Models and axioms for a fragment of real time process algebra

A.S. Klusener

Models and axioms for a fragment of real time process algebra

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven op gezag van de Rector Magnificus, prof.dr. J.H. van Lint voor een commissie aangewezen door het College van Dekanen in het openbaar te verdedigen op vrijdag 10 december 1993 om 16.00 uur.

door

Anton Stefanus Klusener

geboren te Nieuwveen

Dit proefschrift is goedgekeurd
door de promotoren:
prof.dr. J.C.M. Baeten
en
prof.dr. J.A. Bergstra

Cover by Philip Stroomberg.
Printed and bound by CopyPrint 2000, Enschede, The Netherlands.

The work in this thesis has been carried out at the CWI (Centrum voor Wiskunde en Informatica), Amsterdam in the context of ESPRIT Basic Research Projects no. 3006 (CONCUR) and no. 7166 (CONCUR 2).

Acknowledgements

After I finished my undergraduate studies in Computer Science at the University of Amsterdam (supervised by Jan Bergstra), I moved to Jos Baeten's process algebra group at the CWI in Amsterdam.

At the time of this move, Jan and Jos had just completed their initial paper on real time process algebra. Soon I was working on the problem of a complete axiomatization of a subset of their calculus. Initially I had little knowledge of logic and process algebra, hence I had only a rather vague intuition about how to solve the problems, and I had no idea how one could work for several years on this topic.

During the first year Jos guided me with patience about doing research on a mathematical subject. He has forced me to be precise and clear, without tempering my enthusiasm. I thank him for all of this.

Jan Bergstra is thanked for the stimulating way he manages practically and scientifically the process algebra group in Amsterdam that is distributed over the CWI and the University of Amsterdam.

It has been a great pleasure to work together with Willem Jan Fokkink, with whom I coauthored a great deal of the second part of this thesis. Willem Jan has been my sparring partner on whom I could test most of mine ideas.

This thesis could not have been written without the support and company of my colleagues at the CWI and the University of Amsterdam (Programming Research Group). Since any list of names would be incomplete I mention only: Inge Bethke, Doeko Bosscher, Claudia Brovedani, Jacob Brunekreef, Nicolien Drost, Willem Jan Fokkink, Rob van Glabbeek, Jan Friso Groote, Joris Hillebrand, Jan Willem Klop, Henri Korver, Sjouke Mauw, Alban Ponse, Piet Rodenburg, Frits Vaandrager, Gert Veltink, Chris Verhoef, Bas van Vlijmen, Jos van Wamel and Arjan van Waveren. Frits Vaandrager is also thanked for the careful manner he refereed this thesis. His nasty questions have forced me to understand and elaborate several elementary aspects in more detail; I hope to have inherited at least a small bit of his thoroughness. Chris Verhoef is thanked for his remarks on SOS formats.

I am also grateful to Kim Larsen for refereeing this thesis and for the valuable remarks he gave me. I have much appreciated to have (e-mail) conversations on real time process algebra with Liang Chen, Jim Davis, Alan Jeffrey, Matthew Hennessy, Jens Godskesen, Faron Moller, David Murphy, Juan Quemada, Roberto Segala and Yi Wang.

There is more in the world than real time process algebra. I have been lucky to also have done industrial research, within the Esprit project Atmosphere and the Race project BOOST. This thesis has been written outside the scope of these projects, but working in these projects helped me to give me a broader outlook on my work. Therefore, I thank the people of these projects, notably, Loe Feijs and Rob Vader (Philips Research, Eindhoven; Atmosphere) and Jim Macura and Robert Primrose (MARI, Gateshead, UK; BOOST).

Finally, I thank the department of Programming Technology of the CWI, headed by Jaco the Bakker, for the stimulating time I have had.

Abstract

Process algebra is the study of concurrent communicating processes in an algebraic framework. It has been initiated by Milner, who has developed the process algebra CCS [Mil80],[Mil89]. Bergstra and Klop presented in [BK84b] the process algebra ACP, for a textbook we refer to [BW90]. Another process algebra that we mention is CSP that has been developed by Hoare [Hoa85]. In the first part of this thesis we give a brief introduction to ACP, that is based on [BW90].

Over the last few years there have been several attemps to extend process algebra, which has resulted in higher order process algebra, process algebra with value passing and process algebra with time. Baeten and Bergstra have extended ACP with time by decorating the atomic actions with time stamps [BB91]. These time stamps are taken from some time domain, which may be the set of real numbers. For example, $a(t)$ is the process which executes the atomic action a at time t. They also introduced the process $\int_{v \in S} p(v)$, where v is a so-called time variable, S is a subset of the time domain, and p is a process expression in which the free occurrences of the time variable v become bound. This integrated process expression denotes the alternative composition over the time stamps in S.

In this thesis we study this approach in more detail. We restrict ourselves to *prefix integrated* process terms, that is, we will allow only processes of the form $\int_{v \in V} a(v)$ and $\int_{v \in V}(a(v) \cdot p)$, where V is an interval over the time domain. This restriction is necessary to obtain a tractable calculus for which in principle a complete axiomatization can be found. In the second part of this thesis we study the bisimulation semantics and axiom systems for real time ACP with prefix integration without recursion. The problem with process terms with prefixed integration, is to reason with terms containing free occurrences of time variables. To tackle this problem, we generalize our syntax. We allow terms of the form $\int_{\alpha}(a(v) \cdot p)$, where α is a condition, that is a boolean expression over time variables. We introduce a finite axiom system for this generalized class of terms, and we prove completeness and decidability of bisimulation equivalence. The material of this part originates from [BB91], [Klu91b] and [FK92], it has partly been written together with Willem Jan Fokkink.

In the literature several equivalences, and their characterizing laws, are known which deal with abstraction in process algebra without time. In the third part of this thesis we define branching, delay and weak bisimulation equivalence in the context of time, and we study the axiomatizations of these equivalences. Earlier versions of this work can be found in [Klu91a] and [Klu92].

Process algebra can be used in the specification and verification of protocols. In part four of this thesis we study guarded recursion in real time ACP in more detail, as it is an essential feature in the specification of protocols. In this part we show as well how a real time protocol can be verified, using the axioms for abstraction. The verification originates from [Klu91a].

In real time ACP consecutive actions can not happen at the same point in time, although they can occur arbitrary close to each other. In part five of this thesis

we present real time ACP with so called urgent actions, that are actions that may occur consecutively at the same point in time. By defining additional operators we can express phenomena like *maximal progress*. Finally, we show how other timed process algebras can be translated into this variant of real time ACP. In particular, we discuss the axioms for timed weak bisimulation that can be found in several other papers.

Contents

Part I

Introduction

1

An Introduction to Process Algebra

1.1 A short Overview of Process Algebra

In this chapter we give a short presentation of untimed Process Algebra. Most of this chapter is borrowed from [BW90].

A first objective of this chapter is, of course, to provide the reader who is not familiar with Process Algebra, and [BW90] in particular, with the background which is needed to read the rest of this thesis.

A second objective is to introduce notations, conventions and proof techniques that are used in this thesis and that can be explained without referring to time. Since some of the details of [BW90] cannot be extended easily to the timed setting of the following chapters we allow ourselves the freedom to deviate at minor points from [BW90].

In the first section we start with the presentation of BPAδ (Basic Process Algebra with δ), that is the syntax with atomic actions, the special constant δ for deadlock, alternative composition ($+$) and sequential composition ($\cdot$). We present in detail how the operational semantics of a process term, which will be a transition system, can be obtained from so-called action rules. Next, we present the axiom system of BPAδ and we discuss the notions of soundness and completeness. The completeness result will be based on so-called basic terms.

In the second section we extend BPAδ with the parallel composition operator ($\|$) and the auxiliary operators communication merge ($|$) and left merge ($\mathbin{\|\!\!\_}$). Together with the encapsulation operator (∂_H) we obtain ACP (Algebra of Communicating Processes). We show how these additional operators can be eliminated from a process term using the axioms of ACP.

In Section 3 we discuss Milner's silent action τ. This silent action can be used to abstract from internal actions. We discuss three associated equivalences, viz. branching bisimulation, delay bisimulation and weak bisimulation.

Finally, we present recursion in Section 4. We restrict ourselves mainly to the setting of a single finite guarded specification. We discuss the *Recursive Specification*

Principle (RSP) that says that if two process terms are both solutions of a recursive specification, then they are equal as well. We formulate this principle as a conitional axiom and we show that it is sound in the context of *guarded* specifications.

1.2 Basic Process Algebra

1.2.1 The Syntax for BPAδ

We have a, possibly infinite, alphabet A of *atomic actions* [1] and a special constant δ, denoting *deadlock*, which is not in A. The set $A \cup \{\delta\}$ is abbreviated by A_δ. The set of process terms over BPAδ is denoted by $T(\text{BPA}\delta)$, p, q and z will range over this set, which is defined by the following BNF sentence, where $a \in A_\delta$.

$$T(\text{BPA}\delta) \ : \quad p ::= a \mid p + p \mid p \cdot p$$

Here, $p + q$ is the alternative composition of p and q while $p \cdot q$ is the sequential composition of p and q. An element of $T(\text{BPA}\delta)$ is referred to as a process term. We use the standard convention that $\cdot$ binds stronger than $+$; thus $p \cdot q + p'$ is parsed as $(p \cdot q) + p'$.

The *size* of a process term p is the number of operators in p.

1.2.2 Action Rules and Transition Systems

The behavior of a process term p is represented by a labeled transition system with p as root. The states of a transition system are taken from $T(\text{BPA}\delta)$.

For each $a \in A$ there is a transition relation R_a, which is a binary relation over process terms. $R_a(p, p')$ is denoted by $p \xrightarrow{a} p'$ and it is called a transition. A transition $p \xrightarrow{a} p'$ denotes that the process p can evolve into the process p' by executing the atomic action a. The symbol $\surd$ denotes termination, for each $a \in A$ there is a predicate $R_a^{\surd}$. $R_a^{\surd}(p)$ is denoted by $p \xrightarrow{a} \surd$ and it means that the process p can terminate by executing the action a. By abuse of language we will call $p \xrightarrow{a} \surd$ a terminating transition or simply a transition. These transition relations and predicates are defined as the least ones satisfying the action rules of Table 1.1. In other words, $p \xrightarrow{a} p'$ if and only if it can be derived from the action rules. A derivation of a transition $p \xrightarrow{a} p'$ is a proof tree which is constructed from the action rules, two examples of these proof trees are given in Example 1.2.1. This style of giving operational semantics is advocated by Plotkin ([Plo81]).

Example 1.2.1 *Derivations for* $a \cdot (b + c) \xrightarrow{a} b + c$ *and* $a \cdot b + a \cdot c \xrightarrow{a} b$.

[1]In the ACP literature, viz. [BW90], it is common to require that A is finite. In that case, an axiom scheme that is parameterized by an action can be considered as an abbreviation of a finite axiom system. In the timed setting we will need axiom schemes which are parameterized with time stamps. Since the underlying time domain will be infinite these axiom schemes cannot be considered as an abbreviation of a finite axiom system. Hence, it does not help here to require that A is finite.

atom	$a \xrightarrow{a} \surd$		
seq_0	$\dfrac{p \xrightarrow{a} \surd}{p \cdot q \xrightarrow{a} q}$	seq_1	$\dfrac{p \xrightarrow{a} p'}{p \cdot q \xrightarrow{a} p' \cdot q}$
$plus_0^l$	$\dfrac{p \xrightarrow{a} \surd}{p + q \xrightarrow{a} \surd}$	$plus_1^l$	$\dfrac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$
$plus_0^r$	$\dfrac{p \xrightarrow{a} \surd}{q + p \xrightarrow{a} \surd}$	$plus_1^r$	$\dfrac{p \xrightarrow{a} p'}{q + p \xrightarrow{a} p'}$

$a \in A$

Table 1.1: Action Rules for BPAδ

$$seq_0 \; \frac{atom \quad a \xrightarrow{a} \surd}{a \cdot (b + c) \xrightarrow{a} b + c} \qquad plus_1^l \; \frac{seq_0 \; \dfrac{atom \quad a \xrightarrow{a} \surd}{a \cdot b \xrightarrow{a} b}}{a \cdot b + a \cdot c \xrightarrow{a} b}$$

1.2.3 Bisimulation Equivalence

We may identify process terms of which the transition systems represent the same behavior. We define strong bisimulation equivalence, denoted by $\leftrightarroweq$. $p \leftrightarroweq q$ means roughly that every transition of p can be mimicked by q such that the resulting pair is strongly bisimilar again and vice versa. Bisimulation equivalence can be found as well in [Par81], [Mil80], [Mil89], [BK84b] and [BW90].

First we need the definition of a bisimulation. In the sequel $p\mathcal{R}q$ abbreviates that the binary relation $\mathcal{R}$ contains the pair (p, q).

Definition 1.2.2 (Bisimulation)
$\mathcal{R} \subseteq T(\text{BPA}\delta) \times T(\text{BPA}\delta)$ *is a* bisimulation *if whenever* $p\mathcal{R}q$ *then*

1. $p \xrightarrow{a} p'$ *implies* $\exists q'$ *such that* $q \xrightarrow{a} q'$ *and* $p'\mathcal{R}q'$.

2. $p \xrightarrow{a} \surd$ *implies* $q \xrightarrow{a} \surd$.

3. *Respectively (1) and (2) with the role of* p *and* q *interchanged.*

Bisimulation equivalence is now defined by

Definition 1.2.3 (Bisimulation Equivalence)
$p \leftrightarroweq q$ *iff there is a bisimulation* $\mathcal{R}$ *relating* p *and* q.

In the context of process algebra we are interested only in equivalences on process terms which are congruences over the process algebra as well, which means intuitively that we can substitute terms for bisimilar ones.

$\leftrightarroweq$ is a congruence over BPAδ if it is an equivalence relation (i.e. if it is symmetric, reflexive and transitive) and it respects the operators of BPAδ. The latter means that $p_1 \leftrightarroweq p_2$ and $q_1 \leftrightarroweq q_2$ implies $p_1 + q_1 \leftrightarroweq p_2 + q_2$ and $p_1 \cdot q_1 \leftrightarroweq p_2 \cdot q_2$.

Theorem 1.2.4 ($\leftrightarroweq$ is a congruence over BPAδ**)**

Proof. A straightforward proof technique is to show that a bisimulation is symmetric, reflexive and transitive. Next, two bisimulations $\mathcal{R}_1$ and $\mathcal{R}_2$ are given such that $p_1\mathcal{R}_1q_1$ and $p_2\mathcal{R}_2q_2$ and it is shown that from $\mathcal{R}_1$ and R_2 bisimulations $\mathcal{R}'$ and $\mathcal{R}''$ can be constructed such that $(p_1 + q_1)\mathcal{R}'(p_2 + q_2)$ and $(p_1 \cdot q_1)\mathcal{R}''(p_2 \cdot q_2)$. In general, this technique can be quite involved, though it is easy for the specific case of BPAδ.

A much easier way to obtain that $\leftrightarroweq$ is a congruency, is to use a result of Groote and Vaandrager ([GV92]). They have proven that if the action rules fit into a certain format, called the *tyft/tyxt* format, then bisimulation equivalence is a congruence. However, this format does not allow predicates on process terms while the rule *atom* actually defines a predicate $R_a^{\surd}$. But, as suggested by Groote and Vaandrager the *tyft/tyxt* format can be generalized to a format that allows predicates. Baeten and Verhoef have elaborated this generalization in [BV93] and they have called this extension the *path* format.

It is very easy to check that the action rules of Table 1.1 fit into the *path* format. □

1.2.4 The Axiom System BPAδ

As the process terms become more complicated it can be quite involved to show an equivalence by constructing a bisimulation. To simplify this reasoning we introduce a mechanism to reason algebraically instead of operationally. For a motivation on axiomatic reasoning on processes we refer to [Mil80], [Mil89], [BK84b] and [BW90].

We give an axiom system that, together with the rules of equational logic, defines the same equivalence. The rules of equational logic, see Table 1.2, correspond to the fact that bisimulation equivalence is a congruence. p and q denote process terms and $C(p)$ denotes a context in which p may occur.

In this thesis we restrict ourselves to process terms only. In the literature, viz. [BW90], the axioms consider arbitrary process variables, ranged over by X,Y and Z. We restrict ourselves to process terms as we introduce so called time variables, that may occur free or bound in a process term, in the following chapters. In the axioms we have to refer to these time variables, which is not possible in case of process variables X, Y. The axiom system BPAδ is given in Table 1.3. Note that by the axioms A1-A3 we can consider a summation as a set of alternatives. Moreover, A6

$p = p$			reflexivity
$p = q$	$\Longrightarrow$	$q = p$	symmetry
$p = q,\ q = z$	$\Longrightarrow$	$p = z$	transitivity
$p = q$	$\Longrightarrow$	$C(p) = C(q)$	context rule
$C(p) = C'(p)$	$\Longrightarrow$	$C(q) = C'(q)$	substitution rule

Table 1.2: Rules for Equational Logic

tells us that the constant δ can be considered as an empty summation. By means of BPAδ we can prove that $(a + b) + a$ equals $b + a$.

Example 1.2.5

$$\text{BPA}\delta \vdash (a+b)+a \overset{\text{A1}}{=} (b+a)+a \overset{\text{A2}}{=} b+(a+a) \overset{\text{A3}}{=} b+a$$

Note that we have used the rules of Table 1.2 implicitly.

We have to prove that this axiom system indeed defines bisimulation equivalence, we will do this in two steps. First we prove that it is *sound*, i.e. if we can derive that two process terms p, q are equal using the axioms A1-A7 then it must be the case that $p \leftrightarrows q$ as well.

A1	$p + q$	$=$	$q + p$
A2	$(p+q)+z$	$=$	$p+(q+z)$
A3	$p+p$	$=$	p
A4	$(p+q)\cdot z$	$=$	$p\cdot z + q\cdot z$
A5	$(p\cdot q)\cdot z$	$=$	$p\cdot(q\cdot z)$
A6	$p+\delta$	$=$	p
A7	$\delta\cdot p$	$=$	δ

Table 1.3: BPA = A1-A5, BPAδ = BPA + A6 + A7

Theorem 1.2.6 (Soundness of BPAδ**)** $p, q \in T(\text{BPA}\delta)$

$$\text{BPA}\delta \vdash p = q \quad \Longrightarrow \quad p \leftrightarrows q$$

Proof. Since we have proven already that $\leftrightarrows$ is a congruence we know that it is sound to apply the rules of equational reasoning. It is left to prove that if $p = q$ is an instance of one of the axioms of BPAδ then $p \leftrightarrows q$ as well. We will discuss only the axioms A1, A4 and A6 by giving a relation and showing that it is a bisimulation. The other axioms are left to the reader.

A1 $p + q \leftrightarrows q + p$
Take

$$\mathcal{R} = \{(u,u) | u \in T(\text{BPA}\delta)\} \cup \{(p+q, q+p)\}$$

It is obvious that $\mathcal{R}$ is a bisimulation with respect to any pair (u, u), so we have to discuss the pair $(p + q, q + p)$ only.

- Consider the case where $p + q \xrightarrow{a} \surd$, this transition can either be derived by $plus_0^l$ (so $p \xrightarrow{a} \surd$) or by $plus_0^r$ (so $q \xrightarrow{a} \surd$).
  Assume $p \xrightarrow{a} \surd$ then we can apply $plus_0^r$ on $q + p$ and we derive $q + p \xrightarrow{a} \surd$.
  The case $q \xrightarrow{a} \surd$ is symmetric, $plus_0^r$ must be replaced by $plus_0^l$ and vice versa.
- The case $p + q \xrightarrow{a} z$ is analogous to the case $p + q \xrightarrow{a} \surd$; $plus_0^l$ must be replaced by $plus_1^l$ and $plus_0^r$ must be replaced by $plus_1^r$.
  So, from $p + q \xrightarrow{a} z$ we can deduce that $q + p \xrightarrow{a} z$ as well and we are ready since $\mathcal{R}(z, z)$.

A4 $(p + q) \cdot z \leftrightarrows p \cdot z + q \cdot z$
Take

$$\mathcal{R} = \{(u,u) | u \in T(\text{BPA}\delta)\} \cup \{((p+q) \cdot z, p \cdot z + q \cdot z)\}$$

The process term $(p + q) \cdot z$ does not have terminating transitions. Every transition $(p + q) \cdot z \xrightarrow{a} z'$ is either deduced by seq_0 or by seq_1.

- If it can be deduced from seq_0 then z' is syntactically equal to z and either $p \xrightarrow{a} \surd$ or $q \xrightarrow{a} \surd$.
  Assume $p \xrightarrow{a} \surd$ then we can apply rule seq_0 to $p \cdot z$ and we derive $p \cdot z \xrightarrow{a} z$ and we are ready since $\mathcal{R}(z, z)$.
  The case where $q \xrightarrow{a} \surd$ is equivalent.
- If it can be deduced from seq_1 then there is a u such that $p + q \xrightarrow{a} u$ and z' is syntactically equivalent to $u \cdot z$. The transition $p + q \xrightarrow{a} u$ can either be derived from $plus_1^l$ or $plus_1^r$.
  Assume $p + q \xrightarrow{a} u$ can be derived from $plus_1^l$, then $p \xrightarrow{a} u$ and then we can apply seq_1 on $p \cdot z$ and we derive $p \cdot z \xrightarrow{a} u \cdot z$ and we are ready since $\mathcal{R}(z', u \cdot z)$.
  The case where $p + q \xrightarrow{a} u$ can be derived from $plus_1^r$ is equivalent to the previous one.

A6 $p + \delta \mathrel{\underline{\leftrightarrow}} p$
Take

$$\mathcal{R} = \{(u,u) | u \in T(\mathrm{BPA}\delta)\} \cup \{(p+\delta, p)\}$$

δ has no transitions at all. So a transition $p + \delta \xrightarrow{a} \surd$ can not be derived from $plus_0^r$ so the transition has to be derived from $plus_0^l$ from which $p \xrightarrow{a} \surd$ follows.

Similarly we can deduce from $p + \delta \xrightarrow{a} z$ that $p \xrightarrow{a} z$ and we are ready since $\mathcal{R}(u,u)$.

□

The other direction of Theorem 1.2.6 is called *completeness*, its proof is postponed till Section 1.2.6.

1.2.5 Notations for Equivalences and Summand Inclusions

Until now, we have seen already several notions of equivalences between process terms, for each of them we will introduce a notation.

To avoid confusion with the notion of provable equality, we write $\equiv$ for equality over process terms. If $p \equiv q$ then we say that p and q are *syntactically* equivalent.

If Θ is an axiom system, such as BPAδ, and there is a derivation within Θ which identifies the process terms p, q then we denote this by $\Theta \vdash p = q$. Next, we define $\subseteq$, denoting derivable summand inclusion; $p \subseteq_\Theta q$ whenever $\Theta \vdash p + q = q$. We write $p \subseteq q$ instead of $p \subseteq_\Theta q$ if Θ is clear from the context.

Often we are not much interested whether two process terms are syntactically equivalent, but more whether they are equal modulo the axioms A1, A2 and A6. Hence, we write $p \simeq q$ for A1,A2,A6 $\vdash p = q$ and $p \sqsubseteq q$ for $p \subseteq_{\mathrm{A1,A2,A6}} q$. In case of $p \simeq q$ we allow ourselves to say that p and q have the same form, or p is of the form q. If we consider the form of a process term p then we can also say that we consider p as a bag of its alternatives.

The axioms A1 and A2 allow us to remove the brackets in $(p_1 + p_2) + p_3$. A process term of the form $p_1 + ... + p_n$ is abbreviated by $\sum_{i \in \{1,\dots,n\}} p_i$. Furthermore, we use the convention that $\sum_{i \in \emptyset} p_i$ denotes δ.

Intuitively one can consider $\sqsubseteq$ as being bag inclusion and $\subseteq_{\mathrm{A1\text{-}3,A6}}$ as set inclusion.

Example 1.2.7

$$\begin{array}{llcl}
 & (a+b)+a & \not\simeq & a+b \\
\mathrm{BPA}\delta \vdash & (a+b)+a & = & a+b \\
 & ((a+b)+c)+a & \not\equiv & (a+a)+(b+c) \\
 & ((a+b)+c)+a & \simeq & (a+a)+(b+c) \\
 & a+b & \sqsubseteq & (a+a)+b \\
 & a+a & \not\sqsubseteq & a+b \\
 & a \cdot b & \subseteq_{\mathrm{BPA}\delta} & (a+c) \cdot b
\end{array}$$

We close this section with a proposition.

Proposition 1.2.8 $a \in A$

$$p \xrightarrow{a} \surd \iff a \sqsubseteq p$$

Proof.

- $\Longrightarrow$. By induction on the length of the derivation of $p \xrightarrow{a} \surd$.
- $\Longleftarrow$. By induction of the number of summands of p.

1.2.6 Basic Terms and Completeness

In this section we define a set $\mathcal{B}$ of *basic* terms that enables us to prove the Completeness Theorem for BPAδ easily. A basic term will be a process term such that the transition system corresponds closely to the structure of the process term. This will be formalized in Proposition 1.2.12. The definition is very simple, only prefixed multiplication is allowed, i.e. process terms of the form $p \cdot q$ where $p \in A_\delta$.

First we define *head normal forms* ([BG87],[BW90]) and *prefix normal forms.*

Definition 1.2.9 (Head Normal Forms) *p is a* head normal form *if it is of the form*

$$\sum_{i \in I} a_i \cdot p_i + \sum_{j \in J} b_j$$

where $a_i, b_j \in A_\delta$ and I and J are finite index sets.

Note, that the p_i's do not have to be head normal forms as well.

A prefix normal form is a head normal form of which its subterms are in prefix normal form as well.

Definition 1.2.10 (Prefix Normal Forms) *p is a* prefix normal form *if it is a head normal form*

$$\sum_{i \in I} a_i \cdot p_i + \sum_{j \in J} b_j$$

such that for every i p_i is a prefix normal form as well.

We have the following proposition which says that every process term can be reduced to a prefix normal form.

Proposition 1.2.11 *Let $p \in T(\text{BPA}\delta)$ then there is a head normal form p' such that* BPA$\delta \vdash p = p'$

Proof. First we show that for any prefix normal forms z, z' there is a prefix normal form u such that BPA$\delta \vdash z \cdot z' = u$. We prove this by induction on z. The base case is $z \equiv a$ and we are ready since $a \cdot z'$ is a prefix normal form. For the other cases we give the following equations which must be read from left to right, such that on the right hand side induction must be applied.

$$\begin{array}{lcl}(z_0 + z_1) \cdot z' & = & z_0 \cdot z' + z_1 \cdot z' \\ (a \cdot z) \cdot z' & = & a \cdot (z \cdot z')\end{array}$$

We can prove the general case by induction of the number of general multiplications, that is the number of subterms of the form $z \cdot z'$ where z is not an atomic action and z' is not in prefix normal form. For the base case we have that z and z' are prefix normal forms, but z is not an atomic action. This case can has already been discussed in the first part of the proof. □

Proposition 1.2.12 *For every prefix normal form we have:*

$$\begin{array}{lcl}p \xrightarrow{a} p' & \Longrightarrow & a \cdot p' \sqsubseteq p \\ p \xrightarrow{a} \surd & \Longrightarrow & a \sqsubseteq p\end{array}$$

Proof. By induction on the structure of p. □

In the untimed setting a basic term is a prefix normal form, without subterms of the form $\delta \cdot p$. By proposition 1.2.11 and the axiom A6 ($\delta \cdot p = \delta$), it is easy to see that every term p can be reduced to a basic term p_b, such that $\mathrm{BPA}\delta \vdash p = p_b$. We denote the set of basic term by $\mathcal{B}$.

We are now ready to prove the Completeness Theorem for BPAδ.

Theorem 1.2.13 (Completeness of BPAδ**)** $p, q \in T(\mathrm{BPA}\delta)$

$$p \leftrightarroweq q \quad \Longrightarrow \quad \mathrm{BPA}\delta \vdash p = q$$

Proof. An implication of Proposition 1.2.11 is that it is sufficient to prove completeness for basic terms only.

We will explain this implication once in detail. We construct basic terms p_b and q_b, such that $\mathrm{BPA}\delta \vdash p = p_b$ and $\mathrm{BPA}\delta \vdash q = q_b$. By soundness of BPA$\delta$ w.r.t. $\leftrightarroweq$ we have $p \leftrightarroweq p_b$ and $q \leftrightarroweq q_b$. By transitivity of $\leftrightarroweq$ and the assumption $p \leftrightarroweq q$ we obtain $p_b \leftrightarroweq q_b$ and it is left to prove that $\mathrm{BPA}\delta \vdash p_b = q_b$, as this implies $\mathrm{BPA}\delta \vdash p = q$.

- Consider an arbitrary summand $a \cdot p'$ of p_b. Then $p_b \xrightarrow{a} p'$ and since $p_b \leftrightarroweq q_b$ there is a q' such that $q_b \xrightarrow{a} q'$ and $p' \leftrightarroweq q'$. By induction $\mathrm{BPA}\delta \vdash p' = q'$ and since $a \cdot q' \sqsubseteq q_b$ we may conclude $a \cdot p' \subseteq_{\mathrm{BPA}\delta} q_b$.

- Consider an arbitrary summand $a \in A$ of p_b. Then $p_b \xrightarrow{a} \surd$ and since $p_b \leftrightarroweq q_b$ also $q_b \xrightarrow{a} \surd$. Hence, $a \sqsubseteq q$.

Adding these results together we obtain $p_b \subseteq_{\mathrm{BPA}\delta} q_b$. Since bisimulation is symmetric we conclude $q_b \subseteq_{\mathrm{BPA}\delta} p_b$. Finally, from $p_b \subseteq_{\mathrm{BPA}\delta} q_b$ and $q_b \subseteq_{\mathrm{BPA}\delta} p_b$ we conclude $\mathrm{BPA}\delta \vdash p_b = q_b$. □

1.3 Parallelism and Communication

If p and q denote processes then we denote their parallel composition by $p\|q$. ACP ([BK84b],[BW90]) has an interleaving view on parallelism. That is, if two processes p and q run in parallel then either the first action comes from p, or from q, or the first action is a result of a communication of an action from p and an action from q. Also CCS ([Mil80][Mil89]) has an interleaved point of view. CCS has for every action a a so-called complementary action $\overline{a}$ (where $\overline{\overline{a}} = a$) such that the communication of a and $\overline{a}$ results into τ. Moreover, in CCS a cannot communicate with actions $b \neq \overline{a}$.

In order to axiomatize the parallel merge we introduce two auxiliary operators, the *communication merge* [BK82] and the *left merge* [BK84b]. For a discussion for the need of the left merge for a finite axiomatization of the parallel merge we refer to [Mol89] as well.

The communication merge is denoted by $|$, $p|q$ is like $p\|q$ with the restriction that only communication actions are allowed in the first step. If there is no initial communication possible between the atomic actions a and b, then $a|b$ equals δ.

The left merge is denoted by $\mathbin{\|\!\_}$. $p \mathbin{\|\!\_} q$ is like $p\|q$, with the restriction that the initial action must come from the left component, that is p, and that no communication is possible.

