ and $P^2_{\bar{u}}(\{a\}|\{a,b\},b) = 0.3$; on the other hand, for every resolution of process $ap_1 \,\Box\, bq$, the values of these two conditional probabilities are the same. However, in practice, this compatibility requirement does not decrease the expressivity of the language, since one can always rename the labels used in one process before composing. The labels in a process can be seen as process local variables, serving to identify the internal transitions, which are local for the process, when the process is put in a context. Thus, their exact names are irrelevant[4]. Furthermore, note that process $ap_1 \,\Box\, bq$ given above is an unnatural construction. Namely, since the internal choices in p_1 and q are independent from each other, there is no reasonable justification for the requirement that they should be resolved in the same manner in the external choice between p_1 and q.

Remark In this chapter the interest is on the equivalence relation $\approx_{\mathsf{RT}}$, since in what follows we are going to axiomatize it. However, note that from the proof of Theorem 8.4.2, it follows that the preorder relation $\preceq_{\mathsf{RT}}$ is also a precongruence for the operators mentioned in the theorem.

8.5 Axiomatic characterization of $\approx_{\mathsf{RT}}$

In this section we give an axiomatic characterization of probabilistic ready-trace equivalence $\approx_{\mathsf{RT}}$ defined on CG process trees for the operators in $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$.

The set of axioms of theory $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$ is given in Fig. 8.11, Fig. 8.12, and Fig. 8.13, where $x, y, ...$ are arbitrary $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$ terms, and we assume that $\mathcal{A} = \{a_i\}_{i\in N}$ for some index set N. For clarity, we have present the axioms for the binary parallel composition operators, and the axioms for the n-ary operators

[4]See [31] for a similar congruence condition.

(A1) $x \,\Box\, 0 = x$
(A2) $0 \,\Box\, x = x$
(A3) $(\Box_{i\in I}\ a_i x_i) \,\Box\, (\Box_{j\in J}\ a_j y_j) =$
$\quad (\Box_{k\in I\cap J}\ a_k(\tau_{k1}x_k \sqcap \tau_{k2}y_k)) \,\Box\, \big((\Box_{i\in I\setminus J}\ a_i x_i) \,\Box\, (\Box_{j\in J\setminus I}\ a_j y_j)\big)$, if $I\cap J\neq\emptyset$
(A4) $(\Box_{i\in I}\ a_i x_i) \,\Box\, (\Box_{i\in J}\ a_i x_i) = \Box_{i\in I\cup J}\ a_i x_i$, if $I \cap J = \emptyset$
(A5) $\oplus_{i\in I}\ \pi_i x_i = \sqcap_{i\in I}\pi_i x_i$
(A6) $\sqcap_{i\in I}\ \varphi_i x = x$
(A7) $\Box_{i\in I}\ a_i \sqcap_{j\in J}\ \varphi_j x_{ij} = \sqcap_{j\in J}\varphi_j\ \Box_{i\in I}\ a_i x_{ij}$
(A8) $(\sqcap_{i\in I}\varphi_i x_i) \,\Box\, y = \sqcap_{i\in I}\varphi_i\,(x_i \,\Box\, y)$
(A9) $y \,\Box\, (\sqcap_{i\in I}\varphi_i x_i) = \sqcap_{i\in I}\varphi_i\,(y \,\Box\, x_i)$
(A10) $\sqcap_{i\in J}\ \varphi_i(\sqcap_{k\in K}\rho_{ik}x_{ik}) = \sqcap_{i\in J,k\in K}(\varphi_i\rho_{ik})x_{ik}$
(A11) $\Theta 0 = 0$
(A12) $\Theta(\sqcap_{i\in I}\tau_i x_i) = \sqcap_{i\in I}\tau_i\Theta x_i$
(A13) $\Theta(\oplus_{i\in I}\pi_i x_i) = \oplus_{i\in I}\pi_i\Theta x_i$
(A14) $\Theta(ax) = a\Theta x$
(A15) if $a > b$ then $\Theta((ax \,\Box\, by) \,\Box\, z) = \Theta(ax \,\Box\, z)$
(A16) if $a > b$ then $\Theta(z \,\Box\, (ax \,\Box\, by)) = \Theta(z \,\Box\, ax)$
(A17) if $a > b$ then $\Theta((by \,\Box\, ax) \,\Box\, z) = \Theta(ax \,\Box\, z)$
(A18) if $a > b$ then $\Theta(z \,\Box\, (by \,\Box\, ax)) = \Theta(z \,\Box\, ax)$
(A19) if $\{a_i\}_{i\in I}$ are incomparable by $>$, then $\Theta(\Box_{i\in I}\ a_i x_i) = \Box_{i\in I}\ a_i\Theta x_i$

Figure 8.11: Axioms for choice and priority operators

are similar. For axiom A3 it is assumed that the labels of the internal choices, τ_{k1} and τ_{k2}, for every k, are new with respect to the labels that appear in the rest of the terms in the axiom. Note how axiom A5 reflects the view that probabilistic choice is a special type of internal choice. Note also the existence of axiom A7, which is typical for the non-probabilistic CSP.

We refer to a $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$ term as *basic term* if only external action choice, internal choice and probabilistic choice appear in the term. The next proposition says that every term can be rewritten to a basic term by using the axioms of $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$.

$$
\begin{array}{lll}
\text{(P1)} & 0\|^{\omega,l}_{A,2}0 & = 0 \\
\text{(P2)} & 0 \downarrow^{\omega,l}_{A,2} 0 = 0 & \\
\text{(P3)} & 0 \Downarrow^{\omega,l}_{A,2} 0 = 0 & \\
\text{(P4)} & (\Box_{i\in I}\ a_i x_i)\,\|^{\omega,l}_{A,2}0 & = \Box_{j\in J}\ a_j \left(x_j \|^{\omega,l}_{A,2} 0\right), \quad \text{for } \{a_j\}_{j\in J} = \{a_i\}_{i\in I} \setminus A \\
\text{(P5)} & 0\|^{\omega,l}_{A,2}(\Box_{i\in I}\ a_i x_i) & = \Box_{j\in J}\ a_j \left(0 \|^{\omega,l}_{A,2} x_j\right), \quad \text{for } \{a_j\}_{j\in J} = \{a_i\}_{i\in I} \setminus A \\
\text{(P6)} & (\Box_{i\in I}\ a_i x_i) \Downarrow^{\omega,l}_{A,2} 0 & = \Box_{j\in J}\ a_j \left(x_j \|^{\omega,l}_{A,2} 0\right), \quad \text{for } \{a_j\}_{j\in J} = \{a_i\}_{i\in I} \setminus A \\
\text{(P7)} & 0 \Downarrow^{\omega,l}_{A,2} (\Box_{i\in I}\ a_i x_i) & = \Box_{j\in J}\ a_j \left(0 \|^{\omega,l}_{A,2} x_j\right), \quad \text{for } \{a_j\}_{j\in J} = \{a_i\}_{i\in I} \setminus A \\
\text{(P8)} & (\Box_{i\in I}\ a_i x_i) \downarrow^{\omega,l}_{A,2} 0 & = 0 \\
\text{(P9)} & 0\|^{\omega,l}_{A,2}(\Box_{i\in I}\ a_i x_i) & = 0 \\
\text{(P10)} & (\sqcap_{i\in I}\varphi_i x_i)\,\|^{\omega,l}_{A,2} y & = \sqcap_{i\in I}\ \varphi_i \left(x_i \|^{\omega,l}_{A,2} y\right) \\
\text{(P11)} & y\|^{\omega,l}_{A,2}(\sqcap_{i\in I}\varphi_i x_i) & = \sqcap_{i\in I}\ \varphi_i \left(y \|^{\omega,l}_{A,2} x_i\right) \\
\text{(P12)} & (\sqcap_{i\in I}\varphi_i x_i) \Downarrow^{\omega,l}_{A,2} y & = \sqcap_{i\in I}\ \varphi_i \left(x_i \Downarrow^{\omega,l}_{A,2} y\right) \\
\text{(P13)} & y \Downarrow^{\omega,l}_{A,2} (\sqcap_{i\in I}\varphi_i x_i) & = \sqcap_{i\in I}\ \varphi_i \left(y \Downarrow^{\omega,l}_{A,2} x_i\right) \\
\text{(P14)} & (\sqcap_{i\in I}\varphi_i x_i) \downarrow^{\omega,l}_{A,2} y & = \sqcap_{i\in I}\ \varphi_i \left(x_i \downarrow^{\omega,l}_{A,2} y\right) \\
\text{(P15)} & y \downarrow^{\omega,l}_{A,2} (\sqcap_{i\in I}\varphi_i x_i) & = \sqcap_{i\in I}\ \varphi_i \left(y \downarrow^{\omega,l}_{A,2} x_i\right)
\end{array}
$$

Figure 8.12: Axioms for parallel composition, part I

Proposition 8.5.1 (Elimination)**.** *For a* $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$ *process term* x *there exists a basic term* y*, such that* $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}} \vdash x = y$*.*

Proof. The proof is straightforward, by structural induction. □

Observe that each CG process tree can be mapped to a basic term. We overload the notation and use the name p for a process term that represents the CG process tree p.

For $x \equiv \Box_{i\in I}\ a_i x_i,\ y \equiv \Box_{j\in J}\ a_j y_j,\ A = \{a_i\}_{i\in M}$,
$B = \{a_k\}_{k\in K} = \{a_i\}_{i\in I\cap J\cap M},\ C = \{a_i\}_{i\in I\cup J},\ \omega = \{(a_i, n_i)\}_{a_i\in\mathcal{A}\setminus A}$:

(P16) $x\|^{\omega,l}_{A,2} y = \tau^{l,\mathsf{syn}}_B\left(x \downarrow^{\omega,l}_{A,2} y\right) \sqcap \tau^{l,\mathsf{asyn}}_B\left(x \Downarrow^{\omega,l}_{A,2} y\right),$ if $B \neq \emptyset,\ C \not\subseteq A$

(P17) $x\|^{\omega,l}_{A,2} y = x \Downarrow^{\omega,l}_{A,2} y,$ if $B = \emptyset$

(P18) $x\|^{\omega,l}_{A,2} y = x \downarrow^{\omega,l}_{A,2} y,$ if $C \subseteq A$

(P19) $x \downarrow^{\omega,l}_{A,2} y = \sqcap_{k\in K}\tau^l_{(a_k,B)}\left(x_k\|^{\omega,\tau^l_{(a_k,B)}}_{A,2} y_k\right),$ if $B \neq \emptyset$

(P20) $x \downarrow^{\omega,l}_{A,2} y = 0,$ if $B = \emptyset$

(P21) $x \Downarrow^{\omega,l}_{A,2} y = \left(\Box_{i\in I\setminus(J\cup M)}\ a_i\left(x_i\|^{\omega,l}_{A,2} y\right)\right) \Box \left(\Box_{i\in J\setminus(I\cup M)}\ a_i\left(x\|^{\omega,l}_{A,2} y_i\right)\right) \Box$
$\left(\Box_{m\in(I\cap J)\setminus M}\ a_m\left(\tau^{n_m+1}_{1,a_m}\left(x_m\|^{\phi,l}_{A,2} y\right) \sqcap \tau^{n_m+1}_{2,a_m}\left(x\|^{\phi,l}_{A,2} y_m\right)\right)\right),$
where $\phi{=}\omega\setminus\{(a_m, n_m)\}_{m\in(I\cap J)\setminus K}\cup\{(a_m, n_m{+}1)\}_{m\in(I\cap J)\setminus M},$ if $C \not\subseteq A$

(P22) $x \Downarrow^{\omega,l}_{A,2} y = 0,$ if $C \subseteq A$

Figure 8.13: Axioms for parallel composition, part II

Proposition 8.5.2 (Soundness)**.** *Let p and q be* CG *process trees. If* $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}} \vdash p = q$*, then $p \approx_{\mathsf{RT}} q$.*

Proof. Straightforward. □

The following theorem states that two ready-trace equivalent CG process trees can be reduced to almost-equal process trees using the axioms of theory $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}}$.