As example of the parallel merge, communication merge and the left merge we give the following identity:

$$\begin{array}{rcl} (a\cdot p)\|(b\cdot q) & = & (a\cdot p)\mathbin{\|\!\_}(b\cdot q) + (b\cdot q)\mathbin{\|\!\_}(a\cdot p) + (a\cdot p)|(b\cdot q) \\ & & a\cdot(p\|(b\cdot q)) + b\cdot(q\|(a\cdot p)) + (a|b)\cdot(p\|q) \end{array}$$

The last new operator is the *encapsulation* operator, denoted by $\partial_H(p)$. For H a subset of A, $\partial_H(p)$ *encapsulates* all actions of p which occur in p. That is, every action of p in H is blocked, i.e. turned into δ. Assume that a occurs in p and b occurs in q, and $a|b = c$ where $a, b \neq c$, then $\delta_{\{a,b\}}(p|q)$ forces p and q to communicate on the actions a and b. The encapsulation operator originates from [BK84a], and it corresponds with the restriction operator in CCS [Mil80],[Mil89].

1.3.1 The Syntax of ACP

We define $T(\text{ACP})$, the set of process terms over ACP, by the following BNF sentence, where $a \in A_\delta$ and $H \subseteq A$.

$$T(\text{ACP}): \quad p ::= a \mid p_1 + p_2 \mid p_1 \cdot p_2 \mid p_1\|p_2 \mid p_1 \mathbin{\|\!\_} p_2 \mid p_1|p_2 \mid \partial_H(p)$$

We assume a (total) binary function γ on A_δ, which will be called the *communication function*. We require that γ is *commutative* and *associative*, that is:

$$\gamma(a,b) = \gamma(b,a) \ \wedge\ \gamma(\gamma(a,b),c) = \gamma(a,\gamma(b,c))$$

Moreover, we require that $\gamma(a,b) = \delta$ whenever $a = \delta$ or $b = \delta$.

This function γ will be a parameter of the theory ACP.

$$\frac{p \xrightarrow{a} p'}{p \| q \xrightarrow{a} p' \| q \quad p \mathbin{\|\!\_} q \xrightarrow{a} p' \| q} \qquad \frac{p \xrightarrow{a} \surd}{p \| q \xrightarrow{a} q \quad p \mathbin{\|\!\_} q \xrightarrow{a} q}$$

$$\frac{q \xrightarrow{a} q'}{p \| q \xrightarrow{a} p \| q'} \qquad \frac{q \xrightarrow{a} \surd}{p \| q \xrightarrow{a} p}$$

$$\frac{p \xrightarrow{a} p' \,,\; q \xrightarrow{b} q' \,,\; \gamma(a,b) = c}{p \| q \xrightarrow{c} p' \| q' \quad p | q \xrightarrow{c} p' \| q'} \qquad \frac{p \xrightarrow{a} \surd \,,\; q \xrightarrow{b} \surd \,,\; \gamma(a,b) = c}{p \| q \xrightarrow{c} \surd \quad p | q \xrightarrow{c} \surd}$$

$$\frac{p \xrightarrow{a} \surd \,,\; q \xrightarrow{b} q' \,,\; \gamma(a,b) = c}{p \| q \xrightarrow{c} q' \quad p | q \xrightarrow{c} q'} \qquad \frac{p \xrightarrow{a} p' \,,\; q \xrightarrow{b} \surd \,,\; \gamma(a,b) = c}{p \| q \xrightarrow{c} p' \quad p | q \xrightarrow{c} p'}$$

$$\frac{p \xrightarrow{a} p' \,,\; a \notin H}{\partial_H(p) \xrightarrow{a} \partial_H(p')} \qquad \frac{p \xrightarrow{a} \surd \,,\; a \notin H}{\partial_H(p) \xrightarrow{a} \surd}$$

$$a, b, c \in A$$

Table 1.4: Action Rules for ACP Operators

1.3.2 The Action Rules for the ACP Operators

In Table 1.4 we give the action rules for the additional ACP operators. A rule like

$$\frac{p \xrightarrow{a} p'}{p \| q \xrightarrow{a} p' \| q \quad p \mathbin{\|\!\_} q \xrightarrow{a} p' \| q}$$

abbreviates the following two rules

$$\frac{p \xrightarrow{a} p'}{p \| q \xrightarrow{a} p' \| q} \quad \text{and} \quad \frac{p \xrightarrow{a} p'}{p \mathbin{\|\!\_} q \xrightarrow{a} p' \| q}$$

1.3.3 ACPis a conservative extension of BPAδ

We require that the extension of BPA$\delta$to ACPdoes not introduce new identities over BPAδ.

That is, if $\underline{\leftrightarrow}^o$ denotes the old bisimulation equivalence and $\underline{\leftrightarrow}^n$ the new bisimulation equivalence, obtained by the extension, then we require for process terms in the "old" signature ($p, q \in T(\text{BPA}\delta)$) that $p \underline{\leftrightarrow}^n q$ iff $p \underline{\leftrightarrow}^o q$. Note that this is certainly not the case if the extension contains an action rule like

$$\frac{p \xrightarrow{a} p'}{p \xrightarrow{b} p'}$$

But, as the action rules in Table 1.4 only add transitions to terms in $T(\mathrm{ACP}) - T(\mathrm{BPA}\delta)$ it is guaranteed that no new transitions are introduced for terms in $T(\mathrm{BPA}\delta)$.

We refer to a paper of Verhoef [Ver93b], in which a format is studied, in which the action rules of the additional operators have to fit to obtain a conservative extension.

For extensions that are discussed in the following chapters we do not mention anymore the they are indeed conservative.

1.3.4 Axioms for Concurrency

In Table 1.5 we give the axioms for the ACP operators. The axiom CF1 states simply that the communication between two atomic actions is defined by the communication function. Together with the axioms for the left merge and the communication merge, the axiom CM1 states clearly the interleaving character of the parallel merge; either the first action comes from the left or the right component, or it originates from a communication. The other axioms are rather straightforward axioms for the $⫗$, $|$ and ∂_H operators. Again, we have a result that every process term can be reduced to a prefix normal form. The Theorem is called the *Elimination* Theorem since it states that the additional operators of ACP over BPAδ can be eliminated.

Theorem 1.3.1 (Elimination Theorem for ACP)
$\forall p \in T(\mathrm{ACP})\ \exists p'$ *where* p' *is in prefix normal form and* $\mathrm{ACP} \vdash p = p'$

Proof. First, suppose that p is of the from $z\Box z'$, where z and z' are prefix normal forms and $\Box \in \{⫗, |, \|\}$. Then we can prove by induction on $(\mathit{depth}(z+z'), \Box)$ that there is a prefix normal form z such that $\mathrm{ACP} \vdash u = z\Box z'$. We take $(n, \|) > (n, ⫗) = (n, |)$, and $(n, \Box) > (n', \Box')$ when $n > n'$.

The following equations must be read from left to right.

$$\begin{array}{lcll}
a ⫗ z' & = & a \cdot z' & \text{ready}\\
(z_0 + z_1) ⫗ z' & = & z_0 ⫗ z' + z_1 ⫗ z' & \text{use induction}\\
(a \cdot z_0) ⫗ z' & = & a \cdot (z_0 \| z') & \text{use induction}
\end{array}$$

$$\begin{array}{lcll}
a|b & = & \gamma(a,b) & \text{ready}\\
(a \cdot z_0)|b & = & \gamma(a,b) \cdot z_0 & \text{ready}\\
a|(b \cdot z_0') & = & \gamma(a,b) \cdot z_0' & \text{ready}
\end{array}$$

$$\begin{array}{lcll}
(a \cdot z_0)|(b \cdot z_0') & = & \gamma(a,b) \cdot (z_0 \| z_0') & \text{use induction}\\
(z_0 + z_1)|z' & = & z_0|z' + z_1|z' & \text{use induction}\\
z|(z_0' + z_1') & = & z|z_0' + z|z_1' & \text{use induction}
\end{array}$$

$$\begin{array}{lcll}
z \| z' & = & z ⫗ z' + z' ⫗ z + z|z' & \text{use induction}
\end{array}$$

CF1		$a\|b$	$=$	$\gamma(a,b)$
CM1		$p\|\|q$	$=$	$p ⫽ q + q ⫽ p + p\|q$
CM2		$a ⫽ p$	$=$	$a \cdot p$
CM3		$(a \cdot p) ⫽ q$	$=$	$a \cdot (p\|\|q)$
CM4		$(p_1 + p_2) ⫽ q$	$=$	$p_1 ⫽ q + p_2 ⫽ q$
CM5		$(a \cdot p)\|b$	$=$	$(a\|b) \cdot p$
CM6		$a\|(b \cdot p)$	$=$	$(a\|b) \cdot p$
CM7		$(a \cdot p)\|(b \cdot q)$	$=$	$(a\|b) \cdot (p\|\|q)$
CM8		$(p_1 + p_2)\|q$	$=$	$p_1\|q + p_2\|q$
CM9		$p\|(q_1 + q_2)$	$=$	$p\|q_1 + p\|q_2$
D1	$a \notin H$	$\partial_H(a)$	$=$	a
D2	$a \in H$	$\partial_H(a)$	$=$	δ
D3		$\partial_H(p+q)$	$=$	$\partial_H(p) + \partial_H(q)$
D4		$\partial_H(p \cdot q)$	$=$	$\partial_H(p) \cdot \partial_H(q)$

$a, b \in A_\delta,\ H \subseteq A$

Table 1.5: ACP= BPAδ+ CF1+CM1-CM9+D1-D4

Similarly, we can prove for a prefix normal form z that there is a prefix normal form u such that ACP $\vdash u = \partial_H(z)$.

We can prove the general case by induction on the number of occurrences of ACP operators, i.e., $\|\|, ⫽, \|, \partial_H$, using Proposition 1.2.11 and the first part of this proof.

□

1.4 Abstraction

1.4.1 A new Constant for the Silent Step

In practice, if we have an implementation and a specification of a process then we want to be able to abstract from all the internal details of the implementation such that it can be proven equivalent with the specification. Therefore we introduce a constant, τ, called the *silent step*, that denotes internal activity. The silent step is due to Milner ([Mil83],[Mil89]).

For example, consider the process $a \cdot i \cdot b$ where i is supposed to be an internal action, then we want to prove this process somehow to be equal to $a \cdot b$, since these processes equal with respect to their external actions. First we have to express formally that i is an internal action, this is done by applying the τ_I operator which renames every action in I into the silent action τ, i.e. $\tau_{\{i\}}(a \cdot i \cdot b) = a \cdot \tau \cdot b$. Next, we use the features of the silent step τ by which we can show $a \cdot \tau \cdot b$ to be equal with $a \cdot b$. The operator τ_I can be found in [BK85]. More general examples of renaming operators can be found in CCS ([Mil80],[Mil89]) and CSP ([Hoa85]) as well.

The problem is to define an equivalence on transition systems (i.e. some bisimulation) which takes the special character of the silent step into regard. In the literature (among others) three different, but comparable, equivalences have been introduced ([Mil80],[Mil83], [Mil89] and [GW91]).

The strictest one, *branching bisimulation* equivalence ([GW91]), allows to reduce τ in $\tau \cdot p + q$ to p, if we do not disregard any options of q. This means that all options of q must be offered by p as well, in other words, q must be a summand of p.

Branching bisimulation equivalence itself is not a congruence. Therefore, an extra condition, called *rootedness*, is imposed on the branching bisimulation relations.

We extend the alphabets A and A_δ by the constant τ and obtain A_τ resp. $A_{\delta\tau}$. The set of process terms over BPA, where the constants are taken from $A_{\delta\tau}$, is denoted by $T(\text{BPA}\delta\tau)$.

1.4.2 Semantics for the Silent Step

The three different bisimulation equivalences and their rooted versions which regard the silent step are: branching bisimulation ([GW91]), delay bisimulation [Mil83] and weak bisimulation ([Mil80],[Mil89]).

$$\begin{array}{ccccc}
\underline{\leftrightarrow}_b & \subset & \underline{\leftrightarrow}_d & \subset & \underline{\leftrightarrow}_w \\
\cup & & \cup & & \cup \\
\underline{\leftrightarrow}_{rb} & \subset & \underline{\leftrightarrow}_{rd} & \subset & \underline{\leftrightarrow}_{rw}
\end{array}$$

Each of these bisimulation equivalences allows that an a-transition on one side may be mimicked by a a-transition possibly preceded or followed by silent steps on the other side. This is shown in Figure 1.1; the formal definitions are given below.

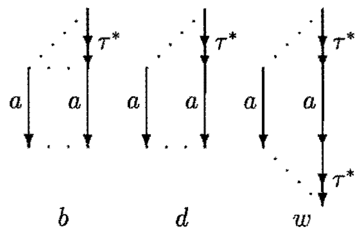


Figure 1.1: Three bisimulations with τ

We have one predicate on $T(\text{BPA}\delta\tau) \cup \{\surd\}$ which is denoted by $\surd$. $\surd(p)$ holds iff all maximal paths starting in p consist of τ's only and end in $\surd$. Note that

$\surd(\surd)$. This predicate is very similar to the weak termination predicate of Aceto and Hennessy [AH92].

In the rest of this section (Section 1.4) we let $\hat{p}$, $\hat{q}$ and $\hat{z}$ range over $T(\mathrm{BPA}\delta\tau) \cup \{\surd\}$. In the following $p \Longrightarrow \hat{p}$ denotes that there is a path $p \xrightarrow{\tau} \ldots \xrightarrow{\tau} \hat{p}$ of length zero or more.

Definition 1.4.1 *$\mathcal{R} \subseteq T(\mathrm{BPA}\delta\tau) \times T(\mathrm{BPA}\delta\tau)$ is a* branching *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{a} \hat{p}$ and $\neg(\surd(\hat{p}))$ then either $a = \tau$ and $\hat{p}\mathcal{R}q$ or $\exists z, q'$ such that $q \Longrightarrow z \xrightarrow{a} q'$, $p\mathcal{R}z$ and $\hat{p}\mathcal{R}q'$.*
2. *If $p \xrightarrow{a} \hat{p}$ and $\surd(\hat{p})$ then $\exists z, \hat{q}$ such that $q \Longrightarrow z \xrightarrow{a} \hat{q}$ with $\surd(\hat{q})$ and $p\mathcal{R}z$.*
3. *Respectively (1) and (2) with the role of p and q interchanged.*

Definition 1.4.2 *$\mathcal{R} \subseteq T(\mathrm{BPA}\delta\tau) \times T(\mathrm{BPA}\delta\tau)$ is a* delay *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{a} \hat{p}$ and $\neg(\surd(\hat{p}))$ then either $a = \tau$ and $\hat{p}\mathcal{R}q$ or $\exists z, q'$ such that $q \Longrightarrow z \xrightarrow{a} q'$ and $\hat{p}\mathcal{R}q'$.*
2. *If $p \xrightarrow{a} \hat{p}$ and $\surd(\hat{p})$ then $\exists z, \hat{q}$ such that $q \Longrightarrow z \xrightarrow{a} \hat{q}$ with $\surd(\hat{q})$.*
3. *Respectively (1) and (2) with the role of p and q interchanged.*

Definition 1.4.3 *$\mathcal{R} \subseteq T(\mathrm{BPA}\delta\tau) \times T(\mathrm{BPA}\delta\tau)$ is a* weak *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{a} \hat{p}$ and $\neg(\surd(\hat{p}))$ then either $a = \tau$ and $\hat{p}\mathcal{R}q$ or $\exists z, z', q'$ such that $q \Longrightarrow z \xrightarrow{a} z' \Longrightarrow q'$ and $\hat{p}\mathcal{R}q'$.*
2. *If $p \xrightarrow{a} \hat{p}$ and $\surd(\hat{p})$ then $\exists z, z', \hat{q}$ such that $q \Longrightarrow z \xrightarrow{a} z' \Longrightarrow \hat{q}$ and $\surd(\hat{q})$*
3. *Respectively (1) and (2) with the role of p and q interchanged.*

We need the predicate $\surd$ to express that "τ-stuttering" afterwards is allowed, as we require that a and $a \cdot \tau$ are branching bisimilar. For $* \in \{b, d, w\}$ we define $*$-bisimulation equivalence.

Definition 1.4.4 *$p \leftrightarroweq_* q$ iff there is an $*$-bisimulation relating p and q.*

None of these equivalences is a congruence over $T(\mathrm{BPA}\delta\tau)$. We have to restrict these equivalences to obtain congruences by imposing a rootedness condition on the bisimulations.

Definition 1.4.5 *A relation $\mathcal{R}$ is* rooted w.r.t. p and q *if $p\mathcal{R}q$ and if $p'\mathcal{R}q'$ implies that $p' \equiv p \Leftrightarrow q' \equiv q$*

We obtain *rooted* $*$-bisimulation equivalences, denoted by $p \leftrightarroweq_{r*} q$, by requiring that there is a rooted $*$-bisimulation relating p and q. Now we have for each $* \in \{b, d, w\}$ that:

Proposition 1.4.6 *$\leftrightarroweq_{r*}$ is a congruence*

For a proof of this proposition we refer to [BW90] and [GW91].

1.4.3 Laws for abstraction

We have the following axiom systems. The laws T1-T3 are taken from Milner

T1	$p \cdot \tau$	$=$	p	B1
	$z \cdot (\tau \cdot (p+q) + p)$	$=$	$z \cdot (p+q)$	B2
T2	$\tau \cdot p$	$=$	$\tau \cdot p + p$	
T3	$a \cdot (\tau \cdot p + q)$	$=$	$a \cdot (\tau \cdot p + q) + a \cdot p$	

Table 1.6: The τ laws

([Mil80],[Mil89]). B2 is Van Glabbeek & Weijland's branching bisimulation law ([GW91]), note that A1-A3+A5+T1+T2 $\vdash$ B2. Each rooted bisimulation equivalence can be axiomatized completely by its corresponding theory.

Theorem 1.4.7 $p, q \in T(\mathrm{BPA}\delta\tau)$

$$\begin{array}{lcl} p \leftrightarroweq_{rb} q & \Longleftrightarrow & \mathrm{BPA}\delta + \mathrm{B1} + \mathrm{B2} \vdash p = q \\ p \leftrightarroweq_{rd} q & \Longleftrightarrow & \mathrm{BPA}\delta + \mathrm{T1} + \mathrm{T2} \vdash p = q \\ p \leftrightarroweq_{rw} q & \Longleftrightarrow & \mathrm{BPA}\delta + \mathrm{T1} + \mathrm{T2} + \mathrm{T3} \vdash p = q \end{array}$$

In [GW91] the completeness is proven for branching bisimulation equivalence first. From this result the other completeness results can be found easily.

The combination of τ and ACP is not trivial. For ACP with τ and branching bisimulation the extension is completely straightforward and no extra axioms are needed. However, for ACP with τ and delay bisimulation one needs additional axioms. A typical example is that

$$a|(\tau \cdot b) = a|(\tau \cdot b + b) = a|(\tau \cdot b) + a|b$$

So, if one assumes that τ cannot communicate, then one obtains $a|(\tau \cdot b) = a|b$, and thus an extra axiom $p|(\tau \cdot q) = p|q$ is needed. Further details of this aspect of abstraction do not fall within the scope of this introductary chapter, and we refer the reader to [BW90] and [Gla87].

1.4.4 Strongly Rootedness

In the timed case we will come across a stronger rootedness condition. There it is required that a rooted bisimulation acts on the pair of root nodes as a strong bisimulation.

Definition 1.4.8 *A relation $\mathcal{R}$ is* strongly rooted w.r.t. p and q *if*

1. $p\mathcal{R}q$

2. *$p \xrightarrow{a} \hat{p}$ implies $\exists \hat{q}$ with $q \xrightarrow{a} \hat{q}$ such that either $\surd(\hat{p})$ and $\surd(\hat{q})$ or, $\neg(\surd(\hat{p}))$, $\neg(\surd(\hat{q}))$ and $p'\mathcal{R}q'$.*

3. *(2) with the role of p and q interchanged.*

In this way we obtain *strongly rooted branching bisimulation* ($\leftrightarrow_{srb}$), *strongly rooted delay bisimulation* ($\leftrightarrow_{srd}$) and *strongly rooted weak bisimulation* ($\leftrightarrow_{srw}$).

For branching bisimulation strongly rootedness is not strictly stronger than rootedness:

Proposition 1.4.9 $p, q \in T(\mathrm{BPA}\delta\tau)$

$$p \leftrightarrow_{rb} q \quad \Longleftrightarrow \quad p \leftrightarrow_{srb} q$$

Proof. We prove only $\Longrightarrow$, the other direction is trivial. Assume $\mathcal{R}$ is a rooted branching bisimulation w.r.t. p and q. We prove that $\mathcal{R}$ is a strongly rooted branching bisimulation w.r.t. p and q as well.

Consider the case where $p \xrightarrow{a} \hat{p}$ and $\neg(\surd(\hat{p}))$, then we have to show that there is a $\hat{q}$ such that, $\neg(\surd(\hat{q}))i$, $q \xrightarrow{a} \hat{q}$ and $\hat{p} \leftrightarrow_b \hat{q}$.

It cannot be the case that $a = \tau$ and $\hat{p}\mathcal{R}q$ since $\mathcal{R}$ is rooted w.r.t. (p, q), as $\hat{p}\mathcal{R}q$ would imply that $\hat{p} \equiv p$. Hence there is a z and a $\hat{q}$ such that $q \Longrightarrow z \xrightarrow{a} \hat{q}$, $p\mathcal{R}z$ and $\hat{p}\mathcal{R}\hat{q}$. Since $\mathcal{R}$ is rooted w.r.t. to (p, q) it follows from $\hat{p}\mathcal{R}z$ that $z \equiv q$, thus $q \xrightarrow{a} \hat{q}$. Morover $\hat{p}\mathcal{R}\hat{q}$ implies $\hat{p} \leftrightarrow_b \hat{q}$. It is left to the reader to prove that $\neg(\surd(\hat{p}))$ and $\hat{p} \leftrightarrow_b \hat{q}$ imply that $\neg(\surd(\hat{q}))$, and we are ready with this case.

The case where $p \xrightarrow{a} \hat{p}$ and $\surd(\hat{p})$ is left to the reader. □

This is certainly not the case for delay bisimulation, as is shown by the following example:

Example 1.4.10 $p \leftrightarrow_{srd} q \quad \not\Longleftrightarrow \quad p \leftrightarrow_{rd} q$

$$\tau \cdot a \leftrightarrow_{rd} \tau \cdot a + a \quad \text{but} \quad \tau \cdot a \not\leftrightarrow_{srd} \tau \cdot a + a$$

1.5 Recursion

1.5.1 Introduction

Until now we have considered finite processes only. In order to express infinite processes we introduce the standard concept of *recursion.* For example, the process which executes the infinite sequence *abab...* can be expressed by

$$X \overset{def}{=} a \cdot b \cdot X$$

Here, X is a so-called *recursion variable* that is bound by the declaration $X \overset{def}{=} a \cdot b \cdot X$. A (recursive) specification E consists of a number of declarations of the form $X_i \overset{def}{=} p_i$. Here, p_i is a process term, which is called the *body* of X_i, in which recursion variables may occur.

For a recursion variable X, with declaration $X = p_X$, and process term p such that p is (rooted bisimilar) bisimilar with $p_X[p/X]$, we say that p is a solution of X modulo (rooted bisimilar) bisimulation equivalence. For example, if we have

$$X \overset{def}{=} (a + b) \cdot X$$
$$Y \overset{def}{=} (a \cdot Y \| b \cdot Y)$$

where $\gamma(a, b) = \delta$, then X is solution for Y modulo bisimulation equivalence, and vice versa.

If we consider

$$Z \overset{def}{=} Z$$

then every process term is a solution for Z.

In algebraic reasoning we often need the principle that certain specifications have unique solutions. We introduce the notion of *guarded* declarations and specifications, for example, the above declaration for Z is not guarded. We show that if p and q are both solutions for the same recursion variable and the same guarded specification, then p and q are equal as well. To show this, we first introduce the projection operator π_n, which restricts a process term p to its first n steps. Then, it is shown that two process terms are equal if they are equal for all their finite projections.

1.5.2 Some Definitions

We assume a set *RVar* of *recursion variables*, with typical element X. If $\mathcal{R}$ is a finite subset of *RVar* then we denote by $T(\mathcal{R}, \mathrm{BPA}\delta_T)$ the set of process terms over $\mathrm{BPA}\delta_T$ in which the recursion variables from $\mathcal{R}$ may occur as atomic constructs. If $p \in T(\mathrm{BPA}\delta_T)(= T(\emptyset, \mathrm{BPA}\delta_T))$ then we call p a finite process term.

A *specification* E is a finite collection of declarations of the form

$$\{X_0 \overset{def}{=} p_0, ..., X_n \overset{def}{=} p_n\}$$

where $p_i \in T(\{X_0, ..., X_n\}, \text{BPA}\delta\tau)$ and $i \neq j$ implies $X_i \not\equiv X_j$. We denote the set $\{X_0, ..., X_n\}$ by $rvar(E)$. For $X \in rvar(E)$ we denote the right hand side of the declaration of X in E by p_X^E. If $X \notin rvar(E)$ then p_X^E denotes δ.

We parameterize the action relations of our operational semantics by a specification E. We have two additional action rules which are given in Table 1.7. We obtain equivalences like $\underline{\leftrightarrow}_s^E$, $\underline{\leftrightarrow}_b^E$ and $\underline{\leftrightarrow}_{rb}^E$ in the obvious way.

$$\boxed{\frac{p_X^E \xrightarrow{a}_E p'}{X \xrightarrow{a}_E p'} \quad \frac{p_X^E \xrightarrow{a}_E \surd}{X \xrightarrow{a}_E \surd}}$$

Table 1.7: Action Rules for Recursion

Definition 1.5.1 (the notion of a solution)
$p \in T(rvar(E), \text{BPA}\delta\tau)$ is a solution *for X in E modulo $\underline{\leftrightarrow}$ if $p \underline{\leftrightarrow}^E p_X[p/X]$.*

We have similar definitions for $\underline{\leftrightarrow}_b$ and $\underline{\leftrightarrow}_{rb}$.

In the literature, such as [BW90], X occurs guarded in p if there is no trace $p \Longrightarrow X$. For example X occurs guarded in $a \cdot X + b$ but not in $\tau \cdot X + b$ or $X + b$. For a specification the definition of guardedness is more involved as the declaration of X in

$$X \stackrel{def}{=} \tau \cdot Y + a \cdot X$$
$$Y \stackrel{def}{=} \tau \cdot Y + b \cdot X$$

is unguarded. In this section we define guardedness as a predicate within the theory. If $G^E(p)$ is true, then p is guarded. We have an auxiliary predicate $G_{\mathcal{R}}^E(p)$, where $\mathcal{R}$ is a set of recursion variables, $G^E(p)$ is defined by $G_\emptyset^E(p)$. This suffix $\mathcal{R}$ contains the recursion variables that are encountered during the "investigation" of p. If a recursion variable is encountered which occurs already in $\mathcal{R}$ then *ff* (false) is returned, otherwise X is added to $\mathcal{R}$ and the investigation continues with the body of X. The axioms for $G_{\mathcal{R}}^E$ are given in Table 1.8. As an example of the use of this predicate we give the derivation for $G_\emptyset^E(X) = ff$ where E is the specification which contains the above declarations for X and Y.

Example 1.5.2

$$\begin{aligned} & G_\emptyset^E(X) \\ = \; & G_{\{X\}}^E(\tau \cdot Y + a \cdot X) \\ = \; & G_{\{X\}}^E(\tau \cdot Y) \wedge G_{\{X\}}^E(a \cdot X) \\ = \; & G_{\{X\}}^E(Y) \wedge tt \end{aligned}$$

$$\begin{aligned}
&= G^E_{\{X,Y\}}(\tau \cdot Y + b \cdot X)\\
&= G^E_{\{X,Y\}}(\tau \cdot Y) \;\wedge\; G^E_{\{X,Y\}}(b \cdot X)\\
&= G^E_{\{X,Y\}}(Y) \;\wedge\; tt\\
&= f\!f
\end{aligned}$$

G1	$G^E_{\mathcal{R}}(a)$	$=$	tt	
G2	$G^E_{\mathcal{R}}(\tau)$	$=$	tt	
G3	$G^E_{\mathcal{R}}(a \cdot p)$	$=$	tt	
G4	$G^E_{\mathcal{R}}(\tau \cdot p)$	$=$	$G^E_{\mathcal{R}}(p)$	
G5	$G^E_{\mathcal{R}}(p + q)$	$=$	$G^E_{\mathcal{R}}(p) \;\wedge\; G^E_{\mathcal{R}}(q)$	
G6	$G^E_{\mathcal{R}}(X)$	$=$	$G^E_{\mathcal{R}\cup\{X\}}(p^E_X)$	if $X \in rvar(E) - \mathcal{R}$
G7	$G^E_{\mathcal{R}}(X)$	$=$	$f\!f$	otherwise
G8	$G^E_{\mathcal{R}}(X \cdot p)$	$=$	$G^E_{\mathcal{R}\cup\{X\}}(p^E_X \cdot p)$	if $X \in rvar(E) - \mathcal{R}$
G9	$G^E_{\mathcal{R}}(X \cdot p)$	$=$	$f\!f$	otherwise
B1	$tt \wedge \alpha$	$=$	α	
B2	$f\!f \wedge \alpha$	$=$	$f\!f$	

$\mathcal{R} \subseteq RVar,\ \alpha \in T(\{tt, f\!f\}, \wedge)$

Table 1.8: Axioms for the (boolean) guardedness function

The following proposition states that for any E, $\mathcal{R}$ and p it can be determined whether $G^E_{\mathcal{R}}(p)$ is tt or $f\!f$.

Proposition 1.5.3 *For all E, $\mathcal{R}$ and $p \in T(RVar, \mathrm{BPA}\delta\tau)$ there is a boolean expression α, either tt or $f\!f$, such that* $\mathrm{A4,5} + \mathrm{G1\text{-}9} + \mathrm{B1,2} \vdash G^E_{\mathcal{R}}(p) = \alpha$

Proof. First we define a lexicographic ordering on pairs of natural numbers. That is, $(n, m) > (n', m')$ whenever $n > n'$ or $n = n'$ and $m > m'$. The proof uses induction on $(|rvar(E) - \mathcal{R}|, size(p))$.

We discuss only the case where $p \equiv p_1 \cdot p_2$ and we introduce an internal induction on the size of p_1.

- $p_1 \equiv a$. Immediate by G1.
- $p_1 \equiv \tau$. Immediate by G2.
- $p_1 \equiv z_1 + z_2$. Then

$$\begin{aligned}
G^E_{\mathcal{R}}((z_1 + z_2) \cdot p_2) &\overset{\mathrm{A4}}{=} G^E_{\mathcal{R}}(z_1 \cdot p_2 + z_2 \cdot p_2)\\
&\overset{\mathrm{G5}}{=} G^E_{\mathcal{R}}(z_1 \cdot p_2) \;\wedge\; G^E_{\mathcal{R}}(z_2 \cdot p_2)
\end{aligned}$$

and by induction we are ready.

- $p_1 \equiv z_1 \cdot z_2$.

$$G^E_{\mathcal{R}}((z_1 \cdot z_2) \cdot p_2) \overset{\text{A5}}{=} G^E_{\mathcal{R}}(z_1 \cdot (z_2 \cdot p_2))$$

and by (internal) induction we are ready.

- $p_1 \equiv X$. If $x \notin rvar(E) - \mathcal{R}$ then we are immediately ready by G9. So, assume $x \in rvar(E) - \mathcal{R}$ then we can apply G8 and we obtain $G^E_{\mathcal{R}\cup\{X\}}(p^E_X \cdot p_2)$. Since $|rvar(E) - (\mathcal{R} \cup \{X\})| < |rvar(E) - \mathcal{R}|$ we can apply induction.