Theorem 8.5.3 (Completeness)**.** *Let p and q be* CG *process trees such that $p \approx_{\mathsf{RT}} q$. There exist* CG *process trees p', g', such that* $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}} \vdash p = p'$, $\mathrm{CSP}^{\mathrm{g}}_{\mathrm{p}} \vdash q = q'$*, and $p' \simeq q'$.*

Proof. By Proposition 8.3.15 and Proposition 8.5.1, it is enough to show that each of the general-process tree transformation steps given in Definition 8.3.2

can be mimicked by the axioms A1–A10 (or, more concretely, axioms A5, A6, A7 and A10). Steps (i) can be mimicked by axiom A5, step (ii) by axiom A6, step (iii) by axioms A6 and A10, step (iv) by using first axiom A10 in a right-to-left direction and then A7, while step (v) can be performed by using first axiom A10 in a right-to-left direction and then A6. □

Note that an infinite set of axioms is required in order to reduce two ready-trace equivalent CG process trees to isomorphic CG process trees, which is why we restrict ourselves to deriving almost-equal process trees. However, the purpose of the present text was to obtain an algebra in which the CSP laws are preserved under presence of probabilistic choice, and in which no new laws regarding the interplay between the different choice operators are added. We leave the decidability problem for the future.

Chapter 9

Concluding remarks to part II

We discuss related work to the results presented in this part, and end with concluding remarks.

9.1 Related work

As closely related to the results presented in Part II we consider the research reports that face the challenge of defining a satisfactory linear-time semantics for concurrent systems with probabilistic and nondeterministic choice; by linear-time here we mean allowing distribution of prefix over internal (probabilistic) choice. In the introduction we discussed why this problem is non-trivial. Work addressing this problem was reported in [83], [101], [97], [87], [63], [78], [38], [30], [32], [82], [33], [47], [31], [1]. Our work is also closely related to the research oriented towards restricting the power of the schedulers for concurrent probabilistic-nondeterministic systems [9, 31, 32, 38, 55], to achieve compositionality for trace preorders [32, 38], or to obtain realistic probabilities for the behaviour of the system [9, 31, 55].

The report [83] defines trace-style semantical equivalences for processes with action choice and probabilistic choice; they are extended for processes with internal nondeterminism such that first the internal nondeterminism is resolved using the almighty (randomized) schedulers discussed in the introduction, and then the processes are compared. Not surprisingly, the equivalences are not congruences, for example they do not equate processes $x \| y$ and $\bar{x} \| y$ from Example 6.0.2 in the introduction. It is interesting to note that reference [83] is the earliest one that notes the problem discussed in this example. Similar problems with compositionality, induced by the underlying almighty schedulers, appear in the trace equivalences defined in [97] and [33]; the latter one characterizes the former one via a testing scenario. We note

that the compositionality problem in [97] remains even when restricted to processes without internal nondeterminism, as processes $p|||d$ and $q|||d$ from Fig. 6.4 are not equated (although p and q are) or, in other words, action choice does not distribute over probabilistic choice.

References [101], [82], and [47] consider processes without internal nondeterminism and define equivalences that equate two processes only if they cannot be distinguished by the environment. However, the environment is not a process itself (as in our case), but rather only a sequence of actions. As a result, although [101], [82], and [47] allow distributivity of external action choice over probabilistic choice, they also make undesirable identifications from the point of view of process theory; for example, they equate processes $\frac{1}{2}ca \oplus \frac{1}{2}cb$ and $\frac{1}{2}c(a \,\Box\, b) \oplus \frac{1}{2}c$.

In [87] a process is a probability distribution over standard CSP processes; two processes are equivalent if the probability distributions are the same. In other words, all the probabilistic choices are resolved before the execution of the process and a resolution is a standard non-probabilistic process. By "lifting" the probabilistic choices to the root, the nondeterministic choices are "pushed" downwards and replicated. As discussed in the introduction, the replication of the nondeterministic choices leads to loss of probability information. With this approach also the idempotence of the internal choice is lost [87], viz. the law $x = x \sqcap x$ does not hold, and action choice does not distribute over probabilistic choice. Similarly, in [1] first the probabilistic choices are resolved and the resolutions are compared under the standard may-testing semantics of [39] (in [1], processes do not contain internal nondeterminism). Again, action choice does not distribute over probabilistic choice, which leads to compositionality problems. The same problems also occur in the testing equivalence of [30], defined with the unrestricted schedulers, similar to the testing semantics of [108] discussed in the introduction.

The loss of idempotence for internal choice was overcome by the button-pushing testing equivalence defined in [78]; however, here still congruence for parallel composition could not be achieved [79], viz. action choice does not distribute over probabilistic choice.

Reference [63] defines a ready-trace equivalence for processes given in denotational semantics, with action choice and internal probabilistic choice, that is, without internal nondeterminism. We conjecture that the ready-trace equivalence of [63] coincides with the one defined here, when restricted to processes without internal nondeterminism. However, our definition, unlike [63], yields a black-box testing scenario in the style of [58], with which the equivalence can be determined. Namely, in our case, if the observer can see the action menus at every moment, then she can deduce the conditional probability of a ready-trace via a simple statistical procedure.

The definition of ready-trace equivalence given in [83] does not yield a testing scenario, since the function that computes the probability of every ready-trace does not yield a probability measure.

The results from the above research reports indicate that mixing probabilistic and nondeterministic choice in concurrent processes creates compositionality problems for linear-time equivalences, no matter whether in order to establish equivalence the internal choices are resolved first [33, 83, 97], or the probabilistic choices are resolved first [1, 87]. This observation, together with the problem with overestimation of probabilities under almighty schedulers explained in Example 6.0.2, and noted already in [83, 87, 97], led to a new research direction, aiming to restrict the power of the schedulers that used to resolve the nondeterminism. Work in this direction has been reported in [31, 32, 38, 55].

Reference [38] is the first one that addresses the problem with compositionality for trace semantics [97] in probabilistic systems. The parallel composition in [38] is synchronous – each time the composed system performs a step, all the components perform a step. The states of the system are valuations over a set of variables, and the information available to each component can be modeled by restricting the variables it is able to read. Reference [32] considers asynchronous systems, and restricts the power of the schedulers in a parallel composition, also to obtain compositionality for a trace equivalence for probabilistic systems distinguishing between input and output actions [110]. When components are composed, the local nondeterminism in a component is resolved based only on the history of the component itself; the global nondeterminism, arising from the choice on which component will generate the next output action on which (the rest of) the components synchronize, is resolved by the components themselves, which pass a token one to another. A component that holds the token decides to which component to forward the token based on its own local history. Alternatively, it is discussed in [32], the global nondeterminism can be resolved by a centralized component-scheduler, which resolves the nondeterminism based on the global history of the composition. In [55] it is observed that such a scheduler, that resolves the global interleaving nondeterminism based on the complete history, may be still too powerful. A restriction is added to the interleaving scheduler in [55], such that it cannot use information from one component in order to decide between two other components.

Comparing our approach to [38] and [32], we can conclude that a common feature, that enables compositional reasoning under linear-time semantics, is that the local nondeterminism in a component is resolved based on local information only. The three approaches differ in the resolution of the global

nondeterminism. While the focus in [38] and [32] is on resolving nondeterminism in special types of concurrent systems, our focus was on deriving a satisfactory extension of a common process language such as CSP. The result was a ready-trace semantics, finer than trace semantics, which conservatively extends CSP with a probabilistic choice.

It is also interesting to see how the schedulers that resolve the interleaving nondeterminism in our case fit with the strongly distributed reasoning of [55]. However, without distinguishing between input and output actions, we cannot apply the reasoning of [55], as the strongly distributed interleaving scheduler cannot choose between input actions. However, the focus in [54,55] is not on devising compositionality, but on improving the probabilistic verification techniques, by obtaining as realistic estimates of probabilities as possible. In the present work, among other things, we were interested in the question "how much freedom can the schedulers retain such that compositionality for linear-time equivalence is preserved?".

In [27] an algorithm is proposed for computing (time-bounded) optimal reachability probabilities with respect to the distributed schedulers of [32] and [55]. The algorithm is based on an interpretation of the model as a parametric Markov chain, such that each state in the latter is a path in the original probabilistic system. The idea is that the scheduler decisions are parameters of the parametric Markov chain, similarly as here. However, there is a conceptual difference between the two approaches. Namely, we are interested in integrating the schedulers in the model itself for a compositional reasoning, and the external choice is left to be resolved by the environment; on the other hand, in [27] the interest is on resolving all nondeterminism in a distributed system in order to compute the bounds of the reachability probabilities, and composing the parametric Markov chains is not an issue. It is interesting that both papers [50] and [27], that propose integrating the scheduler information into labels, appeared at approximately the same time.

The paper [31] takes a dual approach to the previous approaches, by introducing an *explicit* scheduler that communicates to processes via labels: two sub-processes in the process are indistinguishable to a scheduler if they have the same labels. Thus, by a suitable labeling, the modeler specifies which internal or random choices are visible to the scheduler and which are not. When compared to our and all other approaches, the method in [31] allows for a greater flexibility in equating processes. For example, whether processes x and $\bar{x}$ in Fig. 6.2 are testing-equivalent depends on the labeling system. Note that this flexibility means that more responsibility is being delegated to the modeler. In our case, on the other hand, it is enough to ensure only that no label appears twice before processes are composed; as the processes start composing, the labels identify which internal choices are multiple instances

of the same internal choice, or in terms of [31], which random choices are invisible to the scheduler. Moreover, new labels are automatically assigned to the nondeterminism arising from parallelism, reflecting the information based on which this nondeterminism is resolved, and thus serving to identify multiple instances of an internal choice. Also, while in [31] the labels serve to navigate the scheduler and thus the scheduler is deterministic, in our case the labels represent unknown probabilities, yielding randomized schedulers.

The power of the schedulers for verification of security properties has been also restricted in [9]; in this paper, tagged probabilistic automata are defined, such that the transitions in a parallel composition are tagged with the identifier of the component that originates the transition, or with a pair of components if synchronization has happened. At each point of the execution, the scheduler for the composition can see the current tags and the history of chosen tags and performed actions. Thus, this scheduler has similarities to the schedulers defined in Section 8.2, showing once again how approaches with different motivations can converge independently to similar solutions. The framework of [9] has been extended with internal nondeterminism in [2] and corresponding process equivalences (bisimulation and completed trace equivalence), together with a congruence result for bisimulation, have been given in [2].

Finally, note that restricting the power of schedulers that resolve the nondeterminism, for the purpose of realistic modeling, is a research topic in areas other than concurrency theory itself; for example in security (e.g. [28]), in operation research (see e.g. [103]), and in artificial intelligence (e.g. [75]). We do not make formal comparison to work in those areas, as, reasonably, they target either more specific, or different problems.

9.2 Concluding remarks

In this part, we have proposed a new probabilistic extension of may/must testing theory [39] for concurrent processes. The motivation was that the existing approaches for probabilistic may/must testing [42, 44, 74, 90, 98, 108], based on the all-mighty schedulers for resolving nondeterminism, yielded unrealistic probabilities with which a process passes a test. As a result, they equated too few processes when compared to [39], or, in other words, the internal probabilistic choice was observable. We have shown that our testing preorder can be characterized with a probabilistic ready-trace preorder relation, by which the exact moment in time in which a probabilistic or an internal choice happens is unobservable. In order to obtain realistic probabilities to pass a test, in our model the internal transitions are augmented

with labels. The labels represent the information based on which the internal nondeterminism is resolved and thus restrict the schedulers. For finite processes, our model is at least as expressible as the model of probabilistic automata [97], or the alternating model of probabilistic systems [68], since each finite process from the latter two can be mapped in our model by giving different labels to all internal transitions.