□

Definition 1.5.4 (Guardedness) *The specification E is* guarded *if for all $X \in rvar(E)$* A4, 5 + G1-9 + B1, 2 $\vdash G^E(X) = tt$.

And, of course, if a specification E is guarded then all process terms over E are guarded as well.

Proposition 1.5.5 *Let E be a guarded specification and $p \in T(rvar(E), \text{BPA}\delta\tau)$ then* A4, 5 + G1-9 + B1, 2 $\vdash G^E(p) = tt$.

Proof. Omitted. □

1.5.3 Axioms for Recursion and Projection

We need an axiom, REC^E, that "imports" the declarations of the specification E as identities in the axiom system.

In [BW90] the *Recursion Specification Principle* is defined as

A recursive specification has at most one solution.

As we have discussed in the introduction of this section we know that this principle does not hold in general. The *Restricted* Recursion Specification Principle considers only guarded specifications [BK86]:

A guarded recursive specification has at most one solution.

In Table 1.9 we formulate the Restricted Recursion Specification Principle as a conditional axiom RSP^E_G, the G denotes that E is supposed to be a guarded specification; the condition $\wedge_{Y \in rvar(E)} G^E_\emptyset(Y) = tt$ is kept implicit in the premise of the axiom. The conditional axiom RSP^E_G compares two vectors of process terms, $\overline{p} = (p_1, \ldots, p_n)$ and $\overline{q} = (q_1, \ldots, q_n)$. For two such vectors $\overline{p}, \overline{q}$ we abbreviate $p_1 = q_1, \ldots, p_n = q_n$ by $\overline{p} = \overline{q}$. If z is a process term and $\overline{X} = (X_1, \ldots, X_n)$ is a vector of recursion variables, then the simultaneous substitution of p_i for X_i in z is denoted by $z[\overline{p}/\overline{X}]$.

REC^E	$X = p^E_X$
RSP^E_G	$\overline{p} = p_{\overline{X}}[\overline{p}/\overline{X}],\ \overline{q} = p_{\overline{X}}[\overline{q}/\overline{X}] \implies \overline{p} = \overline{q}$

Table 1.9: Additional axioms for recursion

If $\overline{z} = (z_1, \ldots, z_n)$ is a vector of process terms, then the simultaneous substitution $[\overline{p}/\overline{X}]$ on each z_i is denoted by $\overline{z}[\overline{p}/\overline{X}]$. Finally, if $\overline{X} = (X_1, \ldots, X_n)$ is a vector of process variables, then we denote by $p_{\overline{X}}$ the vector of declaration bodies $(p_{X_1}, \ldots, p_{X_n})$.

We introduce the projection operator, though we define it a little different from [BW90]. In [BW90] $\pi_1(a \cdot p) = a$, whereas we have $\pi_1(a \cdot p) = a \cdot \delta$. The reason is that in [BW90] $\pi_0(p)$ is defined equal to ϵ, that is the *empty* process which terminates successfully immediately. Hence, they have derivations like $\pi_1(a \cdot p) = a \cdot \pi_0(p) = a \cdot \epsilon = a$. As we want to define π_o, but not ϵ, we have decided to put $\pi_0(a) = \delta$ for $a \in A$. Since $a = a \cdot \tau$ we put $\pi_0(\tau) = \tau$. The axioms for the projection operator are given in Table 1.10.

PR1	$\pi_0(a)$	$=$	δ
PR2	$\pi_{n+1}(a)$	$=$	a
PR3	$\pi_n(\tau)$	$=$	τ
PR4	$\pi_0(a \cdot p)$	$=$	δ
PR5	$\pi_{n+1}(a \cdot p)$	$=$	$a \cdot \pi_n(p)$
PR6	$\pi_n(\tau \cdot p)$	$=$	$\tau \cdot \pi_n(p)$
PR7	$\pi_n(p+q)$	$=$	$\pi_n(p) + \pi_n(q)$

$a \in A_\delta,\ n \geq 0$

Table 1.10: Axioms for the projection operator

1.5.4 The Soundness of the Restricted Recursion Specification Principle

In this subsection we show the soundness of RSP^E_G for rooted branching bisimulation equivalence. First we show that for any p, p is equal to some head normal form. The proposition and lemmas of this subsection are borrowed from [BW90], though the proofs are different. The proofs below use induction and they are based on the axiomatic definition of guardedness.

Proposition 1.5.6 *Let E be a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\delta\tau)$, then there is a p' such that p' is in head normal form and* $\mathrm{BPA}\delta + \mathrm{REC}^E \vdash p = p'$

Proof. We introduce the function $rvar^E(p)$ that gives for each process term the set of recursion variables which can be reached by passing τ's only. We give some of its axioms, the other axioms are left to the reader.

$$\begin{array}{lcll} rvar^E(\tau \cdot p) & = & rvar^E(p) & \\ rvar^E(X) & = & \{X\} \cup rvar^E(p_X^E) & \\ rvar^E(p_1 \cdot p_2) & = & rvar^E(p_1) \cup rvar^E(p_2) & \text{if } p_1 \Longrightarrow \surd \\ rvar^E(p_1 \cdot p_2) & = & rvar^E(p_1) & \text{otherwise} \end{array}$$

Note that since p is guarded we can define by induction $p \Longrightarrow \surd$ as a predicate.

We have

$$G_\emptyset^E(X) = tt \quad \Longrightarrow \quad X \notin rvar^E(p_X^E)$$

and thus $G^E(X)$ implies that $rvar^E(X) \supset rvar^E(p_X^E)$. We prove the proposition by induction on $(size(p), rvar^E(p))$. We discuss only the case where $p \equiv p_1 \cdot p_2$ and $p \equiv X$.

- $p \equiv p_1 \cdot p_2$. Let $p_1' \simeq \sum_i a_i \cdot p_i + \sum_j b_j$ be the head normal form of p_1, then $p = p'$ if we take $p' \simeq \sum_i a_i \cdot (p_i \cdot p_2) + \sum_j b_j \cdot p_2$ which is in head normal form as well.
- $p \equiv X$. Since $rvar^E(X) \supset rvar^E(p_X^E)$ we may assume that we have already constructed a head normal form p' for p_X^E. By REC^E we obtain $X = p_X^E = p'$, and we are ready.

□

Let us denote by $hnf(p)$ the head normal form of p which is constructed by the proof in the Lemma above. When $G_\emptyset^E(p) = tt$ then we define the function $l(p)$ which corresponds with the length of the derivation of $G_\emptyset^E(p) = tt$. In order to do this we have to fix a derivation. If p is in head normal form then exactly one of the axioms G1-7 is applicable; we put $l(\tau \cdot p) = 1 + l(p)$ and we do similarly for the other cases. If p is not in head normal form then we take $l(p) = 1 + l(hnf(p))$.

Lemma 1.5.7 *If E is a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\delta\tau)$ then for each n there is a finite process term p', without occurrences of the projection operator, such that*

$$\mathrm{BPA}\delta + \mathrm{REC}^E + \mathrm{PR1\text{-}7} \vdash \pi_n(p) = p'$$

Proof. We may assume for each p that $G_\emptyset^E(p) = tt$ and we use induction on $(n, l(p))$.

First we assume that p is a head normal form and $p \simeq \sum_i a_i \cdot p_i + \sum_j b_j$.

$$\begin{aligned}\pi_n(\textstyle\sum_i a_i \cdot p_i + \sum_j b_j) &= \textstyle\sum_i \pi_n(a_i \cdot p_i) + \sum_j \pi_n(b_j)\\ &= \textstyle\sum_i \pi_n(a_i \cdot p_i) + \sum_j b_j\end{aligned}$$

Take an index i, if $a_i \neq \tau$ then we have $\pi_n(a_i \cdot p_i) = a_i \cdot \pi_{n-1}(p_i)$ and by induction there is a finite process term p'_i for $\pi_{n-1}(p_i)$. So, assume $a_i = \tau$, then $\pi_n(\tau \cdot p_i) = \tau \cdot \pi_n(p_i)$ then since $l(\tau \cdot p_i) = 1 + l(p_i)$ we know that we have constructed already a finite process term p'_i for $\pi_n(p_i)$.

If p is not in head normal form then we have $\pi_n(p) = \pi_n(hnf\,(p))$ and we can apply induction, as $l(p) = 1 + l(hnf\,(p))$. □

Next, we have a proposition that says that if p is a solution for X in E modulo $\leftrightarrows$, then for every n the projection of p is bisimilar with the projection of the body of X.

Lemma 1.5.8 *Let E be a guarded specification with $X \in rvar(E)$ such that p is a solution for X modulo $\leftrightarrows_{(rb)}$, then for all n we have $\pi_n(p) \leftrightarrows^E_{(rb)} \pi_n(p_X)$.*

Proof. Since p is a solution for X in E we have $\pi_n(p) \leftrightarrows^E_{(rb)} \pi_n(p_X[p/X])$. Consider the derivation between $\pi_n(p_X)$ and $hnf\,(\pi_n(p_X))$, note that the latter process term is a finite process term, so X does not occur in it. For each step in this derivation for which $\mathrm{REC}^E\ X = p_X$ is used we apply $p = p_X[p/X]$ instead. This latter equality is sound, since p is a solution for X. This gives us a derivation between $\pi_n(p_X[p/X])$ and $hnf\,(\pi_n(p_X))$ that is sound for $\leftrightarrows_{(rb)}$. Since also $hnf\,(\pi_n(p_X)) \leftrightarrows_{(rb)} \pi_n(p_X)$ and we are ready. □

From this proposition we obtain the so-called Projection Lemma, that says that if two process terms are both solutions for X in E modulo (rooted branching) bisimulation equivalence, then for every projection p and q are (rooted branching) bisimilar as well.

Lemma 1.5.9 (Projection Lemma) *If E is a guarded specification with $X \in rvar(E)$ such that both $p, q \in T(rvar(E), \mathrm{BPA}\delta\tau)$ are solutions for X modulo $\leftrightarrows_{(rb)}$ then for all n we have $\pi_n(p) \leftrightarrows_{(rb)} \pi_n(q)$.*

Proof. Immediate from Lemma 1.5.8. □

As in [BW90] we obtain RSP^E_G by proving that two processes are equal if all there finite projections are equal. This principle is known as the *Approximation Induction Principle* (AIP). This principle originates from [BK86]. Restricted versions of AIP can be found in [BBK87] and [Gla87]. In [BW90] the definition and proof of [Gla87] is given. We define AIP for a guarded specification; again the condition that E is guarded, i.e. $\wedge_{Y \in rvar(E)} G^E_\emptyset(Y) = tt$ is kept implicit in the premise of the axiom.

AIP^E_G	$\forall n\ :\ \pi_n(p) = \pi_n(q)$	$\Longrightarrow \quad p = q$

We restrict AIP to the setting with a guarded specification, as we can transfer only the soundness proof of this restricted case to the timed case. The proof of [Gla87] is too subtle to be transferred. The proof below is based on a part, the "easy" one, of the proof of [Gla87]. Before proving the soundness of AIP_G^E we need a proposition which states that in the context of a guarded specification E every p can reach only finitely many p's by a sequence of τ-transitions.

Proposition 1.5.10 *If E is a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\delta\tau)$ then the set $\{p' | p \Longrightarrow p'\}$ is finite.*

Proof. By induction on $l(p)$. □

Now we can prove the soundness of AIP_G^E for rooted branching bisimulation..

Theorem 1.5.11 (Soundness of AIP_G^E) *If E is a guarded specification and $p, q \in T(rvar(E), \mathrm{BPA}\delta\tau)$ then*

$$\forall n \ : \ \pi_n(p) \leftrightarroweq_{rb}^E \pi_n(q) \quad \Longrightarrow \quad p \leftrightarroweq_{rb}^E q$$

Proof. We consider subterms p' of p such that p' can be reached from p in more than zero transitions. Similarly we consider subterms q' of q. We define for each m a relation $\sim_m$ on those subterms p' and q' such that

$$p' \sim_m q' \quad \Longleftrightarrow \quad \pi_m(p') \leftrightarroweq_b \pi_m(q')$$

and we put $p' \sim q'$ if for all m we have $p' \sim_m q'$.

We show first that $\sim$ is a branching bisimulation. Take p', q' such that $p' \sim q'$.

- Consider p'' such that $p' \xrightarrow{a} p''$, where $a \in A$, and put

$$S_n \ = \ \{ \ (z, q^*) \mid q' \Longrightarrow z \xrightarrow{a} q^*, \ p' \sim_{n+1} z, \ p'' \sim_n q^* \}$$

  Then we have

  1. $S_0 \supseteq S_1 \supseteq S_2 \supseteq \ldots$ since $u \sim_{k+1} u'$ implies $u \sim_k u'$.
  2. For all n $S_n \neq \emptyset$ since $p' \sim_{n+1} q'$.
  3. For all n S_n is finite, by Proposition 1.5.10.

  Hence $\bigcap_{n=0}^{\infty} S_n \neq \emptyset$ and we can take a pair $(z, q'') \in \bigcap_{n=0}^{\infty} S_n$ such that $q' \Longrightarrow z \xrightarrow{a} q''$, $p' \sim z$ and $p'' \sim q''$.

- Consider p'' such that $p' \xrightarrow{\tau} p''$ and put

$$\begin{aligned} S_n = \quad & \{ \ (z, q^*) \mid q' \Longrightarrow z \xrightarrow{\tau} q^*, \ p' \sim_n z, \ p'' \sim_n q^* \} \\ \cup \quad & \{ \ (q^*, q^*) \mid q' \Longrightarrow q^*, \ p' \sim_n q^*, \ p'' \sim_n q^* \} \end{aligned}$$

  and continue analogously to the previous case.

- Consider a transition $p' \xrightarrow{a} \surd$, where $a \in A_\tau$. Then

$$\begin{aligned} p' \xrightarrow{a} \surd &\iff \pi_1(p') \xrightarrow{a} \surd \\ &\iff \pi_1(q') \xrightarrow{a} \surd \iff q' \xrightarrow{a} \surd \end{aligned}$$

And by symmetry we have shown that $\sim$ is indeed a branching bisimulation.

It is left to show that $\sim \cup\{(p,q)\}$ is a branching bisimulation that is rooted w.r.t. (p,q).

Consider a transition $p \xrightarrow{a} p'$ $(a \in A)$ then for $n > 0$ we have as well $\pi_n(p) \xrightarrow{a} \pi_{n-1}(p')$ and by the definition of (strongly) rootedness there is a q'_n such that $\pi_n(q) \xrightarrow{a} q'_n$ and $\pi_{n-1}(p') \leftrightarrow_b q'_n$. Since this holds for all n and since q is guarded, there must be a q' such that $q \xrightarrow{a} q'$ and $q'_n \equiv \pi_{n-1}(q')$. Hence for all m we have $\pi_m(p') \leftrightarrow_b \pi_m(q')$, and thus for all m $p' \sim_m q'$ and thus $p' \sim q'$ as well.

The cases $p \xrightarrow{\tau} p'$ and $p \xrightarrow{a} \surd$ $(a \in A_\tau)$ are left to the reader. □

And finally we can prove the soundness of RSP^E_G.

Theorem 1.5.12 (Soundness of $\mathbf{RSP}^E_G$) *If E is a guarded specification with $\overline{X} \subseteq rvar(E)$ such that both $\overline{p}, \overline{q} \subseteq T(rvar(E), \mathrm{BPA}\delta\tau)$ are solutions for $\overline{X}$ in E modulo $\leftrightarrow_{rb}$, then $\overline{p} \leftrightarrow^E_{rb} \overline{q}$.*

Proof. Direct by the Projection Lemma and the Soundness of AIP^E_G. □

Part II

Prefix Integrated Real Time ACP

2

BPA with Time Stamps

2.1 Introduction

In this chapter we present the syntax, semantics and axiomatization of Baeten and Bergstra's ([BB91]) BPA$\rho\delta$, that is Basic Real Time Process Algebra with time stamped actions. So, we will consider processes like $a(1)$ (action a at time 1). The treatment of processes like $\int_{v \in \langle 0,1 \rangle} a(v)$ (action a in between time 0 and time 1) is postponed till Chapter 4.

In this thesis we restrict ourselves mainly to *absolute time.* That is, the time stamps are interpreted from the start of the whole process. In absolute time $a(2) \cdot b(1)$ is equal to $a(2) \cdot \delta$, as first the a action is executed at time 2 after which the b action cannot be executed any more at time 1. Baeten and Bergstra have shown as well how to deal with *relative time* in [BB91]. In relative time the time stamps are interpreted to be the time distance with respect to the previous action, where a process is supposed to start at time 0. Baeten and Bergstra write square brackets for relative time. Thus the relative time term $a[1] \cdot b[3]$ corresponds to the absolute time term $a(1) \cdot b(4)$. In [BB92] and [BB93a] Baeten and Bergstra deal with relative time using the *initial abstraction operator* $\surd v.p(v)$, which denotes a function from time stamps to processes; the process $a[1] \cdot b[3]$ corresponds to $\surd v.a(v{+}1) \cdot b(v{+}4)$. Most other papers on timed process algebras, such as the timed CCS papers [MT90],[MT92],[Wan91a],[Che92] and the timed CSP paper [DS89], use relative time. In Chapter 11 we discuss the relation between these papers and Real Time ACP.

In Section 2.2 of this chapter we introduce the syntax of BPA$\rho\delta$; we introduce the set of time stamped actions and we encounter a new operator, the *initialization operator* denoted by $t \gg p$. Furthermore, we introduce the *ultimate delay*, denoted by $U(p)$, which is, intuitively, the upper bound of points in time to which p can idle.

Section 2.3 presents Baeten and Bergstra's original operational semantics (see [BB91]) in which all, uncountably many, idle transitions are explicit in the transition systems. Since these transition systems cannot be drawn, the behavior is given by so called process diagrams. In Section 2.4 we discuss a timed bisimulation equivalence and we give two different, but equivalent, characterizations.

Section 2.5 we give the axiom system of BPA$\rho\delta$, which will basically be an extension of BPAδ.

In Section 2.6 we give an alternative semantics, which is called the *term semantics*, which does not have idle transitions in the transition systems. The advantage is that the proof techniques which we have seen in Chapter 1 can be used again. The idle behavior of a process term is expressed by a predicate, that corresponds with the *ultimate delay*.

The last section defines a notion of basic terms, which becomes more advanced than in the untimed case since all time stamps have to be taken into account. A (timed) basic term is a process term with increasing time stamps. Using these basic terms we are, finally, able to prove completeness of the axiom system BPA$\rho\delta$ w.r.t. to bisimulation equivalence.

2.2 A Syntax with Time Stamped Actions

Let A be the set of actions, not containing the constants δ and ι. The symbol ι will be used as label in the operational semantics. A_δ denotes $A \cup \{\delta\}$, similarly we have A_ι. In the tables in which the action rules for the operational semantics are given, we let a_δ range over A_δ, and we let a_ι range over A_ι.

As time domain we assume a set *Time* provided with an ordering $<$. In the examples we assume that the natural numbers are part of *Time*, and that $<$ has its usual meaning. In case *Time* contains a least element, then we denote this element by $\bot$.

The initialization operator, $\gg$, takes a $t \in$ *Time* and a process term; $t \gg p$ denotes that part of p, which starts after t. All initial actions before or at t are blocked.

The set $T(\text{BPA}\rho\delta)$, with typical elements p, p_1, p_2, is defined in the following way, where $a \in A_\delta, t \in$ *Time*.

$$p := a(t) \mid p_1 + p_2 \mid p_1 \cdot p_2 \mid t \gg p$$

2.3 A Semantics with Idle Transitions

The semantics of [BB91] assigns to every term in $T(\text{BPA}\rho\delta)$ a transition system in which each state is a pair consisting of a process term and a point in time, and in which each transition is labeled by a timed (non δ) action. Within this semantics each transition system concerns two relations

$$\begin{array}{llllll} \textit{Step} & \subseteq & (T(\text{BPA}\rho\delta) \times \textit{Time}) & \times & (A_\iota \times \textit{Time}) & \times & (T(\text{BPA}\rho\delta) \times \textit{Time}) \\ \textit{Terminate} & \subseteq & (T(\text{BPA}\rho\delta) \times \textit{Time}) & \times & (A \times \textit{Time}) & & \end{array}$$

These two relations are defined as the least relations satisfying the action rules given in Table 2.1. We write

$$\frac{t < r}{< a(r), t > \xrightarrow{a(r)} \surd} \qquad \frac{t < r < s}{< a_\delta(s), t > \xrightarrow{\iota(r)} < a_\delta(s), r >}$$

$$\frac{< p, t > \xrightarrow{a(r)} < p', r >}{< p + q, t > \xrightarrow{a(r)} < p', r >} \qquad \frac{< p, t > \xrightarrow{a(r)} \surd}{< p + q, t > \xrightarrow{a(r)} \surd}$$

$$\frac{< p, t > \xrightarrow{a(r)} < p', r >}{< q + p, t > \xrightarrow{a(r)} < p', r >} \qquad \frac{< p, t > \xrightarrow{a(r)} \surd}{< q + p, t > \xrightarrow{a(r)} \surd}$$

$$\frac{< p, t > \xrightarrow{\iota(r)} < p, r >}{< p + q, t > \xrightarrow{\iota(r)} < p + q, r >} \qquad \frac{< p, t > \xrightarrow{\iota(r)} < p, r >}{< q + p, t > \xrightarrow{\iota(r)} < q + p, r >}$$

$$\frac{< p, t > \xrightarrow{a_\iota(r)} < p', r >}{< p \cdot q, t > \xrightarrow{a_\iota(r)} < p' \cdot q, r >} \qquad \frac{< p, t > \xrightarrow{a(r)} \surd}{< p \cdot q, t > \xrightarrow{a(r)} < q, r >}$$

$$\frac{s < r \quad < p, t > \xrightarrow{a(r)} < p', r >}{< s \gg p, t > \xrightarrow{a(r)} < p', r >} \qquad \frac{s < r \quad < p, t > \xrightarrow{a(r)} \surd}{< s \gg p, t > \xrightarrow{a(r)} \surd}$$

$$\frac{t < r < s}{< s \gg p, t > \xrightarrow{\iota(r)} < s \gg p, r >} \qquad \frac{s \leq r \quad < p, t > \xrightarrow{\iota(r)} < p', r >}{< s \gg p, t > \xrightarrow{\iota(r)} < p', r >}$$

$$(a \in A,\ a_\delta \in A_\delta,\ a_\iota \in A_\iota,\ r, t, s \in \mathit{Time})$$

Table 2.1: Action Rules for idle semantics for BPA$\rho\delta$

$$\begin{array}{lll} < p, t > \xrightarrow{a(r)} < p', t' > & \text{for} \quad (< p, t >, (a, r), < p', t' >) & \in \mathit{Step} \\ < p, t > \xrightarrow{a(r)} \surd & \text{for} \quad (< p, t >, (a, r)) & \in \mathit{Terminate} \end{array}$$

We always have $t' = r$ in *Step*. Moreover, $< p, t > \xrightarrow{\iota(r)} < p', r >$ implies that p' is the same process term as p and we call it an *idle* transition.

The term $a(1)$ denotes the process that performs an action at time 1, after which it is successfully terminated.

From $< a(1), t_0 >$ an idle transition is possible to a state of the form $< a(1), t_1 >$ with $t_0 < t_1 < 1$. An idle transition is a transition that increases the time component only, without the execution of an action. Furthermore, from each state $< a(1), t >$ a $a(1)$-transition to $\surd$ is possible whenever $t < 1$. Since this semantics is based on the notion of an idle transition we refer to this semantics as *idle semantics*. In a

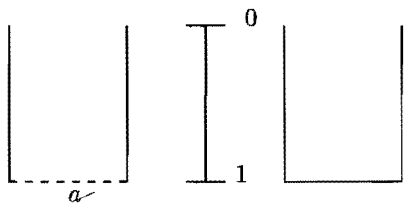


Figure 2.1: Process diagrams of the terms $a(1)$ and $\delta(1)$

later section we will introduce a semantics without these idle transitions.

The transition system of the term $a(1)$ can be represented by the left-hand process diagram given in Figure 2.1. A process diagram is simply a pictorial representation of a transition system. It is not possible to make a picture of the transition system itself, since it has uncountably many transitions. The intuition behind such a process diagram is that the process can idle by going to a lower point without crossing any line, whereas the execution of an action a at time r is reflected by going to a dashed line at level r labeled with a. Only dashed lines may be crossed, after landing on them.

In this thesis we do not assume that time starts at zero, as Baeten and Bergstra do [BB91]. If time contains negative number as well, then the action $a(-1)$ can be executed in a state with time $t < -1$. Our process diagrams are open at the top, which expresses that we do not assume any start time.

The process term $\delta(1)$ can do nothing more then idling until 1. From each state $< \delta(1), t_0 >$ an idle transition to $< \delta(1), t_1 >$ is possible, whenever $t_0 < t_1 < 1$.

The transition system of $a(1) + b(2)$ can be represented by the process diagram given in Figure 2.2. A state μ (in Figure 2.2) is of the form $< a(1) + b(2), t >$ with $0 < t < 1$. From μ both a terminating $a(1)$-transition to $\surd$ and a terminating $b(2)$-transition to $\surd$ are possible. However, from a state like ν of the form $< a(1) + b(2), t >$ with $1 \leq t < 2$ only a terminating $b(2)$-transition to $\surd$ is possible. Hence, by idling from $< a(1) + b(2), t_0 >$ to $< a(1) + b(2), t_1 >$ with $0 \leq t_0 < 1 \leq t_1 < 2$ we have lost the option of executing the $a(1)$-summand. Thus one could say that a choice has been made at time 1; after the choice has been made for $b(2)$ the summand $a(1)$ has become redundant.

The transition system of $a(1) + \delta(1)$ has exactly the same transitions as the transition system of $a(1)$. The summand $\delta(1)$ contributes only idle steps which are contributed by the summand $a(1)$ as well.

However if we consider $a(1)+\delta(2)$, the $\delta(2)$ summand contributes idle transitions which are not contributed by $a(1)$, since $\delta(2)$ has idle transitions to points in time between 1 and 2. The transition system of $a(1) + \delta(2)$ can be represented by the process diagram on the right-hand side in Figure 2.2.

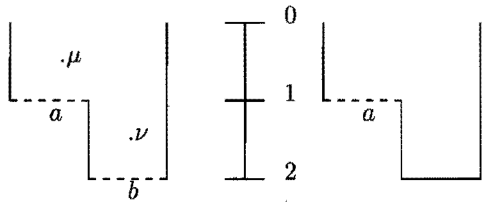


Figure 2.2: Process diagrams of the terms $a(1) + b(2)$ and $a(1) + \delta(2)$

$s \gg p$ denotes the process that idles till s, after which it evolves in that part of p that starts after s. So, if p has no initial actions later then s, then $s \gg p$ equals $\delta(s)$.

Proposition 2.3.1 $a \in A$

$$\begin{array}{lll} <p,t> \xrightarrow{a(r)} \surd & \Longrightarrow & t < r \\ <p,t> \xrightarrow{\iota(r)} <p',t'> & \Longrightarrow & t < r \ \wedge \ t' = r \ \wedge \ p \equiv p' \\ <p,t> \xrightarrow{a(r)} <p',t'> & \Longrightarrow & t < r \ \wedge \ t' = r \end{array}$$

Proof. These statements can be proven by induction on the derivation. □

2.4 Timed Idle Bisimulation Equivalence

The definition of bisimulation for the timed case is analogous to the one of the untimed case. We use the adjective *idle* to stress the fact that the underlying transition systems contain idle steps.

Definition 2.4.1 (Timed Idle Bisimulation)
$\mathcal{R} \subseteq (T(\text{BPA}\rho\delta) \times Time)^2$ *is a* timed idle bisimulation *if whenever* $<p,t> \mathcal{R} <q,t>$ *then*

1. $<p,t> \xrightarrow{a_\iota(r)} <p',r>$ *implies that there is a* q' *such that* $<q,t> \xrightarrow{a_\iota(r)} <q',r>$ *and* $<p',r> \mathcal{R} <q',r>$.
2. $<p,t> \xrightarrow{a(r)} \surd$ *implies that* $<q,t> \xrightarrow{a(r)} \surd$.
3. *Respectively (1) and (2) with the role of p and q interchanged.*

Definition 2.4.2 (Timed Idle Bisimulation Equivalence)
$<p,t> \leftrightarroweq^{\iota}_{p} <q,t>$ *iff there is a timed idle bisimulation* $\mathcal{R}$ *such that* $<p,t> \mathcal{R} <q,t>$.

This definition induces an equivalence relation on process terms, by putting $p \leftrightarrow^{\iota}_{\rho} q$ iff $\forall t \ <p,t> \ \leftrightarrow^{\iota}_{\rho} \ <q,t>$. The fact that $\leftrightarrow^{\iota}_{\rho}$ is a congruence over BPA$\rho\delta$ will be discussed later.

In the rest of this thesis we will consider only *real time* process algebra and we allow ourselves not to write the adjective timed if we consider a timed bisimulation or timed bisimulation. Similarly we will write $\leftrightarrow^{\iota}$ while we mean $\leftrightarrow^{\iota}_{\rho}$.

2.5 The Axiom System BPA$\rho\delta$

BPA$\rho\delta$ is the theory of Basic Real Time Process Algebra ([BB91]), see Table 2.2. It consists of the axioms A1-5, which are the standard axioms of BPA, and timed versions of A6 and A7 (see Table 1.3). The reformulated version of axiom A6 depends on the time information of the terms and therefore we add a ρ to its name. On the other hand, A7 is changed as well, the δ is changed into a $\delta(t)$, this change depends on the fact that the alphabet is now $A_\delta \times \mathit{Time}$ instead of A_δ. Hence, the reformulation does not depend on any time information we do not add a ρ in this case.

Most of the axioms occur already in [BB91], though Baeten and Bergstra use different names. A6$_\rho$ does not occur in [BB91], Baeten and Bergstra use the following two axioms instead

$$\begin{array}{lrcl} t<r & \delta(t)+\delta(r) & = & \delta(r) \\ & a(t)+\delta(t) & = & a(t) \end{array}$$

The axiom RT0$_\rho$ originates from [BB91] as well, where it was formulated by $a(0) = \delta(0)$, as in that paper 0 is the least element of the time domain. We put this axiom in brackets, as we need it only in case *Time* has a least element.

Furthermore we have some axioms stating the specific real-time properties and defining the initialization operator.

Example 2.5.1 *Within* BPA$\rho\delta$ *we can prove:*

$$\begin{array}{lcl} 5 \gg (a(4)+b(6)+c(7)\cdot d(8)) & = & b(6)+c(7)\cdot d(8) \\ 5 \gg (a(4)+b(3)) & = & \delta(5) \\ \delta(1)+a(2)\cdot b(3)+\delta(3)\cdot c(4) & = & a(2)\cdot b(3)+\delta(3) \\ a(-1)+b(2)\cdot(c(1)+c(3))+d(3)\cdot e(2) & = & a(-1)+b(2)\cdot c(3)+d(3)\cdot\delta(3) \end{array}$$

The Ultimate Delay

Intuitively, the ultimate delay of p is the upper limit of points to which it can idle. It is defined within the theory BPA$\rho\delta$I. The ultimate delay has already been introduced by Baeten and Bergstra in [BB91]. In [MT90] Moller & Tofts have introduced a similar construct, which they call the maximum delay.