We have also defined a generalized parallel composition for our model, such that processes synchronize on a set of actions and interleave on the rest of the actions, as in CSP [96], but also the synchronized actions are hidden, as in CCS [85]. We have shown that our ready-trace proerder is a precongruence, that is, is preserved under parallel composition. Based on the ready-trace equivalence and the new parallel composition, we have given a probabilistic extension of CSP, which preserves all the nice properties of internal choice from CSP, such as the distributivity laws and idempotence. Thus, with our approach we have also solved another open problem, namely to give a satisfactory probabilistic extension of CSP, that respects the original laws.

9.2.1 Discussion and future work

The results presented here open several questions that can be addressed in the future. Namely, by the present definition of our parallel composition, in general finite processes are obtained, since every internal choice must be labeled by the information based on which it is resolved, which is usually a history of execution. Thus, it would be interesting to see whether the unfolding could be bypassed in certain cases and whether a finite representation of infinite processes can be obtained. Also, in the present setting we limited ourselves to processes without divergence. There are several ways to treat divergence: one can treat it is as an error in the model (e.g. [39, 71]), or can abstract away from it by assuming fairness (e.g. [11, 85, 95]), or can allow it, but distinguish between processes with and without divergence (see e.g. [61]). In the present setting, divergent processes cannot be obtained by parallel composition; divergence in the non-composed processes could be treated by assuming that the divergence transitions are probabilistic (implying fairness). Then, cycles of probabilistic transitions are easily handled using Markov Chain theory [72]. However, if a parallel composition that yields cyclic processes is defined, then an internal action may be obtained by synchronization and divergence would require a different treatment.

By now, it is clear that defining a separate hiding operator together with probabilistic choice operator does not fit well with our goal of retaining the probabilities: it turns external choice into internal choice and gives even more

(unrealistic) power to the schedulers. Technically speaking, in our model the internal actions resulting from hiding would have to be labeled with the information based on which the corresponding nondeterminism is resolved, but such information does not exist. However, in the non-probabilistic case the hiding operator is useful to abstract away from unimportant information [11]. A compromise that could be made is to hide the actions *before* processes are composed, to abstract away from details, and/or *after* the complete model is obtained by composing, in order to reduce the model.

Another problem that needs to be addressed is the decidability of our ready-trace equivalence. We have defined it using randomized schedulers for the resolution of the nondeterminism, which are in general more powerful than the deterministic schedulers, i.e. those that assign trivial probabilities from the set $\{0, 1\}$ to the alternatives. Indeed, had we defined our equivalence by deterministic schedulers, then it would have distinguished less. For example, processes $(\tau_1 a \sqcap \tau_2 b)|||(\tau_3 c \sqcap \tau_4 d)$ and $\tau_5(a|||c) \sqcap \tau_6(a|||d) \sqcap \tau_7(b|||c) \sqcap \tau_8(b|||d)$ would be equivalent; by our definition, they are distinguished, as the resolution of the later process such that $\tau_5 = \tau_8 = 0.5$ and $\tau_6 = \tau_7 = 0$ cannot be mimicked by any resolution of the former process. Interestingly, if we ignore the internal labels, those two processes are equated by the failures semantics of CSP [71]. Thus, we anticipate that if we restrict to deterministic schedulers, no new equalities w.r.t. CSP would arise. Therefore, restricting to deterministic schedulers for decidability reasons should be reasonable.

We conclude the discussion by describing an algorithm for composing processes, that reveals that the way we define our restricted schedulers is very suitable for modeling a hierarchy of compositions of processes (a component of a system is usually itself made of several simpler components). Namely, the crucial point of our labeling system is that the labels of the internal choices that are obtained as a result of hiding after synchronization or as a result of parallelism include all the necessary information for resolving the nondeterminism. Thus, when the composed process is put in parallel with another process, the dependencies between the internal choices are automatically preserved. Note that if the labels of the internal transitions of a process are bijectively mapped to new labels, the dependencies between the internal choices are preserved, which is all that is needed when the composed process is run in parallel. Thus, in order to compose finite processes, all that needs to be done is to assign different labels to the simple (not composed) processes, compose the processes in parallel, map the label set of the composed process into a new label set, and then compose it again in parallel with other processes.

Bibliography

[1] L. Acciai, M. Boreale, and R. De Nicola. Linear and may-testing semantics in a probabilistic reactive setting. In *FMOODS-FORTE'11*, LNCS 6722, pages 29–43. Springer, 2011.

[2] M. S. Alvim, M. E. Andrés, C. Palamidessi, and P. van Rossum. Safe equivalences for security properties. In *IFIP TCS'10*, pages 55–70, 2010.

[3] S. Andova. *Probabilistic Process Algebra*. PhD thesis, Eindhoven University of Technology, 2002.

[4] S. Andova, J. Baeten, P. D'Argenio, and T. Willemse. A compositional merge of probabilistic processes in the alternating model. In *NWPT '06*, pages 1–4. Reykjavik, Islandia, 2006.

[5] S. Andova, J. Baeten, and T. Willemse. A complete axiomatisation of branching bisimulation for probabilistic systems with an application in protocol verification. In *CONCUR '06*, pages 327–342. LNCS 4137, Springer, 2006.

[6] S. Andova and S. Georgievska. On compositionality, efficiency, and applicability of abstraction in probabilistic systems. In *SOFSEM 2009*, pages 67–78. LNCS 5404, Springer, 2009.

[7] S. Andova, S. Georgievska, and N. Trčka. Branching bisimulation congruence for probabilistic systems. *Theoretical Computer Science* (2011), DOI:10.1016/j.tcs.2011.07.020.

[8] S. Andova and T. Willemse. Branching bisimulation for probabilistic systems: characteristics and decidability. *Theoretical Computer Science*, 356(3):325–355, 2006.

[9] M. E. Andrés, C. Palamidessi, P. v. Rossum, and A. Sokolova. Information hiding in probabilistic concurrent systems. *Theoretical Computer Science*, 412(28):3072–3089, 2011.

[10] J. Baeten and W. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[11] J. C. M. Baeten, T. Basten, and M. A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*. Cambridge University Press, 2010.

[12] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, 9(2):127–168, 1986.

[13] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Ready-trace semantics for concrete process algebra with the priority operator. *The Computer Journal*, 30(6):498–506, 1987.

[14] J. C. M. Baeten, J. A. Bergstra, and S. A. Smolka. Axiomization of probabilistic processes: ACP with generative probabililties (extended abstract). In *CONCUR'92*, LNCS 630, pages 472–485. Springer, 1992.

[15] J. C. M. Baeten and M. Bravetti. A ground-complete axiomatization of finite state processes in process algebra. In *CONCUR '05*, pages 248–262. LNCS 3653, Springer, 2005.

[16] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.

[17] E. Bandini and R. Segala. Axiomatizations for probabilistic bisimulation. In *ICALP '01*, pages 370–381. LNCS 2076, Springer, 2001.

[18] T. Basten. Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996.

[19] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.

[20] J. Bergstra and J. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[21] J. Bergstra and J. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[22] J. Bergstra, A. Ponse, and M. van der Zwaag. Branching time and orthogonal bisimulation equivalence. *Theoretical Computer Science*, 309(1):313–355, 2003.

[23] A. Bianco and L. de Alfaro. Model checking of probabalistic and nondeterministic systems. In *FSTTCS '95*, pages 499–513. LNCS 1026, Springer, 1995.

[24] R. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43:863–914, 1996.

[25] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of ACM*, 31(3):560–599, 1984.

[26] M. Browne, E. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1-2):115–131, 1988.

[27] G. Calin, P. Crouzen, P. R. D'Argenio, E. M. Hahn, and L. Zhang. Time-bounded reachability in distributed input/output interactive probabilistic chains. In *SPIN'10*, LNCS 6349, pages 193–211. Springer, 2010.

[28] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *FOCS'01*, pages 136–145. IEEE, 2001.

[29] S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *CONCUR '02*, pages 371–385. LNCS 2421, Springer, 2002.

[30] D. Cazorla, F. Cuartero, V. Valero, F. L. Pelayo, and J. J. Pardo. Algebraic theory of probabilistic and nondeterministic processes. *Journal of Logic and Algebraic Programming*, 55(1-2):57–103, 2003.

[31] K. Chatzikokolakis and C. Palamidessi. Making random choices invisible to the scheduler. *Information and Computation*, 208(6):694–715, 2010.

[32] L. Cheung, N. Lynch, R. Segala, and F. Vaandrager. Switched PIOA: Parallel composition via distributed scheduling. *Theoretical Computer Science*, 365(1-2):83–108, 2006.

[33] L. Cheung, M. I. A. Stoelinga, and F. W. Vaandrager. A testing scenario for probabilistic processes. *Journal of ACM*, 54(6):29:1–29:45, 2007.

[34] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR'90*, LNCS 458, pages 126–140, 1990.

[35] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, London, UK, 1982. Springer.

[36] N. Coste, H. Hermanns, E. Lantreibecq, and W. Serwe. Towards performance prediction of compositional models in industrial GALS designs. In *CAV'09*, pages 204–218. LNCS 5643, Springer, 2009.

[37] P. R. D'Argenio, H. Hermanns, and J.-P. Katoen. On generative parallel composition. In *PROBMIV'98*, ENTCS 22, pages 30–54. Elsevier, 1999.

[38] L. de Alfaro, T. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *CONCUR'01*, LNCS 2154, pages 351–365. Springer, 2001.

[39] R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[40] R. De Nicola and F. Vaandrager. Three logics for branching bisimulation. *Journal of ACM*, 42(2):458–487, 1995.

[41] Y. Deng, C. Palamidessi, and J. Pang. Compositional reasoning for probabilistic finite-state behaviors. In *Processes, Terms and Cycles: Steps on the Road to Infinity*, pages 309–337. LNCS 3838, Springer, 2005.

[42] Y. Deng, R. van Glabbeek, M. Hennessy, and C. Morgan. Testing finitary probabilistic processes (extended abstract). In *CONCUR'09*, LNCS 5710, pages 274–288, 2009.

[43] Y. Deng, R. van Glabbeek, M. Hennessy, C. Morgan, and C. Zhang. Remarks on testing probabilistic processes. In *Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin*, ENTCS 172, pages 359–397, 2007.

[44] Y. Deng, R. J. van Glabbeek, M. Hennessy, and C. Morgan. Characterising testing preorders for finite probabilistic processes. *Logical Methods in Computer Science*, 4(4:4):1–33, 2008.

[45] J. Desharnais, V. Gupta, J. R, and P. Panangaden. Weak bisimulation is sound and complete for PCTL*. In *CONCUR 2002*, pages 355–370. LNCS 2421, Springer, 2002.

[46] J. L. Doob. *Stochastic Processes*. New York: John Wiley and Sons, 1953.

[47] L. Doyen, T. A. Henzinger, and J.-F. Raskin. Equivalence of labeled Markov chains. *International Journal of Foundations of Computer Science*, 19(3):549–563, 2008.

[48] H. Garavel and M. Sighireanu. A graphical parallel composition operator for process algebras. In *FORTE XII / PSTV XIX '99*, pages 185–202. Kluwer, B.V., 1999.

[49] S. Georgievska and S. Andova. Composing systems while preserving probabilities. In *EPEW 2010*, pages 268–283. LNCS 6342, Springer, 2010.