Note that we can formulate axiom A6$_\rho$ as well by

$$\text{A6}'_\rho \quad r \le U(p) \quad p+\delta(r) = p$$

A1		$p+q$	$=$	$q+p$
A2		$(p+q)+z$	$=$	$p+(q+z)$
A3		$p+p$	$=$	p
A4		$(p+q)\cdot z$	$=$	$p\cdot z+q\cdot z$
A5		$(p\cdot q)\cdot z$	$=$	$p\cdot(q\cdot z)$
A6$_\rho$	$t\geq r$	$a(t)+\delta(r)$	$=$	$a(t)$
A7		$\delta(t)\cdot p$	$=$	$\delta(t)$
(RT0$_\rho$)		$a(\bot)$	$=$	$\delta(\bot)$
RT1$_\rho$		$a(t)\cdot p$	$=$	$a(t)\cdot(t\gg p)$
RT2$^a_\rho$	$t<r$	$t\gg a(r)$	$=$	$a(r)$
RT2$^b_\rho$	$t\geq r$	$t\gg a(r)$	$=$	$\delta(t)$
RT3$_\rho$		$t\gg(p+q)$	$=$	$(t\gg p)+(t\gg q)$
RT4$_\rho$		$t\gg(p\cdot q)$	$=$	$(t\gg p)\cdot q$
U1		$U(p+q)$	$=$	$\max(U(p),U(q))$
U2		$U(a(t))$	$=$	t
U3		$U(p\cdot q)$	$=$	$U(p)$

$(a\in A_\delta,\ r,t\in \mathit{Time})$

Table 2.2: An axiom system for BPA$\rho\delta$

2.6 A Term Semantics for BPA$\rho\delta$

In Section 2.3 we have presented an operational semantics in which the transition systems contain idle transitions. In that section each state was a pair of a process term and a time stamp; an idle transition increased the time while the term remained the same. We now define an operational semantics without idle transitions, which induces the same bisimulation equivalence. The action rules are analogous to the action rules of the operational semantics for untimed BPA, see Table 1.1. Since each state will be process term the semantics is called the *term semantics*.

In untimed BPAδ,as given in Section 1.2.2, we encountered the following transition

$$a \xrightarrow{a} \surd \quad\text{and}\quad a\cdot p \xrightarrow{a} p$$

In a real-time setting we have to take the time stamps into account. Consider the term $a(r)\cdot p$. After executing the $a(r)$-action only that part of p can be executed which starts after r. We will have the following transitions:

$$a(r) \xrightarrow{a(r)} \surd \quad \text{and} \quad a(r) \cdot p \xrightarrow{a(r)} r \gg p$$

In Figure 2.3 the transition systems for the terms $a(1)$ and $a(1) + b(2)$ are given, together with the corresponding process diagrams.

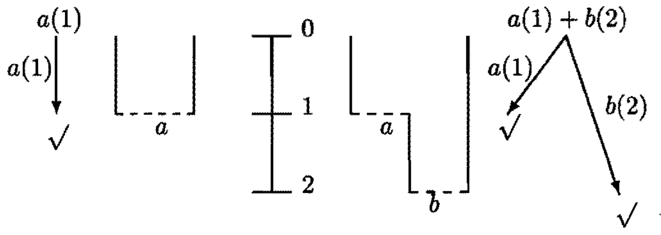


Figure 2.3: Process diagrams and transition systems for the terms $a(1)$ and $a(1) + b(2)$

However, we have to deal somehow with the idle behavior of the process terms, as $a(1)$ must be distinguished from $a(1) + \delta(2)$. Therefore, we introduce a predicate, $U_t(p)$, that corresponds with the idle semantics in the following way:

$$< p, r > \xrightarrow{\iota(t)} < p, t > \quad \Longleftrightarrow \quad U_t(p)$$

In Table 2.3 the action rules of the term semantics are given. Every state is a process term from $T(\text{BPA}\rho\delta)$ and every transition is labeled by a timed atomic action $a(r)$ where $a \in A$. The term semantics concerns two relations, and one predicate:

$$\begin{array}{lll} \textit{Step} & \subseteq & T(\text{BPA}\rho\delta) \times (A \times \textit{Time}) \times T(\text{BPA}\rho\delta) \\ \textit{Terminate} & \subseteq & T(\text{BPA}\rho\delta) \times (A \times \textit{Time}) \\ U & \subseteq & T(\text{BPA}\rho\delta) \times \textit{Time} \end{array}$$

We write:

$$\begin{array}{lll} p \xrightarrow{a(r)} p' & \text{for} & (p, (a, r), p') \in \textit{Step} \\ p \xrightarrow{a(r)} \surd & \text{for} & (p, (a, r)) \in \textit{Terminate} \\ U_t(p) & \text{for} & (p, t) \in U \end{array}$$

The transition relations *Step*, *Terminate* and the predicate U are defined as the least relations satisfying the action rules of Table 2.3.

We define a bisimulation on these transition systems.

Definition 2.6.1 (Term Bisimulation)
$\mathcal{R} \subset T(\text{BPA}\rho\delta) \times T(\text{BPA}\rho\delta)$ *is a* bisimulation *if whenever* $p\mathcal{R}q$ *then*

1. $p \xrightarrow{a(r)} p'$ *implies* $\exists q'$ *such that* $q \xrightarrow{a(r)} q'$ *and* $p'\mathcal{R}q'$.

$a(r) \xrightarrow{a(r)} \surd$		$\dfrac{t < r}{U_t(a_\delta(r))}$
$\dfrac{p \xrightarrow{a(r)} \surd}{p \cdot q \xrightarrow{a(r)} r \gg q}$	$\dfrac{p \xrightarrow{a(r)} p'}{p \cdot q \xrightarrow{a(r)} p' \cdot q}$	$\dfrac{U_t(p)}{U_t(p \cdot q)}$
$\dfrac{p \xrightarrow{a(r)} \surd}{p + q \xrightarrow{a(r)} \surd}$	$\dfrac{p \xrightarrow{a(r)} p'}{p + q \xrightarrow{a(r)} p'}$	$\dfrac{U_t(p)}{U_t(p + q)}$
$\dfrac{p \xrightarrow{a(r)} \surd}{q + p \xrightarrow{a(r)} \surd}$	$\dfrac{p \xrightarrow{a(r)} p'}{q + p \xrightarrow{a(r)} p'}$	$\dfrac{U_t(p)}{U_t(q + p)}$
$\dfrac{t < r \quad p \xrightarrow{a(r)} \surd}{t \gg p \xrightarrow{a(r)} \surd}$	$\dfrac{t < r \quad p \xrightarrow{a(r)} p'}{t \gg p \xrightarrow{a(r)} p'}$	
$\dfrac{t < r}{U_t(r \gg p)}$	$\dfrac{U_t(p)}{U_t(r \gg p)}$	

$(a \in A, \ , a_\delta \in A_\delta, \ r, t \in Time)$

Table 2.3: Action Rules for term semantics for BPA$\rho\delta$

2. $p \xrightarrow{a(r)} \surd$ *implies* $q \xrightarrow{a(r)} \surd$.

3. $U_t(p)$ *implies* $U_t(q)$.

4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

Bisimulation equivalence is now defined as follows:

Definition 2.6.2 $p \leftrightarroweq q$ *if there exists a bisimulation* $\mathcal{R}$ *relating* p *and* q.

And we obtain directly the congruency of $\leftrightarroweq$.

Theorem 2.6.3 (Congruency of $\leftrightarroweq$) $\leftrightarroweq$ *is a congruence over* BPA$\rho\delta$

Proof. The action rules of Table 2.3 are in the *path* format ([BV93]). □

In the rest of this section we discuss the correspondence to the idle semantics of Section 2.3. The main difference between these two operational semantics is how the course of time is recorded. Consider the following two applications of the respective actions rules for sequential composition.

$$\frac{< a(1)\cdot p, r > \xrightarrow{a(1)} < p, 1 >}{< (a(1)\cdot p)\cdot q, r > \xrightarrow{a(1)} < p\cdot q, 1 >}$$

and

$$\frac{a(1)\cdot p \xrightarrow{a(1)} 1 \gg p}{(a(1)\cdot p)\cdot q \xrightarrow{a(1)} (1 \gg p)\cdot q}$$

We see in the latter case that the course of time is encoded in the prefix by an application of the initialization operator. To relate the two semantics formally we define the functions *strip* and *time* on process terms, where we assume a symbol $-\infty \notin Time$.

Definition 2.6.4

$$\begin{array}{lcllcl} strip(a(r)) & = & a(r) & time(a(r)) & = & -\infty \\ strip(p+q)) & = & p+q & time(p+q) & = & -\infty \\ strip(p\cdot q) & = & strip(p)\cdot q & time(p\cdot q) & = & time(p) \\ strip(r \gg p) & = & p & time(r \gg p) & = & r \end{array}$$

We state

Proposition 2.6.5 *$p \in T(\mathrm{BPA}\rho\delta)$ such that $time(p) \neq -\infty$.*

$$\begin{array}{l} p \xrightarrow{a(r)} \surd \iff < strip(p), time(p) > \xrightarrow{a(r)} \surd \\ p \xrightarrow{a(r)} p' \implies < strip(p), time(p) > \xrightarrow{a(r)} < strip(p'), r > \wedge time(p') = r \\ < strip(p), time(p) > \xrightarrow{a(r)} < p', r > \\ \qquad \implies \exists p''\ p \xrightarrow{a(r)} p'' \wedge strip(p'') \equiv p' \wedge time(p'') = r \end{array}$$

Proof. The statements can be proven by induction on the length of the derivation. □

Furthermore we need one property of $\leftrightarrows$.

Lemma 2.6.6 $p, q \in T(\mathrm{BPA}\rho\delta)$, $r \in Time$

$$p \leftrightarrows q \quad \implies \quad r \gg p \leftrightarrows r \gg q$$

Proof. Omitted. □

Using this proposition we can finally prove:

Lemma 2.6.7

$$p \leftrightarrows q \quad \iff \quad \forall t \ < p, t > \ \leftrightarrows^t \ < q, t >$$

Proof.

$\Longrightarrow$ $p \leftrightarrows q$ implies that $\forall t\ t \gg p \leftrightarrows t \gg q$. Hence, we can take R_t such that $R_t\ :\ t \gg p \leftrightarrows t \gg q$. We construct R_t^ι as follows

$$\{(< strip(p'), time(p') >, < strip(q'), time(q') >) | (p', q') \in R_t\}$$

$\Longleftarrow$ Take R_t^ι such that $R_t^\iota\ : < p, t > \ \leftrightarrows^\iota\ < q, t >$ and let R be

$$\{(p', q') | \exists t\ (< strip(p'), time(p') >, < strip(q'), time(q') >) \in R_t^\iota\} \cup \{(p, q)\}$$

It is left to the reader to prove that the constructed relations are indeed bisimulations. □

The following theorem says that the theory BPA$\rho\delta$ is *sound*. This means that if BPA$\rho\delta \vdash p = q$ then $p \leftrightarrows q$. Since we have already shown that $\leftrightarrows$ is a congruence, it is sufficient to prove for each axiom that if it proves two terms equal, then these terms are bisimilar as well.

Theorem 2.6.8 (Soundness of BPA$\rho\delta$**)** $p, q \in T(\text{BPA}\rho\delta)$

$$\text{BPA}\rho\delta \vdash p = q \quad \Longrightarrow \quad p \leftrightarrows q$$

Proof. To prove that any of the axioms A1-A5 is sound w.r.t. $\leftrightarrows_\rho$ is similar as proving that such an axiom is sound w.r.t. $\leftrightarrows$ (untimed bisimulation equivalence), see Theorem 1.2.6. Below we discuss some of the other axioms of BPA$\rho\delta$, the axioms which are left out are left to the reader.

- A6$_\rho$ $t \geq r\ :\ a(t) + \delta(r) = a(t)$. Both process terms have only one transition, namely $\xrightarrow{a(t)} \surd$. $U_s(a(t) + \delta(r))$ implies that $s \leq \max(t, r) = t$ and thus $U_s(a(t))$ as well. Similarly, we can show that $U_s(a(t))$ implies $U_s(a(t) + \delta(r))$.
- A7 is trivial since neither $\delta(t)$ nor $\delta(t) \cdot p$ has any transitions and by definition of U_s we have $U_s(\delta(t) \cdot p)$ iff $U_s(\delta(t))$.
- RT1$_\rho$ $a(t) \cdot p = a(t) \cdot (t \gg p)$. By definition of U_s we have $U_s(a(t) \cdot p)$ iff $U_s(a(t) \cdot (t \gg p))$. Both processes have only one $a(t)$ transition to resp. $t \gg p$ and $t \gg (t \gg p)$, hence, it is sufficient to prove that these latter two terms are bisimilar.
  - Consider a transition $t \gg p \xrightarrow{b(r)} z$, then by the right hand side action rule for $\gg$ we know that $r > t$ and by the same rule we obtain that $t \gg (t \gg p) \xrightarrow{b(r)} z$ as well. Similarly, $t \gg p \xrightarrow{b(r)} \surd$ implies $r > t$ and thus $t \gg (t \gg p) \xrightarrow{b(r)} \surd$ as well.
  - Consider a transition $t \gg (t \gg p) \xrightarrow{b(r)} z$, then by the right hand side action rule for $\gg$ we know that $t \gg p \xrightarrow{b(r)} z$ as well. Similarly, if $t \gg (t \gg p) \xrightarrow{b(r)} \surd$ then it must be the case that $t \gg p \xrightarrow{b(r)} \surd$.

□

Finally, we have a proposition that states the correspondence between $U(p)$ and $U_t(p)$.

Proposition 2.6.9 *Let $p \in T(\mathrm{BPA}\rho\delta)$ and $t \in Time$, then*

$$U_t(p) \iff U(p) > t$$

Proof. Omitted. □

2.7 Basic Terms and Completeness

In this section we prove that BPA$\rho\delta$ is *complete.* This means that if $p \leftrightarrow q$, then BPA$\rho\delta \vdash p = q$. As in Chapter 1 we first show that each process term can be reduced to a *basic* form.

2.7.1 Basic terms

We extend the definition of *head normal forms* and *prefix normal forms* to BPA$\rho\delta$. Since these definitions are analogous to the untimed case, we refer the reader to 1.2.10. Next, we prove that any term can be reduced to a prefix normal form.

Proposition 2.7.1 *For any $p \in T(\mathrm{BPA}\rho\delta)$ there is a prefix normal form p' such that* BPA$\rho\delta \vdash p = p'$

Proof. First we show that for a prefix normal form z and $r \in Time$ there is a prefix normal form u such that BPA$\rho\delta \vdash r \gg z = u$. We do this by induction on z. The following equations must be read from left to right, note that only in the last case induction must be applied.

$$\begin{array}{lcll}
r \gg a(t) & = & a(t) & \text{if } r < t \\
r \gg a(t) & = & \delta(r) & \text{if } r \geq t \\
r \gg (a(t) \cdot z') & = & a(t) \cdot z' & \text{if } r < t \\
r \gg (a(t) \cdot z') & = & \delta(r) & \text{if } r \geq t \\
r \gg (z_0 + z_1) & = & r \gg z_0 + r \gg z_1 &
\end{array}$$

Take prefix normal forms z, z', then we can show, as in the proof of Proposition 1.2.11, by induction on z, that there is a prefix normal form u such that BPA$\rho\delta \vdash z \cdot z' = u$.

Finally, we prove the general case by induction on the number of occurrences of $\gg$ and the number of occurrences of general multiplications (see the proof of 1.2.11). □

We have the following proposition :

Proposition 2.7.2 *Let p be a prefix normal form then*

$$\begin{array}{lll} p \xrightarrow{a(r)} p' & \Longrightarrow & \exists p'' \;\; p' \equiv r \gg p'' \;\wedge\; a(r) \cdot p'' \sqsubseteq p \\ p \xrightarrow{a(r)} \surd & \Longrightarrow & a(r) \sqsubseteq p \end{array}$$

Proof. Omitted. □

In the untimed setting a basic term is a process term in prefix normal form, without subterms of the form $\delta \cdot p$. In the timed setting the definition of a basic term is more involved. In the completeness proof we will use that for a basic term p:

$$p \xrightarrow{a(r)} r \gg p' \quad \Longrightarrow r \gg p' \leftrightarrows p'$$

which means that a basic term must have *ascending* time stamps.

Definition 2.7.3 *For $p \in T(\mathrm{BPA}\rho\delta)$ and $r \in$ Time we define a boolean expression $\mathcal{B}(p,r)$ which reduces to either tt or ff. If p is a prefix normal form*

$$\sum_{i \in I} a_i(r_i) \cdot p_i + \sum_{j \in J} b_j(t_j)$$

then

$$\mathcal{B}(p,r) = \bigwedge_{i \in I} (r < r_i \;\wedge\; \mathcal{B}(p_i, r_i)) \;\wedge\; \bigwedge_{j \in J} (r < t_j)$$

where $\bigwedge_{i \in \emptyset} \alpha_i$ abbreviates tt.

Here, *tt* stands for *true*. $\mathcal{B}(p,r) = tt$ means that p is a basic term, with initial actions later than r. We write $p \in \mathcal{B}(r)$ if $\mathcal{B}(p,r) = tt$ and $p \in \mathcal{B}$ if $p \in \mathcal{B}(r)$ for some r, in which case we say that p is a basic term. The following proposition states the required properties:

Proposition 2.7.4 $p \in \mathcal{B}, a \in A$

$$a(r) \cdot p' \sqsubseteq p \quad \Longrightarrow \quad r \gg p' \leftrightarrows p'$$

Proof. It is sufficient to prove by induction on the size of q that $q \in \mathcal{B}(r)$ implies $r \gg q = q$, since $a(r) \cdot p' \sqsubseteq p$ implies $p' \in \mathcal{B}(r)$. □

Every term can be reduced a basic term.

Theorem 2.7.5 *For each term $p \in T(\mathrm{BPA}\rho\delta)$ there is a basic term p_b such that* $\mathrm{BPA}\rho\delta \vdash p = p_b$.

Proof. First, we prove by induction on the depth of z, where z is a prefix normal form, that for any r there is a basic term z^r such that $\mathrm{BPA}\rho\delta \vdash r \gg z = z^r$.

Assume

$$z \simeq \sum_{i \in I} a_i(r_i) \cdot z_i + \sum_{j \in J} b_j(t_j)$$

Take

$$\begin{array}{rcl} I' & = & \{i \in I \mid r_i > r \ \wedge \ a_i \neq \delta\} \\ J' & = & \{j \in J \mid t_j > r\} \end{array}$$

and we construct z^r such that

$$z^r \simeq \sum_{i \in I'} a_i(r_i) \cdot z_i^{r_i} + \sum_{j \in J'} b_j(t_j)$$

Next, we construct for p a prefix normal form p' such that BPA$\rho\delta \vdash p = p'$ in case *Time* does not contain a least element $\perp$. Otherwise, we construct a prefix normal form p' such that BPA$\rho\delta \vdash \perp \gg p = p'$.

Assume

$$p' \simeq \sum_{i \in I} a_i(r_i) \cdot p_i + \sum_{j \in J} b_j(t_j)$$

Take

$$I' = \{i \in I \mid a_i \neq \delta\}$$

and we construct p_b such that

$$p_b \simeq \sum_{i \in I'} a_i(r_i) \cdot p_i^{r_i} + \sum_{j \in J} b_j(t_j)$$

where $p_i^{r_i}$ is the basic term of $r_i \gg p_i$ as we have constructed in the first part of the proof. □

Basic terms occur already in [Klu91b] and [FK92], though in these papers deadlock summands are removed from a basic term whenever possible; the process terms $a(2) + \delta(1)$ and $a(2) + \delta(2)$ are not basic, they are equal to the basic term $a(2)$. In this thesis we allow more terms to be basic, such that the definition and the construction of basic terms can be simplified; the price to pay is that a few more remarks are to be made in the completeness proof.

2.7.2 Completeness of BPA$\rho\delta$

We can now prove that the theory BPA$\rho\delta$ is complete.

Theorem 2.7.6 (Completeness for BPA$\rho\delta$**)** $\forall p, q \in T(\text{BPA}\rho\delta)$

$$p \leftrightarrows q \quad \Longrightarrow \quad \text{BPA}\rho\delta \vdash p = q$$

Proof. Lemma 2.7.5 together with soundness implies that it is sufficient to consider basic terms only.

- Consider an arbitrary summand $a(r) \cdot p'$ of p. Since $a \neq \delta$, $p \xrightarrow{a(r)} r \gg p'$, and since $p \leftrightarroweq q$ there is q' such that $q \xrightarrow{a(r)} r \gg q'$ and $r \gg p' \leftrightarroweq r \gg q'$. Since p and q are basic terms we have

  $$p' \leftrightarroweq r \gg p' \leftrightarroweq r \gg q' \leftrightarroweq q'$$

  and by induction we obtain $p' = q'$. Since $a(r) \cdot q' \sqsubseteq q$ we may conclude $a(r) \cdot p' \subseteq_{\mathrm{BPA}\rho\delta} q$. Hence, $\sum_i a_i(t_i) \cdot p_i \subseteq_{\mathrm{BPA}\rho\delta} q$.

- Consider an arbitrary summand $a(r)$ of p such that $a \in A$. Then $p \xrightarrow{a(r)} \surd$ and since $p \leftrightarroweq q$ also $q \xrightarrow{a(r)} \surd$, from which we conclude $a(r) \sqsubseteq q$.

  Consider an arbitrary summand $\delta(r)$ of p. For any $t < r$ we have $U_t(p)$, hence, $U_t(q)$ as well. By proposition 2.6.9 we have $\forall t < r$ that $U(q) > t$ and thus $U(q) \geq r$, and by A6$'_\rho$ we obtain $q + \delta(r) = q$. Thus, $\delta(r) \subseteq_{\mathrm{BPA}\rho\delta} q$. Hence, $\sum_j b_j(s_j) \subseteq_{\mathrm{BPA}\rho\delta} q$.

So, $p \subseteq_{\mathrm{BPA}\rho\delta} q$ and by symmetry also $q \subseteq_{\mathrm{BPA}\rho\delta} p$ and thus $\mathrm{BPA}\rho\delta \vdash p = q$. $\square$

3

ACP with Time Stamps

3.1 Introduction

In this chapter we introduce parallelism and synchronization, resulting in the theory ACPρ from [BB91]. In Section 1.3 we have discussed ACP (without time) and we have presented the parallel merge ($\|$) and auxiliary operators such as the left merge ($\llfloor$), communication merge ($|$) and encapsulation (∂_H).

In our timed setting $p\|q$ can idle till r only if both p and q are able to idle till r. Similarly, $p\|q$ can execute an action at r, which originates from p, only if q is able to idle till r. Hence, the most important difference with the untimed case is the phenomenon that in a parallel composition both components most proceed equally in time. The same holds of course for the left merge and communication merge.

For the axiomatization of the left merge we introduce a new operator, the *bounded initialization*, which will be the dual of the initialization operator. $p \gg t$, the bounded initialization of p to t, denotes the process p whose initial behavior is restricted to the time before t, so, all initial actions of p at or after t are blocked.

In Section 3.2 we present the syntax for ACPρ, and we discuss this bounded initialization in more detail.

In Section 3.3 we give the action rules for the idle semantics. These action rules rules are straightforward adapted versions of the action rules for untimed ACP.

In Section 3.4 we give the axiom system for ACPρ. The requirement that both components of a parallel composition have to proceed in time equally, is expressed algebraically in the axioms for the left merge by applying the bounded initialization.

In Section 3.5 we give a term semantics for ACPρ and we obtain the congruence for bisimulation equivalence for free since the action rules are in the *path* format.

Finally, in Section 3.6 we prove the Elimination Theorem for ACPρ, which says that every term in ACPρ can be reduced to a basic term (which is in BPA$\rho\delta$). From this theorem the completeness of ACPρ follows directly from the completeness of BPA$\rho\delta$.

3.2 The Syntax of ACPρ

We discuss only those cases in which we have to take the time information into account. In untimed ACP the term $z \equiv (a \cdot p) \mathbin{\|\!\!\_} q$ denotes the process in which the left component $a \cdot p$ executes its first action a, after which z evolves to $p \| q$. In the real-time setting it is a bit more subtle. Consider the process $(a(t) \cdot p) \mathbin{\|\!\!\_} q$. The left component $a(t) \cdot p$ can execute the action a at time t only if q is able to idle till t. If not, then the whole process can idle only till the ultimate delay of q, because at that time q is not able to idle any further, while $a(t) \cdot p$ is. We will have the following identities:

$$\begin{aligned} a(2) \mathbin{\|\!\!\_} b(3) &= a(2) \cdot b(3) \\ b(3) \mathbin{\|\!\!\_} a(2) &= \delta(2) \end{aligned}$$

In the first example the right component $b(3)$ can wait until the left component $a(2)$ executes its first action. In the second example, however, we see that the right component $a(2)$ cannot wait long enough and a deadlock is the result.

The set $T(\text{ACP}\rho)$, of terms over ACPρ, is defined by the following BNF sentence, where $a \in A_\delta$, $t \in \mathit{Time}$ and $H \subseteq A$:

$$p \ ::= \ a(t) \mid p + p \mid p \cdot p \mid t \gg p \mid p \gg t \mid p \| p \mid p \mathbin{\|\!\!\_} p \mid p | p \mid \partial_H(p)$$

We inherit the communication function γ from Section 1.3. The communication function is applied only on a pair of atomic action with the same time stamps, as it does not make sense, according to [BB91], to have a communication between actions at different points in time. Thus if $\gamma(a, b) = c$ then

$$\begin{aligned} a(2) | b(2) &= c(2) \\ a(1) | b(3) &= \delta(1) \end{aligned}$$

3.3 An Idle semantics for ACPρ

In the Tables 3.1 and 3.2 the action rules for the idle semantics are given. Basically, the only difference with the action rules for untimed ACP, as given in Table 1.4, is that in a parallel composition one component can execute an action at time t only if the other component is able to idle till t. The action rules for the BPA$\rho\delta$ operators can be found in Table 2.1, where p, q now range over $T(\text{ACP}\rho)$.

$$\frac{<p,t> \xrightarrow{a_\iota(r)} <p',r> \quad <q,t> \xrightarrow{\iota(r)} <q,r>}{<p\|q,t> \xrightarrow{a_\iota(r)} <p'\|q,r>} \qquad \frac{<p,t> \xrightarrow{a_\iota(r)} <p',r> \quad <q,t> \xrightarrow{\iota(r)} <q,r>}{<q\|p,t> \xrightarrow{a_\iota(r)} <q\|p',r>}$$

$$\frac{<p,t> \xrightarrow{a(r)} \surd \quad <q,t> \xrightarrow{\iota(r)} <q,r>}{<p\|q,t> \xrightarrow{a(r)} <q,r>} \qquad \frac{<p,t> \xrightarrow{a(r)} \surd \quad <q,t> \xrightarrow{\iota(r)} <q,r>}{<q\|p,t> \xrightarrow{a(r)} <q,r>}$$

$$\frac{\gamma(a,b)=c \quad <p,t> \xrightarrow{a(r)} <p',r> \quad <q,t> \xrightarrow{b(r)} <q',r>}{<p\|q,t> \xrightarrow{c(r)} <p'\|q',r>} \qquad \frac{\gamma(a,b)=c \quad <p,t> \xrightarrow{a(r)} \surd \quad <q,t> \xrightarrow{b(r)} \surd}{<p\|q,t> \xrightarrow{c(r)} \surd}$$

$$\frac{\gamma(a,b)=c \quad <p,t> \xrightarrow{a(r)} \surd \quad <q,t> \xrightarrow{b(r)} <q',r>}{<p\|q,t> \xrightarrow{c(r)} <q',r>} \qquad \frac{\gamma(a,b)=c \quad <p,t> \xrightarrow{a(r)} \surd \quad <q,t> \xrightarrow{b(r)} <q',r>}{<q\|p,t> \xrightarrow{c(r)} <q',r>}$$

$$\frac{a \notin H \quad <p,t> \xrightarrow{a(r)} \surd}{<\partial_H(p),t> \xrightarrow{a(r)} \surd} \qquad \frac{a_\iota \notin H \quad <p,t> \xrightarrow{a_\iota(r)} <p',r>}{<\partial_H(p),t> \xrightarrow{a_\iota(r)} <\partial_H(p'),r>}$$

$(a,b,c \in A,\ a_\iota \in A_\iota,\ r,t \in Time,\ H \subseteq A)$

Table 3.1: Action rules for idle semantics for $\|$ and ∂_H

$$\frac{\begin{array}{c}< p,t > \xrightarrow{\iota(r)} < p,r > \\ < q,t > \xrightarrow{\iota(r)} < q,r >\end{array}}{< p \mathbin{\|\mkern-9mu\lfloor} q,t > \xrightarrow{\iota(r)} < p \mathbin{\|\mkern-9mu\lfloor} q,r >} \qquad \frac{\begin{array}{c}< p,t > \xrightarrow{\iota(r)} < p,r > \\ < q,t > \xrightarrow{\iota(r)} < q,r >\end{array}}{< p|q,t > \xrightarrow{\iota(r)} < p|q,r >}$$

$$\frac{\begin{array}{c}< p,t > \xrightarrow{a(r)} < p',r > \\ < q,t > \xrightarrow{\iota(r)} < q,r >\end{array}}{< p \mathbin{\|\mkern-9mu\lfloor} q,t > \xrightarrow{a(r)} < p' \| q,r >} \qquad \frac{\begin{array}{c}< p,t > \xrightarrow{a(r)} \surd \\ < q,t > \xrightarrow{\iota(r)} < q,r >\end{array}}{< p \mathbin{\|\mkern-9mu\lfloor} q,t > \xrightarrow{a(r)} < q,r >}$$

$$\frac{\begin{array}{l}\gamma(a,b) = c \\ < p,t > \xrightarrow{a(r)} < p',r > \\ < q,t > \xrightarrow{b(r)} < q',r >\end{array}}{< p|q,t > \xrightarrow{c(r)} < p' \| q',r >} \qquad \frac{\begin{array}{l}\gamma(a,b) = c \\ < p,t > \xrightarrow{a(r)} \surd \\ < q,t > \xrightarrow{b(r)} \surd\end{array}}{< p|q,t > \xrightarrow{c(r)} \surd}$$

$$\frac{\begin{array}{l}\gamma(a,b) = c \\ < p,t > \xrightarrow{a(r)} \surd \\ < q,t > \xrightarrow{b(r)} < q',r >\end{array}}{< p|q,t > \xrightarrow{c(r)} < q',r >} \qquad \frac{\begin{array}{l}\gamma(a,b) = c \\ < p,t > \xrightarrow{a(r)} \surd \\ < q,t > \xrightarrow{b(r)} < q',r >\end{array}}{< q|p,t > \xrightarrow{c(r)} < q',r >}$$

$$\frac{r < s \quad < p \gg s,t > \xrightarrow{a(r)} \surd}{< p \gg s,t > \xrightarrow{a(r)} \surd} \qquad \frac{r < s \quad < p,t > \xrightarrow{a(r)} < p',r >}{< p \gg s,t > \xrightarrow{a(r)} < p',r >}$$

$$\frac{r < s \quad < p,t > \xrightarrow{\iota(r)} < p,r >}{< p \gg s,t > \xrightarrow{\iota(r)} < p \gg s,r >}$$

$(a, b, c \in A,\ r, t, s \in \mathit{Time})$

Table 3.2: Action rules for idle semantics for left merge, communication merge and $\gg$

3.4 The Axiom System ACPρ

The axiom system for ACPρ consists of BPA$\rho\delta$ together with the axioms of Table 3.3. The names of the axioms have been taken from untimed ACP.