[50] S. Georgievska and S. Andova. Retaining the probabilities in probabilistic testing theory. In *FOSSACS'10*, LNCS 6014, pages 79–93, 2010.

[51] S. Georgievska and S. Andova. Testing reactive probabilistic processes. In *QAPL'10*, EPTCS 28, pages 99–113, 2010.

[52] S. Georgievska and S. Andova. Probabilistic CSP: Preserving the laws via restricted schedulers. Submitted, 2011.

[53] A. Giacalone, C.-C. Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods*, pages 453–459. North-Holland, 1990.

[54] S. Giro. *On the automatic verification of distributed probabilistic automata with partial information*. PhD thesis, Universidad Nacional de Córdoba, 2010.

[55] S. Giro and P. D'Argenio. On the expressive power of schedulers in distributed probabilistic systems. In *QAPL'09*, ENTCS 253(3), pages 45–71. Elsevier, 2009.

[56] S. Giro and P. R. D'Argenio. Quantitative model checking revisited: neither decidable nor approximable. In *FORMATS'07*, LNCS 4763, pages 179–194. Springer-Verlag, 2007.

[57] S. Giro, P. R. D'Argenio, and L. M. Ferrer Fioriti. Partial order reduction for probabilistic systems: A revision for distributed schedulers. In *CONCUR'09*, LNCS 5710, pages 338–353. Springer-Verlag, 2009.

[58] R. J. v. Glabbeek. The linear time – branching time spectrum I; The semantics of concrete, sequential processes. In *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001.

[59] R. J. v. Glabbeek. The meaning of negative premises in transition system specifications ii. *Journal of Logic and Algebraic Programming*, 60-61:229–258, 2004.

[60] R. v. Glabbeek. What is branching time semantics and why to use it? In *The Concurrency Column*, pages 190–198. *Bulletin of the EATCS* 53, 1994.

[61] R. v. Glabbeek, B. Luttik, and N. Trčka. Branching bisimilarity with explicit divergence. *Fundamenta Informaticae*, 93:371–392, 2009.

[62] R. v. Glabbeek and P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of ACM*, 43(3):555–600, 1996.

[63] F. C. Gomez, D. De Frutos Escrig, and V. V. Ruiz. A sound and complete proof system for probabilistic processes. In *ARTS'97*, LNCS 1231, pages 340–352. Springer, 1997.

[64] J. Groote and F. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In *ICALP'90*, pages 626–638. LNCS 443, Springer, 1990.

[65] J. F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.

[66] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Real-Time Systems Symposium '90*, pages 278 –287, 1990.

[67] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:102–111, 1994.

[68] H. A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. Elsevier, 1994.

[69] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[70] H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*. LNCS 2428. Springer, 2002.

[71] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[72] R. Howard. *Semi-Markov and Decision Processes*. London: Wiley, 1971.

[73] R. A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. Press, 1960.

[74] B. Jonsson and Y. Wang. Testing preorders for probabilistic processes can be characterized by simulations. *Theoretical Computer Science*, 282(1):33–51, 2002.

[75] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence Journal*, 101:99–134, 1998.

[76] J.-P. Katoen, T. Kemna, I. Zapreev, and D. N. Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS'07*, pages 87–101. LNCS 4424, Springer, 2007.

[77] K. N. Kumar, R. Cleaveland, and S. A. Smolka. Infinite probabilistic and nonprobabilistic testing. In *FSTTCS'98*, LNCS 1530, pages 209–220. Springer, 1998.

[78] M. Kwiatkowska and G. Norman. A testing equivalence for reactive probabilistic processes. In *EXPRESS'98*, ENTCS 16(2), pages 1–19. Elsevier, 1998.

[79] M. Z. Kwiatkowska and G. J. Norman. A fully abstract metric-space denotational semantics for reactive probabilistic processes. In *COMPROX '98*, ENTCS 13, pages 1–33. Elsevier, 1998.

[80] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[81] D. V. Lindley. *Introduction to Probability and Statistics from a Bayesian Viewpoint*. Cambridge University Press, 1980.

[82] N. López, M. Núñez, and I. Rodríguez. Specification, testing and implementation relations for symbolic-probabilistic systems. *Theoretical Computer Science*, 353(1):228–248, 2006.

[83] G. Lowe. Representing nondeterministic and probabilistic behaviour in reactive processes. Technical Report PRG-TR-11-93, Oxford University Computing Labs, 1993.

[84] N. Lynch, R. Segala, and F. Vaandrager. Observing branching structure through probabilistic contexts. *SIAM Journal on Computing*, 37(4):977–1013, 2007.

[85] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag, 1980.

[86] R. Milner. Operational and algebraic semantics of concurrent processes. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 1201–1242. Elsevier and MIT Press, 1990.

[87] C. Morgan, A. McIver, K. Seidel, and J. W. Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–647, 1996.

[88] S. Nain and M. Y. Vardi. Branching vs. linear time: Semantical perspective. In *ATVA'07*, pages 19–34. LNCS 4762, Springer, 2007.

[89] R. D. Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24(2):211–237, 1987.

[90] M. C. Palmeri, R. D. Nicola, and M. Massink. Basic observables for probabilistic may testing. In *QEST '07*, pages 189–200. IEEE Computer Society, 2007.

[91] A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In *CONCUR '00*, pages 334–349. LNCS 1877, Springer, 2000.

[92] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *ICALP'85*, LNCS 194, pages 15–32, 1985.

[93] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1:53–72, 1986.

[94] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[95] A. Rensink and W. Vogler. Fair testing. *Information and Computation*, 205:125–198, 2007.

[96] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

[97] R. Segala. *Modeling and Verification of Randomized Distributed Real-time Systems*. PhD thesis, MIT, 1995.

[98] R. Segala. Testing probabilistic automata. In *CONCUR'96*, LNCS 1119, pages 299–314. Springer, 1996.

[99] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[100] R. Segala and A. Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In *QEST'05*, pages 44–53. IEEE Computer Society, 2005.