The axioms of the left merge use the *bounded initialization*. The axiom

$$\text{CM3} \quad (a \cdot p) \mathbin{⌊\!⌊} q = a \cdot (p \| q)$$

can be reformulated by[1]

$$\text{CM3}_\rho \quad (a(t) \cdot p) \mathbin{⌊\!⌊} q = (a(t) \gg U(q)) \cdot (p \| q)$$

since

$$a(t) \gg U(q)$$

means intuitively

$a(t)$ only if q is able to idle till t, otherwise a deadlock at the moment that q cannot idle any further.

The axioms RT6-9, that define the bounded initialization, are very similar to the axioms RT2-4 (see Table 2.2) which define the initialization operator. Only the conditions for the atomic cases have to be changed.

The axiom CM1 is exactly the same as in ACP. However, together with the axioms for the left merge it does not result in arbitrary interleaving, since the time stamps of the atomic actions determine the possible orderings. For example

$$\begin{aligned} a(2) \| b(3) &= a(2) \mathbin{⌊\!⌊} b(3) + b(3) \mathbin{⌊\!⌊} a(2) + a(2) | b(3) \\ &= a(2) \cdot b(3) + \delta(2) + \delta(2) \\ &= a(2) \cdot b(3) \end{aligned}$$

3.5 A Term Semantics for ACPρ

Table 3.4 contains the rules for the $U_t(p)$ predicate. Table 3.5 contains the action rules for the term semantics for the new operators $\|, |, \mathbin{⌊\!⌊}$ and ∂_H. The action rules for the operators of BPA$\rho\delta$ can be found in Table 2.3, where p, q now range over $T(\text{ACP}\rho)$.

Note that we have in the idle semantics $< p, t > \xrightarrow{\iota(r)} < p', r >$ iff we have in the term semantics $U_r(p)$. Hence, the rule

[1]In [FK92] an axiomatization is given without introducing the bounded initialization. There we had two axioms which correspond with CM3$_\rho$, namely CM3$^a_\rho$, $U(q) > t \;:\; (a(t) \cdot p) \mathbin{⌊\!⌊} q = a(t) \cdot (p \| q)$, and CM3$^b_\rho$, $U(q) \leq t \;:\; (a(t) \cdot p) \mathbin{⌊\!⌊} q = \delta(U(q))$. Similarly, we had two axioms which correspond with CM2$_\rho$. We have chosen to follow Baeten and Bergstra ([BB91]) here since CM2$_\rho$ and CM3$_\rho$ are more similar to resp. CM2 and CM3, and moreover, we will need the bounded initialization later on anyway.

$\mathrm{CF1}_\rho$		$a(r) \mid b(r)$	$=$	$\gamma(r)$
$\mathrm{CF2}_\rho$	$r \neq t$	$a(r) \mid b(t)$	$=$	$\delta(min(r,t))$
CM1		$p \parallel q$	$=$	$p \mathbin{\|\!\_} q + q \mathbin{\|\!\_} p + p \mid q$
$\mathrm{CM2}_\rho$		$a(r) \mathbin{\|\!\_} p$	$=$	$(a(r) \gg U(p)) \cdot p$
$\mathrm{CM3}_\rho$		$(a(r) \cdot p) \mathbin{\|\!\_} q$	$=$	$(a(r) \gg U(q)) \cdot (p \parallel q)$
CM4		$(p_1 + p_2) \mathbin{\|\!\_} q$	$=$	$p_1 \mathbin{\|\!\_} q + p_2 \mathbin{\|\!\_} q$
$\mathrm{CM5}_\rho$		$(a(r) \cdot p) \mid b(t)$	$=$	$(a(r) \mid b(t)) \cdot p$
$\mathrm{CM6}_\rho$		$a(r) \mid (b(t) \cdot p)$	$=$	$(a(r) \mid b(t)) \cdot p$
$\mathrm{CM7}_\rho$		$(a(r) \cdot p) \mid (b(t) \cdot q)$	$=$	$(a(r) \mid b(t)) \cdot (p \parallel q)$
CM8		$(p_1 + p_2) \mid q$	$=$	$p_1 \mid q + p_2 \mid q$
CM9		$p \mid (q_1 + q_2)$	$=$	$p \mid q_1 + p \mid q_2$
$\mathrm{D1}_\rho$	$a \notin H$	$\partial_H(a(r))$	$=$	$a(r)$
$\mathrm{D2}_\rho$	$a \in H$	$\partial_H(a(r))$	$=$	$\delta(r)$
D3		$\partial_H(p + q)$	$=$	$\partial_H(p) + \partial_H(q)$
D4		$\partial_H(p \cdot q)$	$=$	$\partial_H(p) \cdot \partial_H(q)$
$\mathrm{RT5}^a_\rho$	$r < t$	$a(r) \gg t$	$=$	$a(r)$
$\mathrm{RT5}^b_\rho$	$r \geq t$	$a(r) \gg t$	$=$	$\delta(t)$
$\mathrm{RT6}_\rho$		$(p + q) \gg t$	$=$	$p \gg t + q \gg t$
$\mathrm{RT7}_\rho$		$(p \cdot q) \gg t$	$=$	$(p \gg t) \cdot q$

$(a, b \in A_\delta,\ r, t \in \mathit{Time},\ H \subseteq A)$

Table 3.3: The axiom system ACPρ

$$\frac{\begin{array}{c}<p,t> \xrightarrow{a(r)} <p',r> \\ <q,t> \xrightarrow{\iota(r)} <q',r>\end{array}}{<p\|q,t> \xrightarrow{a(r)} <p'\|q',r>},$$

where $< q,t > \xrightarrow{\iota(r)} < q',r >$ implies $q \equiv q'$, is reformulated in the term semantics by

$$\frac{p \xrightarrow{a(r)} p' \quad U_r(q)}{p\|q \xrightarrow{a(r)} p'\|q}$$

$$\boxed{\frac{U_t(p) \quad U_t(q)}{U_t(p\|q),\ U_t(p \mathbin{\lfloor\!\lfloor} q),\ U_t(p|q)} \qquad \frac{U_t(p)}{U_t(\partial_H(p))} \qquad \frac{U_t(p) \quad t<r}{U_t(p \gg r)}}$$

$(t, r \in \mathit{Time})$

Table 3.4: Rules for $U_t(p)$

Lemma 3.5.1 (Correspondence between $\leftrightarrows$ and $\leftrightarrows^{\iota}$)

$$p,q \in T(\text{ACP}\rho) \qquad p \leftrightarrows q \quad \Longleftrightarrow \quad p \leftrightarrows^{\iota} q$$

Proof. Omitted. □

We have

Theorem 3.5.2 ($\leftrightarrows$ is a congruence over ACPρ**)**

Proof. The set of action rules is in the *path* format [BV93]. □

The following theorem can be proven by checking it for each axiom separately.

Theorem 3.5.3 (Soundness) $p,q \in T(\text{ACP}\rho)$

$$\text{ACP}\rho \vdash p = q \quad \Longrightarrow \quad p \leftrightarrows q$$

Proof. Omitted. □

$$\frac{p \xrightarrow{a(r)} p' \quad U_r(q)}{p\|q \xrightarrow{a(r)} p'\|(r \gg q)\,, \quad q\|p \xrightarrow{a(r)} (r \gg q)\|p'\,, \quad p⌊\!⌊ q \xrightarrow{a(r)} p'\|(r \gg q)}$$

$$\frac{p \xrightarrow{a(r)} \surd \quad U_r(q)}{p\|q \xrightarrow{a(r)} r \gg q\,, \quad q\|p \xrightarrow{a(r)} r \gg q\,, \quad p⌊\!⌊ q \xrightarrow{a(r)} r \gg q}$$

$$\frac{p \xrightarrow{a(r)} p' \quad q \xrightarrow{b(r)} q' \quad \gamma(a,b)=c}{p\|q \xrightarrow{c(r)} p'\|q'\,, \quad p|q \xrightarrow{c(r)} p'\|q'} \qquad \frac{p \xrightarrow{a(r)} \surd \quad q \xrightarrow{b(r)} \surd \quad \gamma(a,b)=c}{p\|q \xrightarrow{c(r)} \surd\,, \quad p|q \xrightarrow{c(r)} \surd}$$

$$\frac{p \xrightarrow{a(r)} \surd \quad q \xrightarrow{b(r)} q' \quad \gamma(a,b)=c}{p\|q \xrightarrow{c(r)} q'\,, \quad q\|p \xrightarrow{c(r)} q'\,, \quad p|q \xrightarrow{c(r)} q'\,, \quad q|p \xrightarrow{c(r)} q'}$$

$$\frac{p \xrightarrow{a(r)} \surd \quad a \notin H}{\partial_H(p) \xrightarrow{a(r)} \surd} \qquad \frac{p \xrightarrow{a(r)} p' \quad a \notin H}{\partial_H(p) \xrightarrow{a(r)} \partial_H(p')}$$

$$\frac{r < t \quad p \xrightarrow{a(r)} \surd}{p \gg t \xrightarrow{a(r)} \surd} \qquad \frac{r < t \quad p \xrightarrow{a(r)} p'}{p \gg t \xrightarrow{a(r)} p'}$$

$(a,b,c \in A,\ r,t \in \mathit{Time},\ H \subseteq A)$

Table 3.5: Action rules for ACPρ

3.6 Elimination and Completeness

We can show that every process term can be reduced to a prefix normal form.

Theorem 3.6.1 (Elimination Theorem for ACPρ**)**
For each term $p \in T(\mathrm{ACP}\rho)$ there is a prefix normal form p' such that ACP$\rho \vdash p = p'$

Proof. The proof is similar to that of Theorem 1.3.1. So, first we show for any two prefix normal forms z, z' and $\Box\{\|, ⌊\!⌊, |\}$, that there is a prefix normal form u such that ACP$\rho \vdash z\Box z' = u$. This is proven by induction on $depth(z + z', \Box)$. For the details of this induction we refer to the proof of 1.3.1. We have to adapt some rules, some of the new versions are given below

$$\begin{array}{lcll}
a(r) \mathbin{⌊\!⌊} z' & = & a(r) \cdot z' & \text{if } r < U(z') \\
a(r) \mathbin{⌊\!⌊} z' & = & \delta(U(z')) & \text{if } r \geq U(z')
\end{array}$$

$$\begin{array}{lcll}
a(r)|b(r) & = & \gamma(r) & \\
a(r)|b(t) & = & \delta(min(r,t)) & \text{if } r \neq t
\end{array}$$

The other rules for $⌊\!⌊$ and $|$ are adapted analogously. The rule for $p||q$ remains.

Finally we give the rules for $p \gg t$, where p is a prefix normal form:

$$\begin{array}{lcll}
a(r) \gg t & = & a(r) & \text{if } r < t \\
a(r) \gg t & = & \delta(t) & \text{if } r \geq t \\
(z_0 + z_1) \gg t & = & z_0 \gg t + z_1 \gg t & \\
(a(r) \cdot z_0) \gg t & = & a(r) \cdot z_0 & \text{if } r < t \\
(a(r) \cdot z_0) \gg t & = & \delta(t) & \text{if } r \geq t
\end{array}$$

The general case follows by induction to the number of occurrences of ACPρ operators. □

And we have obtained that ACPρ axiomatizes $\leftrightarrows$ completely.

Theorem 3.6.2 $\forall p, q \in T(\text{ACP}\rho)$ $\quad p \leftrightarrows q \implies \text{ACP}\rho \vdash p = q$

Proof. Suppose that $p \leftrightarrows q$. According to Theorem 3.6.1 there are prefix normal forms p', q' such that $\text{ACP}\rho \vdash p = p'$ and $\text{ACP}\rho \vdash q = q'$. Then by the soundness of ACPρ w.r.t. $\leftrightarrows$ and by the transitivity of $\leftrightarrows$ we obtain $p' \leftrightarrows p \leftrightarrows q \leftrightarrows q'$, and since $p', q' \in T(\text{BPA}\rho\delta)$, for which we have already proven the completeness, we get $\text{BPA}\rho\delta \vdash p' = q'$. Hence, $\text{ACP}\rho \vdash p = q$. □

4

BPA with Prefixed Integration

4.1 Introduction

In Chapter 2 we have studied BPA$\rho\delta$, in which all atomic actions are decorated with a fixed time stamp. These time stamped processes do not allow us to express processes that can execute actions within a certain time interval. Therefore, we extend BPA$\rho\delta$ with the *integral construct*, which is the alternative composition over a continuum of alternatives, it is introduced in real time process algebra by Baeten & Bergstra [BB91]. They have process terms like $\int_{v\in S} p$, in which the free occurences of the time variable v in p become bound, and where S is an arbitrary subset of the reals. The process that can execute an action a in the interval $[1,2]$ is expressed by the process term

$$\int_{v\in[1,2]} a(v)$$

In this thesis we take a more restrictive view on integration than in [BB91], called *prefixed* integration. We require that every action has as time stamp a time variable directly preceded by the binding integral. Furthermore, we do not have arbitrary subsets of the reals, but subsets that can be described by boolean expressions over time variables. E.g. we allow the following term

$$\int_{1<v\wedge v<2}(a(v)\cdot\int_{v+1\leq w\wedge w\leq v+2} b(w))$$

which is also denoted by

$$\int_{v\in\langle 1,2\rangle}(a(v)\cdot\int_{w\in[v+1,v+2]} b(w)).$$

But we do not allow terms like

$$\int_{v>1}(\int_{w>v+1} a(w)) \text{ or, } \int_{v>1}(a(2)\cdot b(v)) \text{ or, } \quad \int_{v \text{ is prime}} a(v)$$

The restriction to prefixed integration may seem a severe one. But we have not yet encountered a realistic process for which prefixed integration was too restrictive. In Chapter 12 on related work we show that all known other timed process algebras fall within prefixed integration as well.

We introduce the notions of bounds, conditions and substitutions. A bound is a linear expression over time variables, a condition is boolean expression over bounds, and a substitution is a function that assigns bounds to time variables. For example, $2v + 1 > 3w$ is a condition, that is validated by the substitution $[2/v][1/w]$. We have process terms $\int_\alpha (a(v) \cdot p)$, where α is a condition. The construct $\int_\alpha$ binds the occurrences of the time variable v in p. This gives us the notion of free and bound variables. In Section 4.2 we discuss the time domain in detail and we define the syntax and interpretation of bounds and conditions.

In Section 4.3 we define the syntax for process terms with prefixed integration in detail.

In Section 4.4 we give an operational semantics. First we give action rules for terms without free occurrences of time variables. For α with $var(\alpha) \subseteq \{v\}$ there is a transition $\int_\alpha a(v) \cdot p \xrightarrow{a(r)} r \gg p[r/v]$ whenever α is validated by the substitution $[r/v]$ (that assigns r to v). In this way we obtain bisimulation equivalence for terms without free time variables, we define bisimulation equivalence for terms with free time variables indirectly by considering all possible substitutions. We give also action rules for terms with free occurrences of time variables, since these action rules are in the path format of Baeten and Verhoef bisimulation equivalence is a congruence.

In Section 4.5 we give the axiom system BPA$\rho\delta$I, and we discuss substitution and α-conversion in detail.

Finally, in Section 4.6 we prove that BPA$\rho\delta$I axiomatizes completely bisimulation equivalence for terms with free time variables. To obtain this result we generalize the definition of a prefix normal form and we prove that any term can be reduced to such a prefix normal form. Then we construct for each two terms p and q, possibly containing free time variables, a condition that characterizes for which substitutions p and q bisimulate.

4.2 The Time Domain, Bounds and Conditions

4.2.1 The Time Domain

In Section 2.2 we have introduced our time domain *Time*. Here, we introduce several operators, by which more complex time expressions can be constructed. We introduce the binary operators $+$ and $\cdot$, which will have their usual meaning. Furthermore, we have the unary operators $-$ and $^{-1}$; $-t$ is the opposite of t, i.e., $t + (-t) = 0$, and t^{-1} is the inverse of t, i.e., $t \cdot t^{-1} = 1$.

So, from now on we consider *Time* as a collection of constants, at least containing 0 and 1. Let $\mathcal{S}$ be the *signature* $\{. + . , . \cdot . , -. , .^{-1} , \mathit{Time}\}$. We denote the set of terms over $\mathcal{S}$ by $T(\mathcal{S})$.

On $T(\mathcal{S})$ we assume a total ordering $\preceq$, that is a transitive and reflexive relation that relates every two $t_0, t_1 \in$ *Time*. Furthermore, we assume that $\preceq$ is preserved by addition by $t \in$ *Time*, and preserved by multiplication by positive $t \in$ *Time*.

For technical reasons we split $\preceq$ in $=$ and $<$. So, we assume that $t_0 = t_1$ iff $t_0 \preceq t_1$ and $t_1 \preceq t_0$, and that $t_0 < t_1$ iff $t_0 \preceq t_1$ and $t_1 \not\preceq t_0$.

Let FLD be the theory of *fields* [CK90], as given in Table 4.1. We assume that $=$ satisfies the axioms of FLD.

For convenience, we assume that for every $t \in T(\mathcal{S})$ there is a constant $c_t \in$ *Time* such that $t = c_t$, and that for any two $t_0, t_1 \in$ *Time* we have $t_0 = t_1$.

$$
\begin{array}{llcl}
 & x + y & = & y + x \\
 & (x_0 + x_1) + y & = & x_0 + (x_1 + y) \\
 & x + 0 & = & x \\
 & x + (-x) & = & 0 \\
 & x \cdot (y_0 + y_1) & = & x \cdot y_0 + x \cdot y_1 \\
 & x_0 \cdot y + x_1 \cdot y & = & (x_0 + x_1) \cdot y \\
 & x_0 \cdot (x_1 \cdot y) & = & (x_0 \cdot x_1) \cdot y \\
 & x \cdot y & = & y \cdot x \\
 & 1 \cdot x & = & x \\
 & 0 \cdot x & = & 0 \\
x \neq 0 & x \cdot x^{-1} & = & 1 \\
 & 0^{-1} & = & 1
\end{array}
$$

Table 4.1: FLD, the axioms of a field

4.2.2 Bounds

TVar denotes an infinite, countable set of *time variables*. The set *Bound* of *bounds*, with typical element b, is defined by the following BNF sentence, where $t \in T(\mathcal{S})$ and $v \in$ *TVar*.

$$b \ ::= \ t \ \mid \ v \ \mid \ b_1 + b_2 \ \mid \ t \cdot b$$

The set of variables in a bound b is denoted by $var(b)$. If $var(b) = \emptyset$, then b is a *time closed* bound, otherwise it is a *time open* bound.

4.2.3 Substitutions

By Σ we denote the set of *substitutions*, that are mappings from *TVar* to *Bound*. A typical substitution is denoted by σ. We have a subset Σ^{cl} of (time closed) substitutions:

$$\Sigma^{cl} = \{\sigma \mid \forall v \in \mathit{TVar} : \sigma(v) \in \mathit{Time}\}$$

$\sigma(b)$ denotes the bound that results from substituting $\sigma(v)$ for each occurrence of v in b, for all $v \in var(b)$.

4.2.4 The syntax of conditions

A condition is a boolean expression over time variables; the atomic conditions are of the form $b < b'$ and $b = b'$ for $b, b' \in \mathit{Bound}$. The set of conditions is denoted by *Cond.*

$$\alpha ::= tt \mid f\!f \mid b_1 < b_2 \mid b_1 = b_2 \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \neg\alpha$$

We denote the set of time variables of α by $var(\alpha)$.

4.2.5 The interpretation of a condition

$\models tt$	$\dfrac{t_0 \preceq t_1 \quad t_1 \preceq t_0}{\models t_0 = t_1}$	$\dfrac{t_0 \preceq t_1 \quad t_0 \not\preceq t_1}{\models t_0 < t_1}$
$\dfrac{\models \alpha}{\models \alpha \vee \beta, \quad \models \beta \vee \alpha}$	$\dfrac{\models \alpha \quad \models \beta}{\models \alpha \wedge \beta}$	$\dfrac{\not\models \alpha}{\models \neg(\alpha)}$
$\models \sigma(tt)$	$\dfrac{\models \sigma(b_0) = \sigma(b_1)}{\models \sigma(b_0 = b_1)}$	$\dfrac{\models \sigma(b_0) < \sigma(b_1)}{\models \sigma(b_0 < b_1)}$
$\dfrac{\models \sigma(\alpha) \vee \sigma(\beta)}{\models \sigma(\alpha \vee \beta)}$	$\dfrac{\models \sigma(\alpha) \wedge \sigma(\beta)}{\models \sigma(\alpha \wedge \beta)}$	$\dfrac{\models \neg(\sigma(\alpha))}{\models \sigma(\neg(\alpha))}$

Table 4.2: Rules for validating time closed conditions

In Table 4.2 we define a predicate $\models$ on time closed conditions. For each $\sigma \in \Sigma^{cl}$ and α we have $var(\sigma(\alpha)) = \emptyset$, and thus either $\models \sigma(\alpha)$ or $\not\models \sigma(\alpha)$. We denote the subset of substitutions in Σ^{cl} that validate α by $[\alpha]$.

$$[\alpha] := \{\sigma \in \Sigma^{cl} \mid \models \sigma(\alpha)\},$$

Moreover, for a time open α we take $\models \alpha$ if $[\alpha] = \Sigma^{cl}$. We take $\models \alpha = \beta$ if $[\alpha] = [\beta]$.

In Appendix A we show that $b = b'$ iff $\models \sigma(b = b')$. This is shown by constructing a normal form for each bound, that is a bound of the form:

$$r_1 \cdot v_1 + \ldots + r_n \cdot v_n + t \quad (n \geq 0),$$

where $r_i \in \mathit{Time}\backslash 0$ and all variables are different.

In that appendix we give also an axiom system CA for reasoning with conditions that contain time variables. We have the following proposition:

Proposition 4.2.1 (Soundness and Completeness of CA)

$$\models \alpha = \beta \quad \Longleftrightarrow \quad \mathrm{CA} \vdash \alpha = \beta$$

Proof. See Appendix A. □

4.2.6 Intervals and conditions

We assume two symbols, $-\infty$ (minus infinity) and ∞ (infinity) not in *Bound*. We denote $\mathit{Bound} \cup \{-\infty, \infty\}$ by $\mathit{Bound}_{-\infty,\infty}$. We have some notations that concern ∞ and $-\infty$.

$$\begin{aligned} b < \infty &\overset{not}{=} tt \\ \infty < b &\overset{not}{=} ff \end{aligned}$$

Similarly we have $-\infty < b$, $b < -\infty$, $b = \infty$, and $b = -\infty$ as notations of either tt or ff. The expression $b \leq b'$ abbreviates $b < b' \vee b = b'$, and for $b_0, b_1 \in \mathit{Bound}_{-\infty,\infty}$ we have

$$\begin{aligned} v \in \langle b_0, b_1 \rangle &\overset{abb}{=} b_0 < v \wedge v < b_1 \\ v \in [b_0, b_1 \rangle &\overset{abb}{=} b_0 \leq v \wedge v < b_1 \end{aligned}$$

and similarly we have $v \in \langle b_0, b_1]$ and $v \in [b_0, b_1]$ as abbreviations for conditions.

Thus $v \in \langle b, \infty \rangle$ abbreviates $b < v \wedge v < \infty$, which in turn abbreviates the condition $b < v \wedge tt$, which can be reduced to the condition $b < v$.

4.2.7 Partitions and refinements

Two conditions α_1 and α_2 are *non overlapping* if $[\alpha_1] \cap [\alpha_2] = \emptyset$.

A finite collection of conditions $\{\alpha_1, ..., \alpha_n\}$ is called non-overlapping if each pair in the collection is non-overlapping. A collection of conditions $\{\beta_j\}$ is called a *refinement* of a collection of conditions $\{\alpha_i\}$ if it is non-overlapping, $\cup_j[\beta_j] = \cup_i[\alpha_i]$ and for each j there is an i such that $[\beta_j] \subseteq [\alpha_i]$. A collection of conditions $\{\beta_j\}$ is called a *partition* if it refines $\{tt\}$.

The following Refinement Lemma will play a crucial role in the main theorems, like the decidability theorem for BPA$\rho\delta$I, as will be clear in the sequel. Note that this lemma depends heavily on the syntax of the bounds, if we would allow bounds like v^2, then we do not have this lemma any more.

Lemma 4.2.2 (Refinement Lemma) *Fix a time variable v. For each condition α there is an equivalent condition of the form $\vee_j(\beta_j \wedge v \in V_j)$, where $var(\beta_j) \cup var(V_j) \subseteq var(\alpha)\backslash\{v\}$ for all j.*

Proof. See Appendix A. □

This lemma is our motivation for having $=$ and $<$ in our language for conditions, instead of $\preceq$. If we have only $\preceq$ then an expression like $v \in V$ may abbreviate a condition with negation, which we do not prefer.

4.2.8 Some more abbreviations

We introduce some more abbreviations for conditions, where $b, b_0, b_1, b_2 \in Bound_{-\infty,\infty}$.

$$\begin{array}{rcll}
\alpha \Rightarrow \beta & \overset{abb}{=} & \neg(\alpha) \vee \beta & \\
\langle\!\!| b_0, b_1 |\!\!\rangle = \emptyset & \overset{abb}{=} & b_1 \leq b_0 & \text{if } \langle\!\!| = (\text{ or } |\!\!\rangle =) \\
 & \overset{abb}{=} & b_1 < b_0 & \text{if } \langle\!\!| = [\text{ and } |\!\!\rangle =] \\
\langle\!\!| b_0, b_1 |\!\!\rangle \neq \emptyset & \overset{abb}{=} & \neg(\langle\!\!| b_0, b_1 |\!\!\rangle = \emptyset) & \\
b < \sup(\langle\!\!| b_0, b_1 |\!\!\rangle) & \overset{abb}{=} & \langle\!\!| b_0, b_1 |\!\!\rangle \neq \emptyset \wedge b < b_1 & \\
b > \sup(\langle\!\!| b_0, b_1 |\!\!\rangle) & \overset{abb}{=} & \langle\!\!| b_0, b_1 |\!\!\rangle = \emptyset \vee b_1 < b &
\end{array}$$

Furthermore, $V \sim V'$ abbreviates the condition that V and V' are overlapping or adjacent intervals, such that $V \cup V'$ is an interval as well.

4.3 Terms with conditions

4.3.1 The ultimate delay

In Chapter 2 the ultimate delay of p, denoted by $U(p)$, is a time stamp, that corresponds with the upperbound of points in time to which p can idle. For example $U(a(2) \cdot c(4) + b(3)) = \max(2, 3) = 3$.

For terms in $T^{cl}(\text{BPA}\rho\delta\text{I})$ the ultimate delay can be a time stamp as well, for example $\int_{v \in \langle 1,2]} a(v) \cdot c(4) + \int_{w \in \langle 2,3\rangle} b(w) = \max(2, 3) = 3$. For terms in $T(\text{BPA}\rho\delta\text{I})$, that may contain free time variables, the most obvious generalization seems to be to define the ultimate delay as a bound. However, it makes only sense to put $U(\int_{v \in \langle b_0,b_1\rangle} P(v)) = b_1$ under the condition that $b_0 < b_1$. Since we do not allow conditions in our bounds, we cannot define $U(\int_{v \in \langle b_0,b_1\rangle} P(v))$ properly as a bound. For similar reasons, we can not define $U(\alpha :\!\rightarrow p)$ as a proper bound.

A way out is to add a bound b as parameter to the ultimate delay, and to identify $U_b(\int_{v \in \langle b_0,b_1\rangle} P(v))$ with the expression $b \leq \sup(\langle b_0, b_1\rangle)$, that abbreviates the condition $\langle b_0, b_1\rangle \neq \emptyset \wedge b < b_1$. If for certain t $U_t(p)$ reduces to tt, then it means that p can idle till t. In other words, we have introduced the predicate $U_t(p)$, that we had already in the term semantics, in the calculus as well.

We extend the set *Cond* to $Cond_U$ by allowing conditions of the form $U_b(p)$ as well, where $b \in Bound$. We take $var(U_b(p)) = var(b) \cup fv(p)$. We have to introduce two rules for validating this new condition. Note that the premise $U_t(p)$ of the rule on the left hand side in Table 4.3 is the predicate as defined in the Tables 2.3 and 4.4.

$$\frac{U_t(p)}{\models U_t(p)} \qquad \frac{\models U_{\sigma(b)}(\sigma(p))}{\models \sigma(U_b(p))}$$

Table 4.3: Additional rules for time closed $U_b(p)$

4.3.2 The syntax for process terms

Let $\alpha \in \mathit{Cond}_U$, $a \in A_\delta$ and $b \in \mathit{Bound}$. The set $T(\mathrm{BPA}\rho\delta\mathrm{I})$ of (time open) process terms with conditions is defined by

$$p \ ::= \alpha :\!\longrightarrow p \ \mid \ \int_\alpha a(v) \ \mid \ \int_\alpha (a(v)\cdot p) \ \mid \ p+p \ \mid \ p\cdot p \ \mid \ b \gg p \ \mid \ \sigma(p)$$

We abbreviate $\int_{\mathit{ff}}\delta(v)$ by δ. In some cases we write $\int_\alpha(a(v))\cdot p$ for $\int_\alpha a(v)\cdot p$, in order to stress that the term p is not in the scope of the integral $\int_\alpha$.

4.3.3 Free time variables

We define inductively the collection $fv(p)$ of time variables appearing in a process term p that are not bound by an integral sign in p, the so-called *free* variables:

$$\begin{array}{lcl}
fv(\int_\alpha a(v)) & = & var(\alpha) - \{v\} \\
fv(\int_\alpha (a(v)\cdot p)) & = & (var(\alpha) \cup fv(p)) - \{v\} \\
fv(p+q) & = & fv(p) \cup fv(q) \\
fv(p\cdot q) & = & fv(p) \cup fv(q) \\
fv(\sigma(p)) & = & \{w \mid \exists v \ v \in fv(p) \text{ and } w \in var(\sigma(v))\}
\end{array}$$

A process term p with $fv(p) = \emptyset$ is called a *time-closed* process term. We define

$$T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I}) \quad = \quad \{\ p \in T(\mathrm{BPA}\rho\delta\mathrm{I}) \mid fv(p) = \emptyset\ \}$$

Moreover, a term p with $fv(p) \neq \emptyset$ is a *time open* term.

4.4 An Operational Semantics for Time Open Terms

4.4.1 A generalization of bisimulation equivalence

We provide any time closed process term in $T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I})$ with a transition system. Hence, we add some new action rules, see Table 4.4, that are applicable for time closed terms only.

The action rules for the $+$, $\cdot$ and $\gg$ can be found in Table 2.3, in which case p, q are supposed to range over $T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I})$. The rules for substitution, that is for $\sigma(p)$ where $fv(\sigma(p)) = \emptyset$, are given in Table 4.6.

$\dfrac{\models \alpha[r/v]}{\int_\alpha (a(v)\cdot p) \xrightarrow{a(r)} r \gg p[r/v]}$	$\dfrac{\models \alpha[r/v]}{\int_\alpha a(v) \xrightarrow{a(r)} \surd}$	$\dfrac{\models \alpha[r/v] \quad t < r}{U_t(\int_\alpha P(v))}$
$\dfrac{\models \alpha \quad p \xrightarrow{a(r)} p'}{\alpha :\!\rightarrow p \xrightarrow{a(r)} p'}$	$\dfrac{\models \alpha \quad p \xrightarrow{a(r)} \surd}{\alpha :\!\rightarrow p \xrightarrow{a(r)} \surd}$	$\dfrac{\models \alpha \quad U_t(p)}{U_t(\alpha :\!\rightarrow p)}$

$(r, t \in \mathit{Time})$

Table 4.4: Additional action rules for time closed $\int_\alpha P(v)$ and $\alpha :\!\rightarrow p$

$\models_\sigma tt$	$\dfrac{\sigma(b_0) \preceq \sigma(b_1) \quad \sigma(b_1) \preceq \sigma(b_0)}{\models_\sigma b_0 = b_1}$
$\dfrac{\sigma(b_0) \preceq \sigma(b_1) \quad \sigma(b_1) \not\preceq \sigma(b_0)}{\models_\sigma b_0 < b_1}$	$\dfrac{\models_\sigma \alpha}{\models_\sigma \alpha \vee \beta, \quad \models_\sigma \beta \vee \alpha}$
$\dfrac{\models_\sigma \alpha \quad \models_\sigma \beta}{\models_\sigma \alpha \wedge \beta}$	$\dfrac{\not\models_\sigma \alpha}{\models_\sigma \neg(\alpha)}$
$\dfrac{\models_{\sigma\circ\mu} \alpha}{\models_\sigma \mu(\alpha)}$	$\dfrac{U^\sigma_{\sigma(b)}(p)}{\models_\sigma U_b(p)}$

$b, b_0, b_1 \in \mathit{Bound}, \ \sigma \in \Sigma^{cl}$

Table 4.5: Rules for validating a condition in Σ-semantics

The action rules for the idle semantics are analogous, they are left to the reader. We extend the definition of $\underline{\leftrightarrow}$ to terms of $T(\mathrm{BPA}\rho\delta\mathrm{I})$ by parameterizing the equivalence with a condition, see Definition 4.4.1.

Definition 4.4.1 (α-Bisimulation equivalence)
$p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I}) \qquad p \underline{\leftrightarrow}^\alpha q$ iff $\forall \sigma \in [\alpha] : \ \sigma(p) \underline{\leftrightarrow} \sigma(q)$

We abbreviate $p \underline{\leftrightarrow}^{tt} q$ by $p \underline{\leftrightarrow} q$.

$$\frac{\models \sigma[r/v](\alpha)}{\sigma(\textstyle\int_\alpha (a(v)\cdot p)) \xrightarrow{a(r)} r \gg \sigma[r/v](p)} \qquad \frac{\models \sigma[r/v](\alpha)}{\sigma(\textstyle\int_\alpha a(v)) \xrightarrow{a(r)} \surd}$$

$$\frac{\sigma(p)+\sigma(q) \xrightarrow{a(r)} p'}{\sigma(p+q) \xrightarrow{a(r)} p'} \qquad \frac{\sigma(p)+\sigma(q) \xrightarrow{a(r)} \surd}{\sigma(p+q) \xrightarrow{a(r)} \surd}$$

$$\frac{\sigma(p)\cdot\sigma(q) \xrightarrow{a(r)} p'}{\sigma(p\cdot q) \xrightarrow{a(r)} p'}$$

$$\frac{\sigma(b) \gg \sigma(p) \xrightarrow{a(r)} p'}{\sigma(b \gg p) \xrightarrow{a(r)} p'} \qquad \frac{\sigma(b) \gg \sigma(p) \xrightarrow{a(r)} \surd}{\sigma(b \gg p) \xrightarrow{a(r)} \surd}$$

$$\frac{\sigma(\alpha) :\!\rightarrow \sigma(p) \xrightarrow{a(r)} p'}{\sigma(\alpha :\!\rightarrow p) \xrightarrow{a(r)} p'} \qquad \frac{\sigma(\alpha) :\!\rightarrow \sigma(p) \xrightarrow{a(r)} \surd}{\sigma(\alpha :\!\rightarrow p) \xrightarrow{a(r)} \surd}$$

$$\frac{\sigma\circ\mu(p) \xrightarrow{a(r)} p'}{\sigma(\mu(p) \xrightarrow{a(r)} p'} \qquad \frac{\sigma\circ\mu(p) \xrightarrow{a(r)} \surd}{\sigma(\mu(p) \xrightarrow{a(r)} \surd}$$

$$\frac{\models \sigma[r/v](\alpha) \quad \models t<r}{U_t(\sigma(\textstyle\int_\alpha P(v)))} \qquad \frac{U_t(\sigma(p)+\sigma(q))}{U_t(\sigma(p+q))}$$

$$\frac{U_t(\sigma(p)\cdot\sigma(q))}{U_t(\sigma(p\cdot q))} \qquad \frac{U_t(\sigma(b) \gg \sigma(p))}{U_t(\sigma(b \gg p))}$$

$$\frac{U_t(\sigma(\alpha) :\!\rightarrow \sigma(p))}{U_t(\sigma(\alpha :\!\rightarrow p))} \qquad \frac{U_t(\sigma\circ\mu(p))}{U_t(\sigma(\mu(p)))}$$

$\sigma, \mu \in \Sigma$

Table 4.6: Action rules for substitution in term semantics

Example 4.4.2

$$\begin{array}{lll}
\int_{v\in[b,b]}(a(v)\cdot c(w+1)) & \underline{\leftrightarrow}^{b=w} & \int_{v\in[b,b]}(a(v)\cdot c(v+1)) \\
\int_{v\in\langle b,b'\rangle}(a(v)\cdot \int_{w\in\langle v,e\rangle} c(w)) & \underline{\leftrightarrow}^{b<e<b'} & \\
\multicolumn{3}{r}{\int_{v\in\langle b,e\rangle}(a(v)\cdot \int_{w\in\langle v,e\rangle} c(w)) + \int_{v\in[e,b'\rangle} a(v)\cdot\delta} \\
\int_{v\in[b,b']}(a(v)\cdot \int_{w\in\langle v,b'\rangle} c(w)) & \underline{\leftrightarrow} & \\
\multicolumn{3}{r}{\int_{v\in[b,b'\rangle}(a(v)\cdot \int_{w\in\langle v,b'\rangle} c(w)) + a(b')\cdot\delta}
\end{array}$$

4.4.2 Bisimulation equivalence is a congruence

We do not obtain immediately that $\underline{\leftrightarrow}$ is a congruence over $T(\mathrm{BPA}\rho\delta\mathrm{I})$ as the action rules for substitution, see Table 4.6, are not in the path format of Baeten and Verhoef. Moreover, the action rules define $\underline{\leftrightarrow}$ only on $T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I})$, and $\underline{\leftrightarrow}$ over $T(\mathrm{BPA}\rho\delta\mathrm{I})$ is defined indirectly, see Definition 4.4.1.

In this section we give action rules in the path format for terms in $T(\mathrm{BPA}\rho\delta\mathrm{I})$, that may contain free occurrences of time variables. Each transition is labelled with a timed action *and* a substitution $\sigma \in \Sigma^{cl}$ that determines the values for the free time variables in the target state. This semantics is called Σ-semantics and its action rules are given in Table 4.7. We have to redefine the predicate $\models$ as well, in Table 4.5 we define a predicate $\models_\sigma$ for arbitrary conditions.

The resulting bisimulation equivalence is denoted by $\underline{\leftrightarrow}^\Sigma$.

Before we can prove that $\underline{\leftrightarrow}^\Sigma$ coincides with $\underline{\leftrightarrow}$ we need some properties of both equivalences.

Proposition 4.4.3 $p \in T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I}),\ \sigma \in \Sigma$

$$\sigma(p) \underline{\leftrightarrow} p$$

Proof. By induction to the size of p. □

Corollary 4.4.4 $p, q \in T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I}),\ \sigma \in \Sigma$

$$p \underline{\leftrightarrow} q \quad \Longleftrightarrow \quad \sigma(p) \underline{\leftrightarrow} \sigma(q)$$

Proposition 4.4.5 $p \in T(\mathrm{BPA}\rho\delta\mathrm{I}),\ \sigma \in \Sigma^{cl}$

$$\begin{array}{llcl}
1 & \sigma(p) \xrightarrow{a(r)} p' & \Longleftrightarrow & p \xrightarrow{a(r)}_\sigma p' \\
2 & \sigma(p) \xrightarrow{a(r)} \surd & \Longleftrightarrow & p \xrightarrow{a(r)}_\sigma \surd \\
3 & U_t(\sigma(p)) & \Longleftrightarrow & U_t^\sigma(p)
\end{array}$$

Proof. By induction on the length of the derivation. □

Lemma 4.4.6 *For* $p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$ *we have* $p \underline{\leftrightarrow}^{tt} q \iff p \underline{\leftrightarrow}^\Sigma q$

Proof.

$$\frac{\models_{\sigma[r/v]}(\alpha)}{\int_\alpha a(v)\cdot p \xrightarrow{a(r)}_\sigma r \gg \sigma[r/v](p)} \qquad \frac{\models_{\sigma[r/v]}(\alpha)}{\int_\alpha a(v) \xrightarrow{a(r)}_\sigma \surd}$$

$$\frac{p \xrightarrow{a(r)}_\sigma p'}{p+q \xrightarrow{a(r)}_\sigma p'} \qquad \frac{p \xrightarrow{a(r)}_\sigma \surd}{p+q \xrightarrow{a(r)}_\sigma \surd}$$

$$\frac{p \xrightarrow{a(r)}_\sigma p'}{q+p \xrightarrow{a(r)}_\sigma p'} \qquad \frac{p \xrightarrow{a(r)}_\sigma \surd}{q+p \xrightarrow{a(r)}_\sigma \surd}$$

$$\frac{p \xrightarrow{a(r)}_\sigma p'}{p\cdot q \xrightarrow{a(r)}_\sigma p'\cdot\sigma(q)} \qquad \frac{p \xrightarrow{a(r)}_\sigma \surd}{p\cdot q \xrightarrow{a(r)}_\sigma r \gg \sigma(q)}$$

$$\frac{p \xrightarrow{a(r)}_\sigma p' \quad \models_\sigma b < r}{b \gg p \xrightarrow{a(r)}_\sigma p'} \qquad \frac{p \xrightarrow{a(r)}_\sigma \surd \quad \models_\sigma b < r}{b \gg p \xrightarrow{a(r)}_\sigma \surd}$$

$$\frac{p \xrightarrow{a(r)}_\sigma p' \quad \models_\sigma \alpha}{\alpha :\rightarrow p \xrightarrow{a(r)}_\sigma p'} \qquad \frac{p \xrightarrow{a(r)}_\sigma \surd \quad \models_\sigma \alpha}{\alpha :\rightarrow p \xrightarrow{a(r)}_\sigma \surd}$$

$$\frac{p \xrightarrow{a(r)}_{\sigma\circ\mu} p'}{\mu(p) \xrightarrow{a(r)}_\sigma p'} \qquad \frac{p \xrightarrow{a(r)}_{\sigma\circ\mu} \surd}{\mu(p) \xrightarrow{a(r)}_\sigma \surd}$$

$$\frac{\models_{\sigma[r/v]}(\alpha) \quad \models_\sigma t < r}{U_t^\sigma(\int_\alpha P(v))} \qquad \frac{U_t^\sigma(p)}{U_t^\sigma(p+q),\ U_t^\sigma(q+p)}$$

$$\frac{U_t^\sigma(p)}{U_t^\sigma(p\cdot q)} \qquad \frac{\models_\sigma t \leq b}{U_t^\sigma(b \gg p)} \qquad \frac{U_t^\sigma(p) \quad \models_\sigma b < t}{U_t^\sigma(b \gg p)}$$

$$\frac{U_t^\sigma(p) \quad \models_\sigma \alpha}{U_t^\sigma(\alpha :\rightarrow p)} \qquad \frac{U_t^{\sigma\circ\mu}(p)}{U_t^\sigma(\mu(p))}$$

$\sigma \in \Sigma^{cl},\ \mu \in \Sigma$

Table 4.7: Action rules for Σ-semantics

- $\Longrightarrow$. Assume $p \leftrightarroweq^{tt} q$, we construct

$$\mathcal{R} = \{ (p', q') \mid p', q' \in T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I})\ p' \leftrightarroweq q' \}$$

and we show that $\mathcal{R} \cup \{(p,q)\} : p \leftrightarroweq^{\Sigma} q$.

First we discuss the pair (p,q). Consider $p \xrightarrow{a(r)}_{\sigma} p'$, then we have to show that there is a q' such that $q \xrightarrow{a(r)}_{\sigma} q'$ and $p'\mathcal{R}q'$. By Proposition 4.4.5, part 1, $\Longleftarrow$, we have $\sigma(p) \xrightarrow{a(r)} p'$. Since we have $\sigma(p) \leftrightarroweq \sigma(q)$, there is a q' such that $\sigma(q) \xrightarrow{a(r)} q'$ and $p' \leftrightarroweq q'$, and thus $p'\mathcal{R}q'$ as well.

The cases $p \xrightarrow{a(r)}_{\sigma} \surd$ and $U_t^{\sigma}(p)$ are left to the reader.

Next, we discuss a pair $(p', q') \in \mathcal{R}$. Consider $p' \xrightarrow{a(r)}_{\sigma} p''$, then by Proposition 4.4.5, part 1, $\Longleftarrow$, we have $\sigma(p') \xrightarrow{a(r)} p''$. Since $(p', q') \in \mathcal{R}$ also $p' \leftrightarroweq q'$, and since p', q' are time closed we have $\sigma(p') \leftrightarroweq \sigma(q')$ and thus there is a q'' such that $\sigma(q') \xrightarrow{a(r)} q''$, from which we obtain that $q' \xrightarrow{a(r)}_{\sigma} q''$ and $p''\mathcal{R}q''$.

The cases $p' \xrightarrow{a(r)}_{\sigma} \surd$ and $U_t^{\sigma}(p')$ are left to the reader.

- $\Longleftarrow$. Assume $p \leftrightarroweq^{\Sigma} q$. Take

$$\mathcal{R} = \{ (p', q') \mid p', q' \in T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I})\ p' \leftrightarroweq^{\Sigma} q' \}$$

and we show that $\mathcal{R} \cup \{(\sigma(p), \sigma(q))\} : \sigma(p) \leftrightarroweq \sigma(q)$

First we discuss the pair $(\sigma(p), \sigma(q))$. Consider $\sigma(p) \xrightarrow{a(r)} p'$, then we have to show that there is a q' such that $\sigma(q) \xrightarrow{a(r)} q'$ and $p'\mathcal{R}q'$. By Proposition 4.4.5, part 1, $\Longrightarrow$, we have $p \xrightarrow{a(r)}_{\sigma} p'$, since $p \leftrightarroweq^{\Sigma} q$ there is a q' such that $q \xrightarrow{a(r)}_{\sigma} q'$ and $p' \leftrightarroweq^{\Sigma} q'$, and thus $p'\mathcal{R}q'$ as well. By Proposition 4.4.5, part 1, $\Longleftarrow$, we obtain $\sigma(q) \xrightarrow{a(r)} q'$ and we are ready.

The cases $\sigma(p) \xrightarrow{a(r)} \surd$ and $U_t(\sigma(p)$ are left to the reader.

Next, we discuss a pair $(p', q') \in \mathcal{R}$. Consider $p' \xrightarrow{a(r)} p''$, then we have to find a q'' such that $q' \xrightarrow{a(r)} q''$ and $(p'', q'') \in \mathcal{R}$. Take an arbitrary $\sigma \in \Sigma^{cl}$. Since p' is time closed we have $p' \leftrightarroweq \sigma(p')$, and thus there is a p''_{σ} such that $\sigma(p') \xrightarrow{a(r)} p''_{\sigma}$ and $p'' \leftrightarroweq p''_{\sigma}$. From $\sigma(p') \xrightarrow{a(r)} p''_{\sigma}$ it follows that $p' \xrightarrow{a(r)}_{\sigma} p''_{\sigma}$, and since $p' \leftrightarroweq^{\Sigma} q'$ there is a q''_{σ} such that $q' \xrightarrow{a(r)}_{\sigma} q''_{\sigma}$ and $p''_{\sigma} \leftrightarroweq^{\Sigma} q''_{\sigma}$. From $q' \xrightarrow{a(r)}_{\sigma} q''_{\sigma}$ it follows that $\sigma(q') \xrightarrow{a(r)} q''_{\sigma}$, and since $q' \leftrightarroweq \sigma(q')$ there must be a q'' such that $q' \xrightarrow{a(r)} q''$ and $q''_{\sigma} \leftrightarroweq q''$. From the previous part of this proof and $p'' \leftrightarroweq p''_{\sigma}$, $q'' \leftrightarroweq q''_{\sigma}$ we obtain $p'' \leftrightarroweq^{\Sigma} p''_{\sigma}$, $q'' \leftrightarroweq^{\Sigma} q''_{\sigma}$. Finally, by $p''_{\sigma} \leftrightarroweq^{\Sigma} q''_{\sigma}$ and the transitivity of $\leftrightarroweq^{\Sigma}$ we obtain $p'' \leftrightarroweq^{\Sigma} q''$, and thus $p''\mathcal{R}q''$ and we are ready.

The cases $\sigma(p') \xrightarrow{a(r)} \surd$ and $U_t(\sigma(p')$ are left to the reader.

□

4.5 Reasoning with Time Open Terms

4.5.1 Substitution and α-conversion

In Table 4.8 we give the axioms for substitution. Consider the process term

$$\int_{v>1}(a(v)\cdot\int_{w>v+1}b(w))$$

and we assume that want to replace v by w, using α-conversion. Obviously, it is not right to obtain

$$\int_{w>1}(a(w)\cdot\int_{w>w+1}b(w))$$

as both occurrences of the variables w in $w > w+1$ are bound by the same integral. So, first we have to substitute a variable w' for the bound variable w, in order to avoid the above clash of bindings.

This renaming of bound variables is forced by the condition on SU2 and SU3. The associated derivation that uses the axioms SU1-6 is given in the following example:

Example 4.5.1

$$\begin{aligned}
&\textstyle\int_{v>1}(a(v)\cdot\int_{w>v+1}b(w))\\
\overset{\mathrm{SU6}}{=}\;&\textstyle\int_{w>1}(a(w)\cdot\int_{w>v+1}b(w)[w/v])\\
\overset{\mathrm{SU5}}{=}\;&\textstyle\int_{w>1}(a(w)\cdot\int_{w'>v+1}b(w')[w/v])\\
\overset{\mathrm{SU2}}{=}\;&\textstyle\int_{w>1}(a(w)\cdot\int_{w'>w+1}b(w'))
\end{aligned}$$

In the literature it is usual to deal with process terms modulo α-conversion and to have the above renaming of bound variables implicit in the notion of substitution, for details we refer to [Sto88]. α-conversion also implies that the objects of study are not expressions, but congruence classes of process terms. Since we do not want to deal with congruence classes, but with concrete process terms, we have decided not to work modulo α-conversion.

4.5.2 The axiom system BPA$\rho\delta$I

The axiom system BPA$\rho\delta$I is given in Table 4.9, the process terms left and right from the =-symbol are arbitrary process terms from $T(\mathrm{BPA}\rho\delta\mathrm{I})$. The axiom A6$_C$ says that all idle behavior from the neighbors of a δ-summand can be subtracted from that δ summand. For example

SU1	$\sigma(p+q)$	$=$	$\sigma(p)+\sigma(q)$
SU2	$w \in fv(\int_\alpha a(v))$		$v \notin var(\sigma(w))$
	$\sigma(\int_\alpha a(v))$	$=$	$\int_{\sigma\backslash_v(\alpha)} a(v)$
SU3	$w \in fv(\int_\alpha (a(v)\cdot p))$		$v \notin var(\sigma(w))$
	$\sigma(\int_\alpha (a(v)\cdot p))$	$=$	$\int_{\sigma\backslash_v(\alpha)} (a(v)\cdot \sigma\backslash_v(p))$
SU4	$\sigma(\sigma'(p))$	$=$	$\sigma\circ\sigma'(p)$
SU5	$w \notin fv(\int_\alpha a(v))$		
	$\int_\alpha a(v)$	$=$	$\int_{\alpha[w/v]} a(w)$
SU6	$w \notin fv(\int_\alpha (a(v)\cdot p))$		
	$\int_\alpha (a(v)\cdot p))$	$=$	$\int_{\alpha[w/v]} (a(w)\cdot p[w/v])$

$(a \in A_\delta,\ \sigma \in \Sigma)$

Table 4.8: Axioms for substitution

$$\begin{aligned}
& \int_{v\in\langle 1,5\rangle} a(v) + \int_{v\in[5,5]} \delta(w) \\
\stackrel{A6_C}{=}\ & \int_{v\in\langle 1,5\rangle} a(v) + \int_{v\in[5,5]\wedge\neg(v\leq 5)} \delta(v) \\
\stackrel{CA}{=}\ & \int_{v\in\langle 1,5\rangle} a(v) + \int_{ff} \delta(v) \\
\stackrel{abbr}{=}\ & \int_{v\in\langle 1,5\rangle} a(v) + \delta \\
\stackrel{A6}{=}\ & \int_{v\in\langle 1,5\rangle} a(v)
\end{aligned}$$

4.5.3 The Lifting Lemma

In the sequel we will need to lift conditions to the top of a process term. Hence, we have following Lemma:

Lemma 4.5.2 (Lifting Lemma) *If $\{\alpha_i \wedge v \in W_i\}$ is a partition and $v \notin var(\alpha_i) \cup var(W_i)$, then*

$$\int_\alpha (a(v)\cdot \Sigma_i\{\alpha_i \wedge v \in W_i :\rightarrow p_i\}) = \Sigma_i \int_{\alpha\wedge\alpha_i\wedge v\in W_i} (a(v)\cdot p_i)$$

Proof. The proof is based on the fact that if $\{\gamma_j\}$ is a partition then

$$\text{BPA}\rho\delta\text{I} \vdash \gamma_j :\rightarrow \sum_{j'}\{\gamma_{j'} :\rightarrow z_{j'}\} \quad = \quad \beta :\rightarrow z_j$$

and we have BPA$\rho\delta$I$\vdash$

A1		$p + q$	$=$	$q + p$
A2		$(p + q) + z$	$=$	$p + (q + z)$
A3$_C$		$\int_\alpha P(v) + \int_\beta P(v)$	$=$	$\int_{\alpha\vee\beta} P(v)$
A4		$(p + q) \cdot z$	$=$	$p \cdot z + q \cdot z$
A5^a_C	$v \notin fv(q)$	$\int_\alpha (a(v)) \cdot q$	$=$	$\int_\alpha (a(v) \cdot q)$
A5^b_C	$v \notin fv(q)$	$\int_\alpha (a(v) \cdot p) \cdot q$	$=$	$\int_\alpha (a(v) \cdot (p \cdot q))$
A6		$p + \delta$	$=$	p
A6$_C$	$v \notin fv(p)$	$p + \int_\alpha \delta(v)$	$=$	$p + \int_{\alpha\wedge\neg(U_v(p))} \delta(v)$
A7$_C$		$\int_\alpha (\delta(v) \cdot p)$	$=$	$\int_\alpha \delta(v)$
RT0$_C$		$\int_{ff} P(v)$	$=$	δ
RT1$_C$		$\int_\alpha (a(v) \cdot p)$	$=$	$\int_\alpha (a(v) \cdot (v \gg p))$
RT2$_C$	$v \notin var(b)$	$b \gg \int_\alpha P(v)$	$=$	$\int_{\alpha\wedge v>b} P(v) + \delta(b)$
RT3$_C$		$b \gg (p + q)$	$=$	$(b \gg p) + (b \gg q)$
C1		$\alpha :\rightarrow (p + q)$	$=$	$\alpha :\rightarrow p + \alpha :\rightarrow q$
C2	$v \notin var(\alpha)$	$\alpha :\rightarrow \int_\beta P(v)$	$=$	$\int_{\alpha\wedge\beta} P(v)$
C3		$\int_\alpha (a(v) \cdot p)$	$=$	$\int_\alpha (a(v) \cdot \alpha :\rightarrow p)$
U1$_C$		$U_b(p + q)$	$=$	$U_b(p) \vee U_b(q)$
U2$_C$		$U_b(\int_{v\in V} P(v))$	$=$	$b < \sup(V)$
U3$_C$		$U_b(\alpha :\rightarrow p)$	$=$	$\alpha \wedge U_b(p)$

($a \in A_\delta$, $b \in Bound$, $P(v)$ is either of the form $a(v)$ or $a(v) \cdot p$)

Table 4.9: An axiom system for BPA$\rho\delta$I

$$\begin{array}{ll} & \int_{\alpha}(a(v) \cdot \Sigma_i\{\alpha_i \wedge v \in W_i :\rightarrow p_i\}) \\ = & \sum_i \int_{\alpha \wedge \alpha_i \wedge v \in V_i}(a(v) \cdot \sum_i\{\alpha_i \wedge v \in W_i :\rightarrow p_i\}) \\ = & \sum_i \int_{\alpha \wedge \alpha_i \wedge v \in W_i}(a(v) \cdot \sum_{i'}\{\alpha_{i'} \wedge v \in W_{i'} :\rightarrow p_{i'}\}) \\ = & \sum_i \int_{\alpha \wedge \alpha_i \wedge v \in W_i}(a(v) \cdot \\ & \quad \{\alpha_i \wedge v \in V \cap W_i \ :\rightarrow \ \sum_{i'}\{\alpha_{i'} \wedge v \in W_{i'} :\rightarrow p_{i'}\}\}) \\ = & \sum_i \int_{\alpha \wedge \alpha_i \wedge v \in W_i}(a(v) \cdot \{\alpha_i \wedge v \in V \cap W_i \ :\rightarrow \ p_i\}) \\ = & \sum_i \int_{\alpha \wedge \alpha_i \wedge v \in W_i}(a(v) \cdot p_i) \end{array}$$

□

4.6 Completeness and Decidability

Definition 4.6.1 (Prefix normal forms)
p is a prefix normal form, if it is of the form

$$\sum_i \int_{\alpha_i}(a_i(v) \cdot p_i) + \sum_j \int_{\beta_j} b_j(v)$$

where $\alpha_i, \beta_j \in Cond$, $a_i \in A$, $b_j \in A_\delta$ and each p_i is a prefix normal form as well.

Proposition 4.6.2 *For every $p \in T(\mathrm{BPA}\rho\delta\mathrm{I})$ there is a p' in prefix normal form such that* $\mathrm{BPA}\rho\delta\mathrm{I} \vdash p = p'$.

Proof. First we introduce some "intermediate" versions of prefix normal forms.

A U-prefix normal form is a prefix normal form as above, with that respect, that $\alpha_i, \beta_j \in Cond_U$ (they may still contain conditions of the form $U_b(z)$), and the variables that are bound by different initial integrals may differ as well. Furthermore, each p_i is a U-prefix normal form as well.

A var-prefix normal form is a prefix normal form as above, with that respect, that $\alpha_i, \beta_j \in Cond$, though the variables that are bound by different initial integrals may still differ. Furthermore, each p_i is a var-prefix normal form as well.

The proof consists of three steps. First we show that every term can be reduced to a U-prefix normal form. Next, we show how a U-prefix normal can be reduced to a var-prefix normal form, by replacing all conditions of the form $U_b(z)$ by conditions in *Cond*. Finally, we show that a var-prefix normal form can be reduced to a prefix normal form.

- First we discuss the cases where the subterms are already in U-prefix normal form.

  1. We show that for any substitution σ and U-prefix normal form z there is a U-prefix normal form u such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash \sigma(z) = u$. The cases where $z \equiv z_0 + z_1$ and $z \equiv \sigma'(z_0)$ follow directly from the axioms SU1 and SU4 respectively and by induction.

     Consider $\sigma(\int_\alpha a(v))$, we take a variable v' such that $\forall w \in fv(\int_\alpha a(v))$ we have $v' \notin \sigma(w)$. Then

$$\mathrm{BPA}\rho\delta\mathrm{I} \vdash \sigma(\int_\alpha a(v)) \overset{\mathrm{SU5}}{=} \sigma(\int_{\alpha[v'/v]} a(v')) \overset{\mathrm{SU2}}{=} \int_{\sigma[v'/v](\alpha)} a(v')$$

Consider $\sigma(\int_\alpha a(v) \cdot z_0)$, again, we take a variable v' such that $\forall w \in fv(\int_\alpha a(v) \cdot z_0)$ we have $v' \notin \sigma(w)$. By induction there is U-prefix normal form z_0' such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash z_0[v'/v] = z_0'$, and we have

$$\begin{array}{lll} \mathrm{BPA}\rho\delta\mathrm{I} \vdash & & \sigma(\int_\alpha a(v) \cdot z_0) \\ & \overset{\mathrm{SU6}}{=} & \sigma(\int_{\alpha[v'/v]} a(v') \cdot z_0[v'/v]) \\ & \overset{ind}{=} & \sigma(\int_{\alpha[v'/v]} a(v') \cdot z_0') \\ & \overset{\mathrm{SU3}}{=} & \int_{\sigma[v'/v](\alpha)} a(v') \cdot \sigma\backslash_{v'}(z_0') \end{array}$$

And we are ready by induction.

2. We show that for any b and U-prefix normal form z there is a U-prefix normal u such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash b \gg z = u$. We use induction to the size of z. In case $b \gg (z_0 + z_1)$ we reduce it to $b \gg z_0 + b \gg z_1$ and by induction we are ready.

   In case $b \gg \int_\alpha (a(v) \cdot z_0)$, then we take a variable $v' \notin var(b)$. By the previous case there is a U-prefix normal form z_0', such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash z_0[v'/v] = z_0'$. Then, we have

$$\begin{array}{lll} \mathrm{BPA}\rho\delta\mathrm{I} \vdash & & b \gg \int_\alpha (a(v) \cdot z_0) \\ & \overset{\mathrm{SU6}}{=} & b \gg \int_{\alpha[v'/v]} (a(v') \cdot z_0[v'/v]) \\ & \overset{\text{case } 1}{=} & b \gg \int_{\alpha[v'/v]} (a(v') \cdot z_0') \\ & \overset{\mathrm{RT2}_C}{=} & \int_{\alpha[v'/v] \wedge v' > b} (a(v') \cdot z_0') + \delta(b) \end{array}$$

   Similarly, we can show that for any α and any U-prefix normal form z, there is a U-prefix normal u such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash \alpha :\!\rightarrow z = u$.

3. We show that for any two U-prefix normal forms z and z' there is a U-prefix normal form u such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash z \cdot z' = u$. In case of $(z_0 + z_1) \cdot z'$ we reduce it to $z_0 \cdot z' + z_1 \cdot z'$, and we are ready by induction.

   Consider $\int_\alpha (a(v) \cdot z_0) \cdot z'$. Fix a variable v' such that $v' \notin fv(z_0)$. By the first case there is a U-prefix normal form z_0' such that $\mathrm{BPA}\rho\delta\mathrm{I} \vdash z_0[v'/v] = z_0'$. So, we have

$$\begin{array}{lll} \mathrm{BPA}\rho\delta\mathrm{I} \vdash & & \int_\alpha (a(v) \cdot z_0) \cdot z' \\ & \overset{\mathrm{SU6}}{=} & \int_{\alpha[v'/v]} (a(v') \cdot z_0[v'/v]) \cdot z' \\ & \overset{\text{case } 1}{=} & \int_{\alpha[v'/v]} (a(v') \cdot z_0') \cdot z' \\ & \overset{\mathrm{A5}^b_C}{=} & \int_{\alpha[v'/v]} (a(v') \cdot (z_0' \cdot z')), \end{array}$$

   and by induction we are ready.