[101] K. Seidel. Probabilistic communicating processes. *Theoretical Computer Science*, 152:219–249, 1995.

[102] A. Sokolova and E. d. Vink. Probabilistic automata: system types, parallel composition and comparison. In *Validation of Stochastic Systems: A Guide to Current Research*, pages 1–43. LNCS 2925, Springer, 2004.

[103] E. J. Sondik. *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University, 1971.

[104] N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. PhD thesis, Eindhoven University of Technology, 2007.

[105] N. Trčka and S. Georgievska. Branching bisimulation congruence for probabilistic systems. In *QAPL'08*, pages 129–143. ENTCS 220(3), Elsevier, 2008.

[106] N. Trčka, S. Georgievska, J. Markovski, S. Andova, and E. de Vink. Performance analysis of χ models using discrete-time probabilistic reward graphs. In *WODES'08*, pages 113 –118. IEEE XPlore, 2008.

[107] M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS'85*, pages 327–338. IEEE Computer Society Press, 1985.

[108] Y. Wang and K. G. Larsen. Testing probabilistic and nondeterministic processes. In *Proceedings of the IFIP TC6/WG6.1 Twelth International Symposium on Protocol Specification, Testing and Verification XII*, pages 47–61, 1992.

[109] V. Wolf. Testing theory for probabilistic systems. In *Model-Based Testing of Reactive Systems*, LNCS 3472, pages 233–275. 2005.

[110] S.-H. Wu, S. Smolka, and E. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176(1–2):1–38, 1997.

Curriculum vitae

Sonja Georgievska was born on 10th of January 1979 in Strumica, Macedonia (then Yugoslavia). She studied computer science at the Faculty of Sciences and Mathematics within the Ss. Cyril and Methodius University in Skopje, Macedonia, and obtained the degrees of Graduated Engineer and M.Sc. in informatics in 2001 and in 2007, respectively. From 2001 until 2007 she worked as a teaching/research assistant at the same institution. In June 2007 she started a PhD project in the Department of Mathematics and Computer Science at the Eindhoven University of Technology in The Netherlands of which the results are presented in this dissertation.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue*. Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms*. Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications*. Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures*. Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability*. Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions*. Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments*. Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression*. Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages*. Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Pro-*

cesses with Replication. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*.

Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*.

Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains.* Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use.* Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems.* Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML.* Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery.* Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding.* Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks.* Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data.* Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy.* Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS.* Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code.* Faculty

of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates*. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments*. Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design*. Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems*. Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms*. Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation*. Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems*. Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification*. Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development*. Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation*. Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean*. Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques*. Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time*. Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation*. Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML*. Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web*. Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks*. Faculty of Science, UU. 2009-23

T.K. Cocx. *Algorithmic Tools for Data-Oriented Law Enforcement*. Faculty of Mathematics and Natural Sciences, UL. 2009-24

A.I. Baars. *Embedded Compilers*. Faculty of Science, UU. 2009-25

M.A.C. Dekker. *Flexible Access Control for Dynamic Collaborative Environments*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-26

J.F.J. Laros. *Metrics and Visualisation for Crime Analysis and Genomics*. Faculty of Mathematics and Natural Sciences, UL. 2009-27

C.J. Boogerd. *Focusing Automatic Code Inspections*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2010-01

M.R. Neuhäußer. *Model Checking Nondeterministic and Randomly Timed Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-02

J. Endrullis. *Termination and Productivity*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-03

T. Staijen. *Graph-Based Specification and Verification for Aspect-Oriented Languages*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-04

Y. Wang. *Epistemic Modelling and Protocol Dynamics*. Faculty of Science, UvA. 2010-05

J.K. Berendsen. *Abstraction, Prices and Probability in Model Checking Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2010-06

A. Nugroho. *The Effects of UML Modeling on the Quality of Software*. Faculty of Mathematics and Natural Sciences, UL. 2010-07

A. Silva. *Kleene Coalgebra*. Faculty of Science, Mathematics and Computer Science, RU. 2010-08

J.S. de Bruin. *Service-Oriented Discovery of Knowledge - Foundations, Implementations and Applications*. Faculty of Mathematics and Natural Sciences, UL. 2010-09

D. Costa. *Formal Models for Component Connectors*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-10

M.M. Jaghoori. *Time at Your Service: Schedulability Analysis of Real-Time and Distributed Services*. Faculty of Mathematics and Natural Sciences, UL. 2010-11

R. Bakhshi. *Gossiping Models: Formal Analysis of Epidemic Protocols*. Faculty of Sciences, Department of Computer Science, VUA. 2011-01

B.J. Arnoldus. *An Illumination of the Template Enigma: Software Code Generation with Templates*. Faculty of Mathematics and Computer Science, TU/e. 2011-02

E. Zambon. *Towards Optimal IT Availability Planning: Methods and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-03

L. Astefanoaei. *An Executable Theory of Multi-Agent Systems Refinement*. Faculty of Mathematics and Natural Sciences, UL. 2011-04

J. Proença. *Synchronous coordination of distributed components*. Faculty of Mathematics and Natural Sciences, UL. 2011-05

A. Moralı. *IT Architecture-Based Confidentiality Risk Assessment in Networks of Organizations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-06

M. van der Bijl. *On changing models in Model-Based Testing*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-07

C. Krause. *Reconfigurable Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-08

M.E. Andrés. *Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2011-09

M. Atif. *Formal Modeling and Verification of Distributed Failure Detectors*. Faculty of Mathematics and Computer Science, TU/e. 2011-10

P.J.A. van Tilburg. *From Computability to Executability – A process-theoretic view on automata theory*. Faculty of Mathematics and Computer Science, TU/e. 2011-11

Z. Protic. *Configuration management for models: Generic methods for model comparison and model co-evolution*. Faculty of Mathematics and Computer Science, TU/e. 2011-12

S. Georgievska. *Probability and Hiding in Concurrent Processes*. Faculty of Mathematics and Computer Science, TU/e. 2011-13