Finally, we can prove by induction on the number of occurrences of $b \gg ..$, $\alpha :\!\rightarrow ..$, general multiplications and substitutions, that p can be reduced to U-prefix normal form.

- Assume we have a U-prefix normal form p, then we have to show that p can be reduced to a var-prefix normal form, that is, we have to show that all occurrences of $U_b(z)$ in p can be removed by conditions in α.

  For U-prefix normal forms we apply the usual definition of subterms; for a U-prefix normal form q with $\int_{\alpha \wedge U_b(z)}(a(v) \cdot q') \sqsubseteq q$, we do not consider z as a subterm of q.

  If $U_b(z)$ occurs in an initial integral of a certain subterm p' of p, then we replace z by its U-prefix normal form.

  We define the *U-depth* of a U-prefix normal form q as the longest chain $q \equiv q_0 \longrightarrow q_1 \longrightarrow \ldots q_n$ such that for some b the condition $U_b(p_{i+1})$ occurs in the condition of an initial integral of a subterm of q_i. We show by induction on *U-depth*(p) the the U-prefix normal form p can be reduced to a var-prefix normal form.

  If *U-depth*(p) $= 0$, then p is already a var-prefix normal form, and we are ready.

  Let *U-depth*(p) $= n > 0$, then we have to show that any occurrence of $U_b(p')$ in a subterm of p can be reduced to a condition $\alpha \in$ *Cond*. Obviouslu *U-depth*(p') $< n$, and by induction we have already constructed a var-prefix normal form p'' for p'. Consider a summand $\int_\alpha P(v)$ of p''. We can reduce this summand to $\sum_i \int_{\alpha_i \wedge v \in V_i} P(v)$, where $\{\alpha_i \wedge v \in V_i\}$ is the v-refinement of α. Hence, we have $U_b(\int_\alpha P(v)) = \vee_i \alpha_i \wedge b \leq \sup(V_i)$. Since, $U_b(p'')$ distributes over all summands, we have shown that there is a condition $\alpha \in$ *Cond* such that $U_b(p'')$ can be reduced to α, and we are ready.

- Assume we have a var-prefix normal form p, then we have to show that p can be reduced to a prefix normal form.

  Let *depth*(p) $= n$, take n time variables, $w_1, \ldots, w_n$, that do not occur free in p.

  Consider a subterm p' of p, with *depth*(p') $= k \leq n$ and let

$$p' \simeq \sum_i \int_{\alpha_i} (a_i(v_i) \cdot p_i) + \sum_j \int_{\beta_j} (b_j(v'_j))$$

  Assume that we have replaced all variables at depth $k' < k$ already by $w_{k'}$. We replace all v_i's and v'_j's by w_k and by the choice of w_k we can replace v_i in each p_i by w_k without any problems.

$\square$

In the above proof we have used that the set of time variables is infinite. Assume this set is finite; for example, that there are only two time variables, v_0, v_1. Then we can not reduce $(\int_{v_1=v_0}(a(v_1)))[v_1/v_0]$, as we need at least one other time variable v_2.

By abuse of notation we allow ourselves to omit the binding brackets of the integrals in a prefix normal form. So, we write

$$\sum_i \int_{\alpha_i} a_i(v) \cdot p_i + \sum_j \int_{\beta_j} b_j(v)$$

while we mean

$$\sum_i \int_{\alpha_i} (a_i(v) \cdot p_i) + \sum_j \int_{\beta_j} b_j(v).$$

Next, we construct for each pair p and q a *characterizing condition* which determines under which substitutions p and q bisimulate.

A term p in prefix normal form is also in *interval prefix normal form* if for every summand $\int_\alpha P(v)$ the condition α is of the form $v \in V$.

Lemma 4.6.3 (Characterizing condition lemma)
For all $p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$ we can construct a condition α with $var(\alpha) \subseteq fv(p+q)$ such that

$$\models \sigma(\alpha) \quad \Longleftrightarrow \quad \sigma(p) \leftrightarrows \sigma(q)$$

Proof. First we discuss the case where p and q are prefix normal forms, by induction to the size on $p+q$.

The first part of the proof presents the construction of the condition. In the second part, we prove that this condition indeed has the required properties.

(*Begin of construction.*)
First construct $\alpha(p,q)$ for the case where p, q are interval prefix normal forms. Assume

$$\begin{array}{rcl} p & \simeq & \sum_{i\in I} \int_{v\in V_i} a_i(v) \cdot p_i + \sum_{j\in J} \int_{v\in V'_j} b_j(v) \\ q & \simeq & \sum_{k\in K} \int_{v\in W_k} c_k(v) \cdot q_k + \sum_{l\in L} \int_{v\in W'_l} d_l(v) \end{array}$$

where $a_i, b_j, c_k, d_l \in A_\delta$

- Consider $\int_{v\in V} a(v) \cdot p' \sqsubseteq p$ where $a \in A$. We construct a condition φ such that $\int_{v\in V} a(v) \cdot p'$ is a semantic summand of q. That is, if $\models \sigma(\varphi)$ and $\sigma(\int_{v\in V} a(v) \cdot p') \xrightarrow{a(r)} z$ then there is a z' such that $\sigma(q) \xrightarrow{a(r)} z'$ and $z \leftrightarrows z'$.

  Take $K(a) = \{k \in K | c_k = a\}$. By induction there is for each $k \in K(a)$ a condition α_k such that

  $$\models \sigma(\alpha_k) \quad \Longleftrightarrow \quad \sigma(v \gg p') \leftrightarrows \sigma(v \gg q_k)$$

  By the Refinement Lemma (4.2.2) there is for α_k a refinement

  $$\{\beta_k^x \wedge v \in Z_k^x\}$$

where $v \notin var(\beta_k^x) \cup var(Z_k^x)$. We construct a partition $\{\gamma_y\}$ which refines each $\{\beta_k^x\}$, by taking the cartesian product of all partitions $\{\beta_k^x\}$. Hence, for each y there is a index set $X(k,y)$ such that $x \in X(k,y)$ implies $\gamma_y \Rightarrow \beta_k^x$. Furthermore, since $\{\beta_k^x\}$ is a partition we have that $x, x' \in X(k,y)$ implies that $\beta_k^x = \beta_k^{x'}$.

The condition that $\int_{v \in V} a(v) \cdot p'$ is a summand of

$$\sum_{k \in K(a)} \int_{v \in W_k} a(w) \cdot q_k$$

is denoted by the following

$$\varphi = \bigvee_y \gamma_y \wedge V \subseteq \bigcup_{k \in K(a)} (W_k \cap \bigcup_{x \in X(k,y)} Z_k^x).$$

Note that $y \neq y'$ implies $\gamma_y \wedge \gamma_{y'} = \mathit{ff}$.

- Consider $\int_{v \in V} a(v) \sqsubseteq p$ where $a \in A$, then the condition χ such that $\int_{v \in V} a(v)$ is a semantic summand of q is simply

$$\chi = V \subseteq \bigcup_{L(a)} W_l'$$

We do the same for every syntactic summand of q. Let $I(A) = \{i \in I | a_i \in A\}$, that is the subset of I of non-δ summands. Similarly we have $J(A), K(A)$ and $L(A)$. We have obtained:

for each	$i \in I(A)$:	φ_i
for each	$j \in J(A)$:	χ_j
for each	$k \in K(A)$:	ψ_k
for each	$l \in L(A)$:	ω_l

Finally we construct $U_=(p,q)$ as follows. Take an arbitrary time variable $v \notin fv(p+q)$, then we will construct a condition in which v does not occur, and which is equivalent with $U_v(p) \Leftrightarrow U_v(q)$.

As p has only summands of the form $\int_{v \in V} P(v)$ we know that $U_v(p)$ reduces to $\vee_i \; v < \sup(V_i)$. Furthermore, $v < \sup(V)$ reduces to $V \neq \emptyset \wedge v < \sup(V)$, which can be reduced further to $V \neq \emptyset \wedge v < b$, for some bound $b \in Bound_\infty$. Hence, $U_v(p)$ can be reduced further to $\vee_i \; V_i \neq \emptyset \wedge v < b_i$. We can reduce this latter condition to the form $\vee_i \; \alpha_i \wedge v < b_i$, where $\{\alpha_i\}$ is a partition such that $\alpha_i \wedge \alpha_{i'} = \mathit{ff}$, and $b_i \in Bound_{-\infty,\infty}$ Similarly, we can reduce $U_v(q)$ to $\vee_j \; \beta_j \wedge v < b_j'$.

We take the cartesian product of $\{\alpha_i\}$ and $\{\beta_j\}$, and we reduce $U_v(p) \Leftrightarrow U_v(q)$ to the condition

$$\{\vee_{i,j} \; \alpha_i \wedge \beta_j \wedge v < b_i\} \Leftrightarrow \{\vee_{i,j} \; \alpha_i \wedge \beta_j \wedge v < b_j'\}$$

which can be further reduced to

$$\vee_{i,j}\ \alpha_i \wedge \beta_j \wedge b_i = b'_j$$

and we call this latter condition $U_=(p,q)$.

We collect all the conditions we have constructed so far, and we define $\alpha(p,q)$ by

$$\alpha(p,q) = \bigwedge_{i\in I(A)} \varphi_i \wedge \bigwedge_{j\in J(A)} \chi_j \wedge \bigwedge_{k\in K(A)} \psi_k \wedge \bigwedge_{l\in L(A)} \omega_l \wedge U_=(p,q)$$

Next, suppose p or q are prefix normal forms, but not *interval* prefix normal forms. Then we rewrite each summand $\int_\gamma a(v)\cdot z$ of p and q, to $\sum_l \gamma_l :\!\!\longrightarrow \int_{v\in V_l} z$, where $\{\gamma_l \wedge v \in V_l\}$ is the v-refinement of γ. In this way we obtain for p a term $\sum_k \gamma'_k :\!\!\longrightarrow p_k$, and we rewrite it further to $\sum_i \alpha_i :\!\!\longrightarrow p_i$ such that $\{\alpha_i\}$ is a partition. We do the same for q, by which we obtain $\sum_j \beta_j :\!\!\longrightarrow q_j$. Finally, we take

$$\alpha(p,q) = \bigvee_{i,j} \alpha_i \wedge \beta_j \wedge \alpha(p_i, q_j).$$

(End of construction.)

It is now left to prove that for any σ

$$\models \sigma(\alpha) \quad\Longleftrightarrow\quad \sigma(p) \leftrightarrows \sigma(q)$$

We prove it for interval prefix normal forms. The case where p and q are not both in interval prefix normal forms is left to the reader.

$\Longrightarrow$ Take σ such that $\models \sigma(\alpha)$. We will show $\sigma(p) \leftrightarrows \sigma(q)$.

- Consider a transition $\sigma(p) \xrightarrow{a(r)} z$ then we will show that there is a z' such that $\sigma(q) \xrightarrow{a(r)} z'$ and $z \leftrightarrows z'$.

  For $\sigma(p) \xrightarrow{a(r)} z$ there must be an index i such that $a = a_i$ and $z \equiv \sigma[r/v](v \gg p_i)$. Since $\alpha \Rightarrow \varphi_i$ we have $\models \sigma(\varphi_i)$.

  Let φ be of the form

  $$\bigvee_y \gamma_y \wedge V \subseteq \bigcup_{k\in K(a)} (W_k \cap \bigcup_{x\in X(k,y)} Z_k^x),$$

  as constructed above, then there is exactly one y such that $\models \sigma(\gamma_y)$. Moreover, there must be a $k \in K(a)$ and an $x \in X(k,y)$ such that $r \in \sigma(W_k \cap Z_k^x)$ and thus

  $$\sigma \quad \xrightarrow{a(r)} \sigma[r/v](v \gg q_k)(q)$$

  Since

  $$\models \sigma[r/v](\beta_k^x \ \wedge\ v \in Z_k^x)$$

  we have by induction

$$\sigma \qquad {}_i) \leftrightarrows \sigma[r/v]([r/v](q_k)) \gg p$$

- Consider $\sigma(p) \xrightarrow{a(r)} \surd$, then there is a j such that $a = b_j$ and $r \in \sigma(V_j')$. Since $\models \sigma(\chi_j)$ and $r \in \sigma(V_j')$ there must be an index $l \in L(a)$ and $r \in \sigma(W_l')$ hence $\sigma(q) \xrightarrow{a(r)} \surd$.
- Take a t such that $U_t(\sigma(p))$, then we have to show that $U_t(\sigma(q))$.
  Fix i, j such that $\models \sigma(\alpha_i \wedge \beta_j)$, where α_i, β_j are taken from the construction of $U_=(p, q)$. Without proof we state that $U_t(\sigma(p))$ iff $\models t < \sigma(b_i)$ and $U_t(\sigma(q))$ iff $\models t < \sigma(b_j')$. Since $\models \sigma(b_i = b_j')$, it follows that $U_t(\sigma(q))$ as well.

And by symmetry we are done.

$\Longleftarrow$ Take σ such that $\sigma(p) \leftrightarrows \sigma(q)$ then we have to show that $\models \sigma(\alpha)$.

- Fix an $i \in L(a)$ then we show that $\models \sigma(\varphi_i)$. Take $a = a_i$, $V = V_i, p' = p_i$ and $\varphi = \varphi_i$. Let φ be of the form

  $$\bigvee_y (\gamma_y \wedge V \subseteq \bigcup_{k \in K(a)} (W_k \cap \bigcup_{x \in X(k,y)} Z_k^x)),$$

  as constructed above, then there is exactly one y such that $\models \sigma(\gamma_y)$ and it is left to prove that

  $$\sigma \qquad \bigcup_{k \in K(a)} (W_k \cap \bigcup_{x \in X(k,y)} (V) \supseteq \sigma($$

  Take an arbitrary $r \in \sigma(V)$, then

  $$\sigma \qquad \xrightarrow{a(r)} \sigma[r/v](v \gg p')\,(p)$$

  Then since $\sigma(p) \leftrightarrows \sigma(q)$ there must be an index j such that

  $$\sigma \qquad \xrightarrow{b_j(r)} \sigma[r/v](v \gg q_j)\,(q)$$

  where $b_j = a$ and $\sigma[r/v](v \gg p') \leftrightarrows \sigma[r/v](v \gg q_j)$. By induction there is a characterizing condition $\alpha(p', q_j)$ such that $\models \sigma[r/v](\alpha(p', q_j))$. Consider the extra v-refinement of this condition $\alpha(p', q_j)$, let it be of the form $\{\beta_k^x \ \wedge \ v \in Z_k^x\}$, then there is exactly one index x such that

  $$\models \sigma[r/v](\beta_k^x \ \wedge \ v \in Z_k^x)$$

  and since $r \in \sigma(W_k)$ we have

  $$r \in \sigma(W_k \ \wedge \ Z_k^x)$$

  and as well

  $$r \in \sigma(\bigcup_{k' \in K(a)} (W_{k'} \ \bigcup_{x \in X(k,y)} Z_k^x)).$$

  Since this holds for arbitrary $r \in \sigma(V)$ we conclude

$$\sigma \qquad \bigcup_{k \in K(a)} (W_k \cap \bigcup_{x \in X(k,y)} (V) \; [illegible] \; \sigma($$

- The proof that $\models \sigma(\chi_j)$ is left to the reader.
- We have to show that $\models \sigma(U_=(p,q))$. It is sufficient to show that $\models \sigma(b_i = b'_j)$, where i, j such that $\models \sigma(\alpha_i \wedge \beta_j)$.
  Assume $\models \sigma(b_i \neq b'_j)$. Then we can find a t such that $\models \sigma(b_i) < t < \sigma(b'_j)$. Hence, $\neg U_t(\sigma(p))$ and $U_t(\sigma(q))$, but this contradicts $\sigma(p) \underline{\leftrightarrow} \sigma(q)$, and we are ready.

And by symmetry we are done.

If p or q is not in prefix normal form, then we can develop an algorithm, based on Proposition 4.6.2, that assigns to each process term z a prefix normal form z_{pnf}. Note that $\forall \sigma \in \Sigma^{cl}$ we have $\sigma(z) \underline{\leftrightarrow} \sigma(z_{pnf})$. We put $\alpha(p,q) = \alpha(p_{pnf}, q_{pnf})$, and we are ready. □

Proposition 4.6.4

$$p \underline{\leftrightarrow}^{\alpha} q \quad \Longrightarrow \quad \mathrm{CA} \vdash (\alpha \Rightarrow \alpha(p,q)) = tt$$

Proof.

$$\begin{aligned} p \underline{\leftrightarrow}^{\alpha} q &\Longrightarrow \forall \sigma \in [\alpha] \quad \sigma(p) \underline{\leftrightarrow} \sigma(q) \\ &\Longrightarrow [\alpha] \subseteq [\alpha(p,q)] \\ &\Longrightarrow [\alpha \Rightarrow \alpha(p,q)] = \Sigma^{cl} = [tt] \\ &\Longrightarrow \mathrm{CA} \vdash (\alpha \Rightarrow \alpha(p,q)) = tt \end{aligned}$$

□

Corollary 4.6.5 (Decidability of $\underline{\leftrightarrow}^{\alpha}$)
For each $p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$ and each condition α we can decide whether $p \underline{\leftrightarrow}^{\alpha} q$

Proof. Construct the characterizing condition $\alpha(p,q)$ as is done in the proof of the previous lemma. Then $p \underline{\leftrightarrow}^{\alpha} q$ whenever $(\alpha \Rightarrow \alpha(p,q)) = tt$. □

We abbreviate $\mathrm{BPA}\rho\delta\mathrm{I} \vdash \alpha :\rightarrow p = \alpha :\rightarrow q$ by $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash p = q$.

Lemma 4.6.6 $\forall p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$

$$\mathrm{BPA}\rho\delta\mathrm{I}, \alpha(p,q) \vdash p = q$$

Proof. We prove the theorem first for the case where p and q are interval prefix normal forms. We abbreviate $\alpha(p,q)$ by α and we assume

$$\begin{aligned} p &\simeq \textstyle\sum_{i \in I} \int_{v \in V_i} a_i(v) \cdot p_i + \sum_{j \in J} \int_{v \in V'_j} b_j(v) \\ q &\simeq \textstyle\sum_{k \in K} \int_{v \in W_k} c_k(v) \cdot q_k + \sum_{l \in L} \int_{v \in W'_l} d_l(v) \end{aligned}$$

where $a_i, c_k, b_j, d_l \in A_\delta$.

- Consider $\int_{v\in V} a(v) \cdot p' \sqsubseteq p$. Then $\alpha \Rightarrow \bigvee_y \varphi_y$ where φ_y denotes the condition

  $$\gamma_y \wedge V \subseteq \bigcup_{k\in K(a)} (W_k \cap \bigcup_{x\in X(k,y)} Z_k^x)$$

  such that $\gamma_y \wedge v \in Z_k^x \Rightarrow \alpha(p', q_k) = tt$

  Without proof we state

  $$\text{BPA}\rho\delta\text{I}, V \subseteq \bigcup_k W_k \vdash \int_{v\in V} P(v) \subseteq \sum_k \int_{v\in W_k} P(v)$$

  And the induction hypothesis, together with

  $$\gamma_y \wedge v \in Z_k^x \Rightarrow \alpha(p', q_k) = tt,$$

  says us that

  $$\text{BPA}\rho\delta\text{I}, \gamma_y \wedge v \in Z_k^x \vdash p' \overset{ind}{=} q_k$$

  And we prove BPA$\rho\delta$I, $\varphi_y \vdash$

  $$\begin{array}{ll} & \int_{v\in V} a(v) \cdot p' \\ \subseteq & \sum_{k\in K(a)} \sum_{x\in X(k,y)} \int_{v\in W_k\cap Z_k^x} a(v) \cdot p' \\ = & \sum_{k\in K(a)} \sum_{x\in X(k,y)} \int_{v\in W_k\cap Z_k^x} a(v) \cdot \{\varphi_y \wedge v \in Z_k^x :\to p'\} \\ \overset{ind}{=} & \sum_{k\in K(a)} \sum_{x\in X(k,y)} \int_{v\in W_k\cap Z_k^x} a(v) \cdot \{\varphi_y \wedge v \in Z_k^x :\to q_k\} \\ = & \sum_{k\in K(a)} \sum_{x\in X(k,y)} \int_{v\in W_k\cap Z_k^x} a(v) \cdot q_k \\ \subseteq & \sum_{k\in K(a)} \int_{v\in W_k} a(v) \cdot q_k \end{array}$$

  And thus BPA$\rho\delta$I, $\alpha \vdash \int_{v\in V} a(v) \cdot p' \subseteq q$

- Consider $\int_{v\in V} a(v) \sqsubseteq p$ where $a \in A$. Then $\alpha \Rightarrow V \subseteq \bigcup_{l\in L(a)} W_l'$ and we have

  $$\text{BPA}\rho\delta\text{I}, \alpha \vdash \int_{v\in V} a(v) \subseteq \sum_{l\in L(a)} \int_{v\in W_l'} d_l(v)$$

  And thus BPA$\rho\delta$I, $\alpha \vdash \int_{v\in V} a(v) \subseteq q$.

- Consider $\int_{v\in V} \delta(v) \sqsubseteq p$.

  Take $\gamma_{i,j} = \alpha_i \wedge \beta_j \wedge \max(\overline{b}_{i,j}) = \max(\overline{b'}_{i,j})$, (see construction of $U_=(p,q)$ in the proof of 4.6.3). Then we show $\gamma_{i,j} \vdash \int_{v\in V} \subseteq q$.

  By construction, we have either CA $\vdash \gamma_{i,j} \Rightarrow V = \emptyset) = tt$ or CA $\vdash \gamma_{i,j} \Rightarrow V \neq \emptyset) = tt$. In the first case we have

$$\mathrm{BPA}\rho\delta\mathrm{I}, \gamma_{i,j} \vdash \int_{v \in V} \delta(v) = \int_{v \in \emptyset} \delta(v) = \int_{ff} \delta(v) = \delta \subseteq q$$

and in the second case we have

$$\mathrm{BPA}\rho\delta\mathrm{I}, \gamma_{i,j} \vdash \int_{v \in V} \delta(v) \subseteq \int_{v \leq \max(\bar{b}_{i,j})} = \int_{v \leq \max(\overline{b'}_{i,j})} \subseteq q$$

Hence, $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash p \subseteq q$ and by symmetry also $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash q \subseteq p$ from which we obtain $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash p = q$.

If p, q are not both in interval prefix normal forms, then we can rewrite them into $\sum_i \alpha_i :\rightarrow p_i$ and $\sum_j \beta_j :\rightarrow q_j$ respectively, where $\{\alpha_i\}$, $\{\beta_j\}$ are partitions and p_i, q_j are interval prefix normal forms. Then $\alpha(p,q) = \vee_{i,j} \alpha_i \wedge \beta_j \wedge \alpha(p_i, q_j)$ Since $\{\alpha_i \wedge \beta_j\}_{i,j}$ is a partition it is sufficient to show that for each (i,j) we have $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha_i \wedge \beta_j \wedge \alpha(p,q) \vdash p = q$ which reduces to $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha_i \wedge \beta_j \wedge \alpha(p_i, q_j) \vdash p_i = q_j$ and we are ready.

If p or q are not in prefix normal form then we have $\alpha(p,q) \overset{def}{=} \alpha(p_{pnf}, q_{pnf})$ moreover we have $\mathrm{BPA}\rho\delta\mathrm{I} \vdash p = p_{pnf}$ and thus

$$\mathrm{BPA}\rho\delta\mathrm{I}, \alpha(p,q) \vdash p = p_{hnf} = q_{hnf} = q$$

□

Theorem 4.6.7 (Completeness of $\mathrm{BPA}\rho\delta\mathrm{I}$**)** $\forall p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$

$$p \leftrightarrows^{\alpha} q \quad \Longrightarrow \quad \mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash p = q$$

Proof. In $\mathrm{BPA}\rho\delta\mathrm{I}$ we have $\mathrm{CA} \vdash (\beta \Rightarrow \beta') = tt$ and $\mathrm{BPA}\rho\delta\mathrm{I}, \beta' \vdash z = z'$ imply that $\mathrm{BPA}\rho\delta\mathrm{I}, \beta \vdash z = z'$ as well.

Hence, the completeness follows direct from $\mathrm{CA} \vdash (\alpha \Rightarrow \alpha(p,q)) = tt$ and Lemma 4.6.6. □

Note, that the completeness and decidability of $\mathrm{BPA}\rho\delta\mathrm{I}$ are heavily dependent on the characterizing condition lemma (Lemma 4.6.3), that depends in turn on the refinement lemma for conditions (Lemma 4.2.2). These lemmas motivate the restriction to prefixed integration. Due to the restriction to subsets that can be described by our conditions we can use our the refinement lemma. Due to the restriction that every integral must preceed directly the action that uses the bound variable we can construct characterizing lemmas.

5

ACP with Prefixed Integration

5.1 Introduction

In this section we extend BPA$\rho\delta$I with the operators $\|, |, \lfloor\!\lfloor$ and ∂_H. The set $T(\text{ACP}\rho\text{I})$ is defined by the following BNF sentence, where $a \in A_\delta$, $\alpha \in \mathit{Cond}_U$, $b \in \mathit{Bound}$ and $H \subseteq A$.

$$\begin{aligned} p ::= \quad & \textstyle\int_\alpha a(v) \mid \int_\alpha (a(v) \cdot p) \mid p + p \mid p \cdot p \mid b \gg p \mid \alpha :\!\rightarrow p \mid \sigma(p) \mid \\ & p \| p \mid p \lfloor\!\lfloor p \mid p | p \mid \partial_H(p) \end{aligned}$$

There is no need any more for the operator $p \gg b$, as will be discussed later. We extend the definition of the set fv of free time variables, see Subsection 4.3.3, by putting $fv(p\|q) = fv(p|q) = fv(p \lfloor\!\lfloor q) = fv(p) \cup fv(q)$ and $fv(\partial_H(p)) = fv(p)$.

The action rules will not be given in this chapter, they can be gathered from the Tables 2.3, 3.5, 4.6, 4.4. and Table 4.6. The action rules for substitution for a term semantics for the additional ACPρI operators, such as given in Table 4.6 for BPA$\rho\delta$, are left to the reader.

We inherit the definition of bisimulation equivalence for time open process terms from Chapter 4, Definition 4.4.1. It is left to the reader to prove that bisimulation equivalence for time open terms is a congruence for ACPρI. As in the previous chapter one has to give a Σ-semantics in the path format of Baeten and Verhoef, and one has to prove that the bisimulation equivalence of this Σ-semantics coincides with the bisimulation equivalence of the term semantics.

5.2 The Axiom System ACPρI

We obtain the axiom system ACPρI by adding the axioms of Table 5.1 to the axiom system BPA$\rho\delta$I, where p, q range over $T(\text{ACP}\rho\text{I})$. We have the following theorem.

Theorem 5.2.1 $p, q \in T(\text{ACP}\rho\text{I})$

$$\text{ACP}\rho\text{I} \vdash p = q \quad \Longrightarrow \quad p \leftrightarroweq q$$

CF_I	$\int_\alpha a(v) \mid \int_\beta b(v)$	$=$	$\int_{\alpha\wedge\beta} \gamma(a,b)(v) + \int_{U_v(\alpha,\beta)} \delta(v)$	
CM1	$p \parallel q$	$=$	$p \llfloor q + q \llfloor p + p \mid q$	
$\mathrm{CM2}_I$	$v \notin fv(p)$			
	$\int_\alpha a(v) \llfloor p$	$=$	$\int_{\alpha\wedge U_v(p)}(a(v)\cdot p) + \int_{U_v(\alpha,p)} \delta(v)$	
$\mathrm{CM3}_I$	$v \notin fv(q)$			
	$\int_\alpha(a(v)\cdot p) \llfloor q$	$=$	$\int_{\alpha\wedge U_v(q)}(a(v)\cdot(p \parallel q)) + \int_{U_v(\alpha,q)} \delta(v)$	
CM4	$(p_1+p_2) \llfloor q$	$=$	$p_1 \llfloor q + p_2 \llfloor q$	
$\mathrm{CM5}_I$	$\int_\alpha(a(v)\cdot p) \mid (\int_\beta b(v))$	$=$	$\int_{\alpha\wedge\beta}(\gamma(a,b)\cdot p) + \int_{U_v(\alpha,\beta)} \delta(v)$	
$\mathrm{CM6}_I$	$\int_\alpha a(v) \mid \int_\beta(b(v)\cdot p)$	$=$	$\int_{\alpha\wedge\beta}(\gamma(a,b)\cdot p) + \int_{U_v(\alpha,\beta)} \delta(v)$	
$\mathrm{CM7}_I$	$\int_\alpha(a(v)\cdot p) \mid \int_\beta(b(v)\cdot q)$	$=$	$\int_{\alpha\wedge\beta}(\gamma(a,b)(v)\cdot(p \parallel q)) + \int_{U_v(\alpha,\beta)} \delta(v)$	
CM8	$(p_1+p_2) \mid q$	$=$	$p_1 \mid q + p_2 \mid q$	
CM9	$p \mid (q_1+q_2)$	$=$	$p \mid q_1 + p \mid q_2$	
$\mathrm{D1}_I^a$	$\partial_H(\int_\alpha a(v))$	$=$	$\int_\alpha a(v)$	$a \notin H$
$\mathrm{D1}_I^b$	$\partial_H(\int_\alpha(a(v)\cdot p))$	$=$	$\int_\alpha(a(v)\cdot\partial_H(p))$	$a \notin H$
$\mathrm{D2}_I^a$	$\partial_H(\int_\alpha a(v))$	$=$	$\int_\alpha \delta(v)$	$a \in H$
$\mathrm{D2}_I^b$	$\partial_H(\int_\alpha(a(v)\cdot p))$	$=$	$\int_\alpha \delta(v)$	$a \in H$
D3	$\partial_H(p+q)$	$=$	$\partial_H(p)+\partial_H(q)$	

$(a, b \in A_\delta)$

Table 5.1: Additional axioms for ACPρI

Proof. Omitted. □

We use the following abbreviations.

$$\begin{array}{lcl} U_v(\alpha) & \overset{abb}{=} & U_v(\int_\alpha \delta(v)) \\ \int_{U_v(\alpha,\beta)} \delta(v) & \overset{abb}{=} & \int_{U_v(\alpha)\wedge U_v(\beta)} \delta(v) \\ \int_{U_v(\alpha,p)} \delta(v) & \overset{abb}{=} & \int_{U_v(\alpha)\wedge U_v(p)} \delta(v) \end{array}$$

Note that $U_v(\alpha) = U_v(\int_\alpha P(v))$ for arbitrary $P(v)$, in other words, $U_v(\alpha)$ expresses the idle behavior of an arbitrary process term $\int_\alpha P(v)$. Furthermore $U_v(\alpha, \beta)$ expresses the idle behavior of terms like $\int_\alpha P(v) \| \int_\beta P(v)$, $\int_\alpha P(v) | \int_\beta P(v)$ and $\int_\alpha P(v) ⌊⌊ \int_\beta P(v)$.

The axioms CF$_I$, CM2$_I$,3$_I$ and CM5$_I$-7$_I$ have a $\int_{U_v(\ldots)} \delta(v)$ summand on their righthand side for the case that the other summand on the righthand side is of the form $\int_{ff} P(v)$. For in this case we have to guarantee that the process terms left and right from the =-sign have the same idle behavior.

Consider the axiom CF$_I$, and take $p \equiv \int_{v\le 1} a(v)$ and $q \equiv \int_{v=2} b(v)$, so $\alpha = (v \le 1)$ and $\beta = (v = 2)$. Obviously $\alpha \wedge \beta = ff$, which means that no communication is possible. Hence, $p|q$ can not execute any actions, and since $p|q$ is able to idle until 1, the axiom CF$_I$ must imply $p|q = \delta(1)$. Hence, we have to add a δ summand on the right hand side of CF$_I$ that has the same idle behavior as $p|q$. We denote this particular δ-summand by $\int_{U_v(v\le 1, v=2)} \delta(v)$, that abbreviates $\int_{U_v(\int_{v\le 1} \delta(v))\wedge U_v(\int_{v=2} \delta(v))} \delta(v)$, that reduces to $\int_{v\le 1 \wedge v \le 2} \delta(v)$, that reduces further to $\int_{v\le 1} \delta(v)$. Finally, we note that $\int_{v\le 1} \delta(v)$ indeed equals $\delta(1)$.

In ACPρ we need the auxiliary operator $p \gg t$ to axiomatize the left merge, see Table 3.3:

$$\text{CM2}_\rho \quad a(r) ⌊⌊ p \;=\; (a(r) \gg U(p)) \cdot p,$$

since the $a(r)$ is enabled only in case $U(p) > r$. In the context of prefixed integration we can easily express this phenomenon in the condition of the integral, as is done in the axiom CM2$_I$. Hence, there is no need for the operator $p \gg b$ in ACPρI. For example, take $p_0 \equiv \int_{v\le 5} a(v)$, $p_1 \equiv \int_{v>5} a(v)$ and $q \equiv b(3)$. Then $U_v(b(3)) = (v \le 3)$. The expressions $\int_{U_v(v\le 5,q)} \delta(v)$ and $\int_{U_v(v>5,q)} \delta(v)$ both denote processes that equal $\int_{v\le 3} \delta(v)$.

$$\begin{array}{cl} & p_0 ⌊⌊ q \\ = & \int_{v\le 5} a(v) ⌊⌊ b(3) \\ = & \int_{v\le 5\wedge v\le 3}(a(v)\cdot b(3)) + \int_{v\le 3} \delta(v) \\ = & \int_{v\le 3}(a(v)\cdot b(3)) + \int_{v\le 3} \delta(v) \\ = & \int_{v\le 3}(a(v)\cdot b(3)) \end{array} \qquad \begin{array}{cl} & p_1 ⌊⌊ q \\ = & \int_{v>5} a(v) ⌊⌊ b(3) \\ = & \int_{v>5\wedge v\le 3}(a(v)\cdot b(3)) + \int_{v\le 3} \delta(v) \\ = & \int_{ff}(a(v)\cdot b(3)) + \int_{v\le 3} \delta(v) \\ = & \delta + \int_{v\le 3} \delta(v) \\ = & \int_{v\le 3} \delta(v) \end{array}$$

5.3 Elimination and Completeness

Theorem 5.3.1 (Elimination Theorem for ACP**)**
$\forall p \in T(\mathrm{ACP}\rho\mathrm{I})$ $\exists p'$ *where* p' *is in prefix normal form and* $\mathrm{ACP} \vdash p = p'$

Proof. The proof is a combination of the proof of Theorem 1.3.1 (the Elimination Theorem for ACP) and that of Theorem 4.6.2 (every term in BPA$\rho\delta$I can be reduced to a prefix normal form).

In the proof of Theorem 4.6.2 we defined the auxiliary notion of an U-prefixed normal form, that is a term that is "almost" in normal form, in the sense that conditions of the form $U_b(p)$ are still allowed and summands may still bind different variables.

Following the proof of Theorem 4.6.2 it is sufficient to show that for any two U-prefix normal forms z and z' and $\Box \in \{\|, \mathbin{\lfloor\!\lfloor}, |\}$ there is a U-prefix normal form u such that $\mathrm{ACP}\rho \vdash z \Box z' = u$. As usual we show this by induction; as norm for the induction we take $depth(z + z', \Box)$ as in the proof of Theorem 1.3.1. Also we can show by induction on z, where z is a U-prefix normal form, that there is a U-prefix normal form u such that $\mathrm{ACP}\rho \vdash \partial_H(z) = u$, this case is left to the reader.

There is only one complication with respect to the previous elimination proofs; in some cases we have to take "fresh" variables in some of the components.

$$\begin{array}{l}
\int_\alpha a(v) \mathbin{\lfloor\!\lfloor} z' = \\
\int_{\alpha[v'/v]\wedge U_v(z')} (a(v') \cdot z') + \int_{U_{v'}(\alpha, z')} \delta(v) \\
\text{ for some } v' \notin fv(\int_\alpha a(v) + z')
\end{array}$$

$$\begin{array}{l}
\int_\alpha (a(v) \cdot z_0) \mathbin{\lfloor\!\lfloor} z' = \\
\int_{\alpha[v'/v]\wedge U_{v'}(z')} (a(v') \cdot (z_0[v'/v] \| z')) + \int_{U_v(\alpha, z')} \delta(v) \\
\text{ for some } v' \notin fv(\int_\alpha (a(v) \cdot z_0) + z')
\end{array}$$

$$(z_0 + z_1) \mathbin{\lfloor\!\lfloor} z' \;=\; z_0 \mathbin{\lfloor\!\lfloor} z' + z_1 \mathbin{\lfloor\!\lfloor} z'$$

$$\begin{array}{lcl}
(z_0 + z_1) | z' & = & z_0 | z' + z_1 | z' \\
z | (z'_0 + z'_1) & = & z | z'_0 + z | z'_1
\end{array}$$

$$\begin{array}{l}
\int_\alpha a(v_0) | \int_\beta b(v_1) \\
= \; \int_{\alpha[v/v_0]\wedge\beta[v/v_1]} \gamma(a, b)(v) + \int_{U_v(\alpha[v/v_0], \beta[v/v_1])} \delta(v) \\
\text{ for some } v \notin fv(\int_\alpha a(v_0) + \int_\beta b(v_1))
\end{array}$$

$$\begin{array}{l}
\int_\alpha (a(v_0) \cdot z_0) | \int_\beta b(v_1) \\
= \; \int_{\alpha[v/v_0]\wedge\beta[v/v_1]} (\gamma(a, b)(v) \cdot z_0[v/v_0]) + \int_{U_v(\alpha[v/v_0], \beta[v/v_1])} \delta(v) \\
\text{ for some } v \notin fv(\int_\alpha (a(v_0) \cdot z_0) + \int_\beta b(v_1))
\end{array}$$

$$\begin{aligned}
&\textstyle\int_\alpha a(v_0) | \int_\beta (b(v_1) \cdot z_0') \\
&= \textstyle\int_{\alpha[v/v_0] \wedge \beta[v/v_1]} (\gamma(a,b)(v) \cdot z_0'[v/v_1]) + \int_{U_v(\alpha[v/v_0], \beta[v/v_1])} \delta(v) \\
&\text{ for some } v \notin fv(\textstyle\int_\alpha a(v_0) + \int_\beta (b(v_1) \cdot z_0'))
\end{aligned}$$

$$\begin{aligned}
&\textstyle\int_\alpha (a(v_0) \cdot z_0) | \int_\beta (b(v_1) \cdot z_0') \\
&= \textstyle\int_{\alpha[v/v_0] \wedge \beta[v/v_1]} (\gamma(a,b)(v) \cdot (z_0[v/v_0] \| z_0'[v/v_1])) \\
&\qquad\qquad\qquad\qquad + \textstyle\int_{U_v(\alpha[v/v_0], \beta[v/v_1])} \delta(v) \\
&\text{ for some } v \notin fv(\textstyle\int_\alpha (a(v_0) \cdot z_0) + \int_\beta (b(v_1) \cdot z_0'))
\end{aligned}$$

$$z \| z' \;=\; z \mathbin{\|\!\_} z' + z' \mathbin{\|\!\_} z + z | z'$$

□

Finally, we have the completeness for ACPρI.

Theorem 5.3.2 (Completeness for ACPρI) $p, q \in T(\text{ACP}\rho\text{I})$

$$p \leftrightarrows^{\alpha} q \quad\Longrightarrow\quad \text{ACP}\rho\text{I}, \alpha \vdash p = q$$

Proof. In the proof of the completeness for ACPρ (see Theorem 3.6.2) we have shown how the completeness follows from the Elimination Theorem. □

Part III

The Silent Step in Time

6

Branching Bisimulation and Time

6.1 Introduction

In this chapter we propose a notion of branching bisimulation for Real Time Process Algebra. An earlier presentation of the contents of this chapter can be found in [Klu92].

In Section 6.2 we argue that when we deal with time, a τ-transition can be matched with an idle transition, and vice versa. We will show this by various examples.

In Section 6.3 we give the formal definitions of branching bisimulation in the context of the idle semantics. We show as well that we need a rooted version to obtain a congruence.

In Section 6.4 we present a law for branching bisimulation equivalence over BPA$\rho\delta\tau$, and we encounter various conditions on the time stamps involved. This law allows us to remove at τ, if it does not determine a moment of choice.

In Section 6.5 we generalize this law to the setting of prefixed integration and we obtain a law that corresponds closely with the branching law B2 of Section 1.4. We encounter non trivial conditions on the bounds involved.

We discuss in Section 6.6 the embedding of BPA$\delta\tau$ in BPA$\rho\delta\tau$I.

In Section 6.7 we develop a definition of a term branching bisimulation by changing the definition of idle branching bisimulation step by step. Finally, in Section 6.8 we show that rooted branching bisimulation equivalence is a congruence.

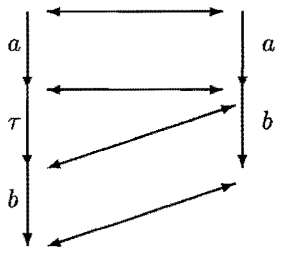
The proof that the law for branching bisimulation equivalence with integration is complete, is postponed to Chapter 7. We weaken the definition of branching bisimulation to delay and even further to weak bisimulation in Chapter 8. Branching bisimulation in the context of guarded recursion is discussed in Chapter 9, and in Chapter 10 we verify a protocol using rooted branching bisimulation equivalence. In Chapter 11 we define branching bisimulation in the context of a two phase semantics, that is a semantics in which consecutive actions at the same point in time are allowed. We show that in such a semantics the notion of branching bisimulation and the resulting equivalence differs from the one in the present chapter. In the last chapter of this thesis, Chapter 12, we relate our work on abstraction with other papers, that discuss weak bisimulation only.

6.2 Some Examples

In untimed branching bisimulation it is allowed that

a τ-transition on one side may be matched with *no transition* *at all at the other side.*

if this τ-transition does not determine a choice. Take for example:

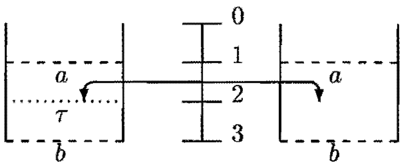


In our real time context each transition increases the course of time and we relate states with the same *time* value only. A statement analogous to the above one is

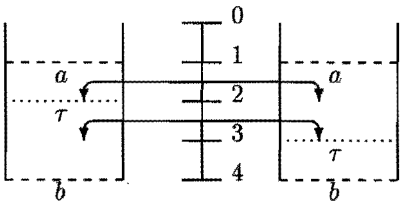
a (timed) τ-transition on one side may be matched with *an idling* *at the other side.*

In the sequel we will give examples of (timed) branching bisimilar process terms by giving their process diagrams and showing the crucial points in time at which the underlying transitions systems can be related by a branching bisimulation.

Example 6.2.1 *$p \equiv a(1) \cdot \tau(2) \cdot b(3)$ is branching bisimilar with $q \equiv a(1) \cdot b(3)$:*

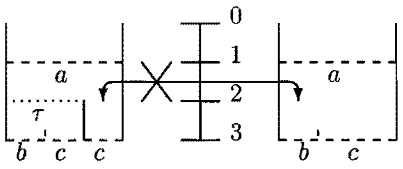


Example 6.2.2 *$a(1) \cdot \tau(2) \cdot b(4)$ is branching bisimilar with $a(1) \cdot \tau(3) \cdot b(4)$.*



Next, we give some examples of process terms which are distinguished by branching bisimulation.

Example 6.2.3 $a(1) \cdot (\tau(2) \cdot (b(3) + c(3)) + c(3))$ *is distinguished from* $a(1) \cdot (b(3) + c(3))$, *since in the first process term it may be the case that at 2 it is determined that the c will be executed at 3, while in the latter process term the choice between the b and the c at 3 can not be done earlier than* 3.



In timed branching bisimulation no "τ-stuttering" is allowed afterwards; we enforce that two bisimilar process terms terminate at the same points in time.

Example 6.2.4 $a(1)$ *differs from* $a(1) \cdot \tau(2)$ *as the* $a(1)$ *terminates succesfully at* 1 *and* $a(1) \cdot \tau(2)$ *at* 2. *Moreover,* $a(1) \cdot b(2)$ *is certainly not branching bisimilar with* $a(1) \cdot \tau(2) \cdot b(2)$ *since the latter process term has a deadlock.*

6.3 Branching Bisimulation in BPA$\rho\delta$

The examples have shown us that the idle steps play a vital role when we deal with branching bisimulation. Therefore, we will define *branching bisimulation equivalence* in the context of the idle semantics first.

Before giving the definition of branching bisimulation for these transition systems we have to define $< p, t > \Longrightarrow < q, r >$, which means that $< p, t >$ can evolve into $< q, t >$ by idle transitions and τ-transitions only. In the rest of this chapter κ and κ' denote arbitrary elements of $\{\tau, \iota\}$.

Definition 6.3.1
$\Longrightarrow \subseteq ((T^{cl}(\text{BPA}\rho\delta\tau\text{I}) \times \mathit{Time}) \times \mathit{Time} \times (T^{cl}(\text{BPA}\rho\delta\tau\text{I}) \times \mathit{Time})$
is defined as the least relation satisfying:

- $< p, t > \Longrightarrow < p, t >$
- *if* $< p, t > \Longrightarrow < q, r >$ *and* $< q, r > \xrightarrow{\kappa(r')} < q', r' >$ *then* $< p, t > \Longrightarrow < q', r' >$

Remember from the definition of (untimed) branching bisimulation that a transition $p \xrightarrow{a} p'$ (where $p \leftrightarroweq_b q$) can be matched with a series of transitions $q \Longrightarrow z \xrightarrow{a} q'$ such that $p \leftrightarroweq_b z$ and $p' \leftrightarroweq_b q'$. In the timed setting we have to consider idle transitions as well and we will identify idle transitions with τ transitions, as is shown in the previous examples.

Moreover, in the untimed setting the so called Stuttering Lemma (see [GW89]) holds, which states that if

$$q_0 \xrightarrow{\tau} q_1 \dots \xrightarrow{\tau} q_n$$

then

$$p \leftrightarrow_b q_0 \text{ and } p \leftrightarrow_b q_n \text{ imply } p \leftrightarrow_b q_i \text{ for } 0 \leq i \leq n$$

This Lemma is characteristic for branching bisimulation.

In the timed setting we start with a definition of branching bisimulation in which this stuttering property is put in the definition itself. In the sequel we simplify the definition, and we prove the Stuttering Lemma for this simplified definition for the the case where the states are process terms in $T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I})$. We do not start with this simplified definition right away as it is not clear whether we can prove the Stuttering Lemma for more general cases with for example recursion.

We need an auxiliary definition for expressing that all intermediate states along a sequence are related as well.

Definition 6.3.2 *$(< p,t > \xrightarrow{a(r)} < p',r >)\mathcal{R}(< q,t > \xRightarrow{a(r)} < q',r >)$ denotes that $< p',r > \mathcal{R} < q',r >$ and that there is a sequence*

$$< q_0, t_0 > \xrightarrow{\kappa_1(t_1)} < q_1, t_1 > \ \dots \ \xrightarrow{\kappa_n(t_n)} < q_n, t_n > \xrightarrow{a(r)} < q', r >$$

with $q_0 \equiv q$ and $t_0 = t$ such that for each $i \in \{0, \dots, n\}$ and $s \in [t_i, t_{i+1})$, where $t_{n+1} = r$, it holds that $< p,s > \mathcal{R} < q_i, s >$.

Similarly we define $(< p,t > \xrightarrow{\kappa(r)} < p',r >)\mathcal{R}(< q,t > \Longrightarrow < q',r >)$ and $(< p,t > \xrightarrow{a(r)} \surd)\mathcal{R}(< q,t > \xRightarrow{a(r)} \surd)$. Using these definitions we can define timed branching bisimulation.

Definition 6.3.3 (Idle Branching Bisimulation)
$\mathcal{R} \subset (T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I}) \times Time)^2$ is an idle branching bisimulation *if whenever $< p,t > \mathcal{R} < q,t >$ then*

1. *$< p,t > \xrightarrow{a(r)} < p',r >$ $(a \in A)$ implies that there is a q' such that $(< p,t > \xrightarrow{a(r)} < p',r >)\mathcal{R}(< q,t > \xRightarrow{a(r)} < q',r >)$.*

2. *$< p,t > \xrightarrow{\kappa(r)} < p',r >$ implies that there is q' such that $(< p,t > \xrightarrow{\kappa(r)} < p',r >)\mathcal{R}(< q,t > \Longrightarrow < q',r >)$.*

3. *$< p,t > \xrightarrow{a(r)} \surd$ $(a \in A_\tau)$ implies that $(< p,t > \xrightarrow{a(r)} \surd)\mathcal{R}(< q,t > \xRightarrow{a(r)} \surd)$*

4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

Note that $< p,t > \xrightarrow{a(r)} \surd$ implies that $a \in A_\tau$.

Definition 6.3.4 (Idle Branching Bis. Eq.)
$< p,t > \;\underline{\leftrightarrow}^{\iota}_{b}\; < q,t >$ iff there is an idle branching bisimulation $\mathcal{R}$ such that $< p,t > \mathcal{R} < q,t >$.

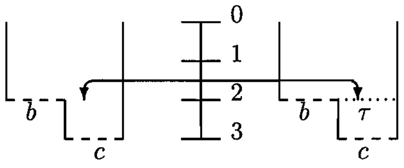
We define $\underline{\leftrightarrow}^{\iota}_{b}$ on $T^{cl}(\text{BPA}\rho\delta\tau\text{I})$ by requiring that $p \;\underline{\leftrightarrow}^{\iota}_{b}\; q$ iff forall $t \in \mathit{Time}$ it holds that $< p,t > \;\underline{\leftrightarrow}^{\iota}_{b}\; < q,t >$.

Proposition 6.3.5 *$\underline{\leftrightarrow}^{\iota}_{b}$ is an equivalence over $T^{cl}(\text{BPA}\rho\delta\tau\text{I})$.*

Proof. Omitted. □

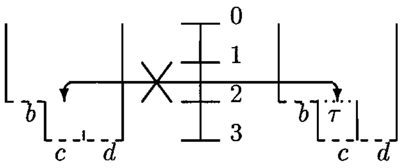
As expected, $\underline{\leftrightarrow}^{\iota}_{b}$ is not a congruence as is shown by the following two examples.

Example 6.3.6 $b(2) + c(3) \;\underline{\leftrightarrow}^{\iota}_{b}\; b(2) + \tau(2) \cdot c(3)$



On the other hand, we have

Example 6.3.7 $b(2) + c(3) + d(3) \;\not\underline{\leftrightarrow}^{\iota}_{b}\; b(2) + \tau(2) \cdot c(3) + d(3)$



Hence, $\underline{\leftrightarrow}^{\iota}_{b}$ is not a congruence, and we need a *rootedness* condition as in the untimed case.

Definition 6.3.8 ($< p,t >$-rooted) *$< p',r >$ is $< p,t >$*-rooted *if $p' \equiv p$ and $r = t$, or $< p,t > \xrightarrow{\iota(r)} < p',r >$.*

We define when $\mathcal{R}$ is rooted w.r.t. the pair of states $(< p,t >, < q,t >)$.

Definition 6.3.9 *An idle bisimulation $\mathcal{R}$ is rooted w.r.t. $< p,t >$ and $< q,t >$ if $< p,t > \mathcal{R} < q,t >$, and $< p',r > \mathcal{R} < q',r >$ implies that $< p',r >$ is $< p,t >$-rooted iff $< q',r >$ is $< q,t >$-rooted.*

Finally, we define *rooted idle branching bisimulation* equivalence, denoted by $\underline{\leftrightarrow}^{\iota}_{rb}$.

Definition 6.3.10 (Rooted Idle Branching Bis. Eq.)
$< p,t > \;\underline{\leftrightarrow}^{\iota}_{rb}\; < q,t >$ if there is an idle branching bisimulation $\mathcal{R}$ that is rooted w.r.t. $< p,t >$ and $< q,t >$.

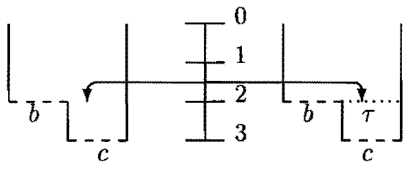
We obtain rooted idle branching bisimulation equivalence on process terms by putting $p \leftrightarrows^{\iota}_{rb} q$ iff forall $t \in \mathit{Time}$ we have $< p, t > \leftrightarrows^{\iota}_{rb} < q, t >$. Note, that we have $a(2) + b(3) \not\leftrightarrows^{\iota}_{rb} a(2) + \tau(2) \cdot b(3)$.

In Section 6.8 we motivate that $\leftrightarrows^{\iota}_{rb}$ is a congruence over BPA$\rho\delta\tau$, and in Section 6.7 we define (rooted) branching bisimulation equivalence in the context of the term semantics.

6.4 A Single Law for the Silent Step

A typical identity is given by the following example.

Example 6.4.1 $b(2) + c(3) \leftrightarrows^{\iota}_{b} b(2) + \tau(2) \cdot c(3)$



This example shows us that for some $t \in \mathit{Time}$ we have that

$$p \leftrightarrows^{\iota} p_0 + p_1 \leftrightarrows^{\iota}_{b} p_0 + \tau(t) \cdot p_1$$

where p_0 denotes that part of p which starts before or at t, and p_1 denotes that part of p which starts after t. Since in this case $t \gg p_0 \leftrightarrows^{\iota} \delta(t)$, we have as well $p \leftrightarrows^{\iota}_{b} p_0 + \tau(t) \cdot p$.

In order to express p_0 properly we introduce a variant of the operator $p \gg t$, that allows also actions of p *at* time t. We denote this operator by $p \ggeq t$, its defining axioms are given in Table 6.1, its action rules are left to the reader. In case $t < U(p)$ we have

$$p \leftrightarrows^{\iota} p \ggeq t + t \gg p \leftrightarrows_{b} p \ggeq t + \tau(t) \cdot p$$

Note that in case $t = U(p)$ we have

$r \leq t$	$a(r) \ggeq t$	$=$	$a(r)$
$r > t$	$a(r) \ggeq t$	$=$	$\delta(t)$
	$(p + q) \ggeq t$	$=$	$p \ggeq t + q \ggeq t$
	$(p \cdot q) \ggeq t$	$=$	$(p \ggeq t) \cdot q$

Table 6.1: Axioms for $\ggeq$ -operator

$$\begin{array}{lll} p \;\underline{\leftrightarrow}^{\iota}\; p+\delta(t) & \underline{\leftrightarrow}^{\iota} & p \gg t + t \gg p \\ \quad\; \not\underline{\leftrightarrow}^{\iota}_{b} & & \\ \quad\; p+\tau(t)\cdot\delta & \underline{\leftrightarrow}^{\iota} & p \gg t + \tau(t)\cdot p \end{array}$$

and if $t > U(p)$ we have

$$\begin{array}{lll} p & \underline{\leftrightarrow}^{\iota} & p \gg t \\ \not\underline{\leftrightarrow}^{\iota} & & \\ p \gg t + t \gg p & \underline{\leftrightarrow}^{\iota} & p+\delta(t) \end{array}$$

To obtain an identity for *rooted* branching bisimulation, we have to consider the above terms in a context $a(r)\cdot(\ldots)$. We have, for $t < U(p)$,

$$a(r)\cdot p \;\underline{\leftrightarrow}^{\iota}\; a(r)\cdot(p \gg t + t \gg p) \;\underline{\leftrightarrow}^{\iota}_{rb}\; a(r)\cdot(p \gg t + \tau(t)\cdot p),$$

if $r < t$, whereas for $r' \geq t$ we have

$$a(r')\cdot p \;\;\not\underline{\leftrightarrow}\;\; a(r')\cdot(p \gg t + \tau(t)\cdot p) \;\underline{\leftrightarrow}^{\iota}\; a(r)\cdot\delta$$

We can express this algebraically by the following real time τ-law:

$$\boxed{\mathrm{T1}_{\rho} \quad r < t < U(p) \quad a(r)\cdot p = a(r)\cdot(p \gg t + \tau(t)\cdot p)}$$

In Chapter 8 we explain the connection between this law and the untimed law T1 in more detail.

6.5 The Extension with Integration

6.5.1 A first generalization of the axiom T1$_{\rho}$

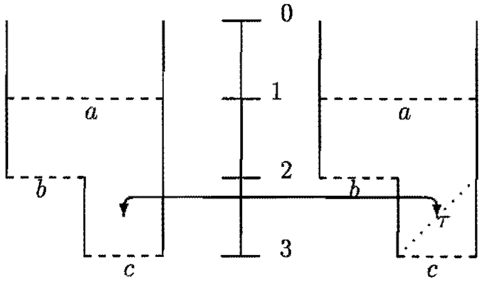
First we give a few examples.

Example 6.5.1 *In the calculus without integrals we have the following typical example*

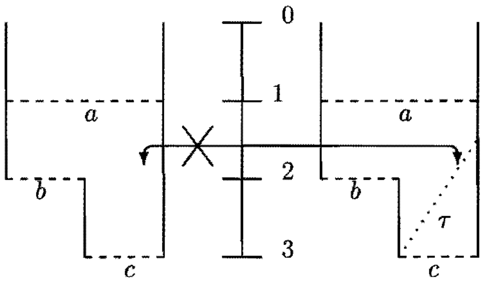
$$a(1)\cdot(b(2)+c(3)) \quad \underline{\leftrightarrow}_{rb} \quad a(1)\cdot(b(2)+\tau(2)\cdot c(3))$$

We can adapt this example to

$$a(1)\cdot(b(2)+c(3)) \;\underline{\leftrightarrow}_{b}\; a(1)\cdot\left(b(2)+\int_{v\in\langle 2,3\rangle}\tau(v)\cdot c(3)\right)$$



Example 6.5.2 *If we can make a choice for the* $c(3)$ *before time 2 (by allowing a* τ *before 2) this pair is not bisimilar anymore;*
$a(1) \cdot (b(2) + c(3)) \;\not\leftrightarrow_b\; a(1) \cdot (b(2) + \int_{v \in \langle 1.5, 3\rangle} \tau(v) \cdot c(3))$.



In the sequel we denote by $\int_\alpha(\tau(w)) \cdot p$ the term $(\int_\alpha(\tau(w)) \cdot p$, that is, we stress that p is not in the scope of the integral $\int_\alpha$.

The above examples suggest us to generalize $\mathrm{T1}_\rho$ to the setting of prefixed integration as follows.

$$\int_\alpha(a(v) \cdot p) \;=\; \int_\alpha(a(v) \cdot (p \gg b_0 + \int_{w \in \langle\!| b_0, b_1 |\!\rangle}(\tau(w)) \cdot p)$$

under the following conditions.

- The interval $\langle\!| b_0, b_1 |\!\rangle$ may not be empty, since otherwise $\int_{w \in \langle\!| b_0, b_1 |\!\rangle}(\tau(w)) \cdot p$ reduces to δ and $\int_\alpha(a(v) \cdot p)$ is in general not equal to $\int_\alpha(a(v) \cdot (p \gg b_0 + \delta))$.
- We must require that $v < b_1$, since otherwise there may be a deadlock on the righthand side after the execution of the a at some $t \geq b_1[t/v]$.
- We do not allow a deadlock after the execution of the τ in $\int_{w \in \langle\!| b_0, b_1 |\!\rangle}(\tau(w)) \cdot p)$, that is, we require that p is able to idle till b_1. This is expressed by the condition $U_{b_1}(p)$.

The above observations are summarised by the law $\mathrm{T1}_I$; the I in the name refers to *integration.*

$$\boxed{\begin{array}{l} \mathrm{T1}_I \quad \alpha \;=\; (\; \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \;\wedge\; v < b_1 \;\wedge\; U_{b_1}(p)\;) \\[1em] \qquad \int_{\alpha \wedge \beta}(a(v) \cdot p) \;=\; \int_{\alpha \wedge \beta}(a(v) \cdot (p \gg b_0 + \int_{w \in \langle\!| b_0, b_1 |\!\rangle}(\tau(w)) \cdot p)) \end{array}}$$

So, we have

Example 6.5.3
$a(1) \cdot \int_{v \in \langle 2,4\rangle} b(v) \;\leftrightarrow^{\iota}_{rb}\; a(1) \cdot (\int_{v \in \langle 2,3]} b(v) + \tau(3) \cdot \int_{v \in (3,4\rangle} b(v))$ *This identity can be derived as follows. Note, that we can apply* $\mathrm{T1}_I$ *with* $\langle\!| b_0, b_1 |\!\rangle = [3,3]$, *and* $p \equiv \int_{v \in \langle 2,4\rangle} b(v)$. *The condition*

$$U_{b_1}(p) \;=\; U_3(\int_{v\in\langle 2,4\rangle} b(v)) \;=\; (3<4)$$

reduces to tt. Hence,

$$v=1 \;\wedge\; (\,[3,3]\neq\emptyset \;\wedge\; v<3 \;\wedge\; U_3(\int_{v\in\langle 2,4\rangle} b(v))\,)$$

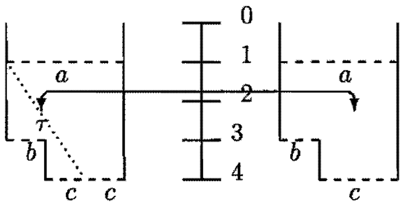
reduces to $v=1$. *We have the following derivation within* BPA$\rho\delta$I + B$_I$.

$$\begin{array}{ll}
\vdash & a(1)\cdot \int_{v\in\langle 2,4\rangle} b(v)\\
\overset{abb}{=} & \int_{v=1} a(v)\cdot \int_{v\in\langle 2,4\rangle} b(v)\\
\overset{AC}{=} & \int_{v=1\wedge[3,3]\neq\emptyset\wedge v<3\wedge U_3(\int_{v\in\langle 2,4\rangle} b(v))} a(v)\cdot \int_{v\in\langle 2,4\rangle} b(v)\\
\overset{T1\rho}{=} & a(1)\cdot(\int_{v\in\langle 2,4\rangle} b(v) \gg 3 \;+\; \tau(3)\cdot \int_{v\in\langle 2,4\rangle} b(v))\\
= & a(1)\cdot(\int_{v\in\langle 2,3]} b(v) \;+\; \tau(3)\cdot(3 \gg \int_{v\in\langle 2,4\rangle} b(v))\\
= & a(1)\cdot(\int_{v\in\langle 2,3]} b(v) \;+\; \tau(3)\cdot \int_{v\in\langle 3,4\rangle} b(v))
\end{array}$$

6.5.2 The Timed Branching Law

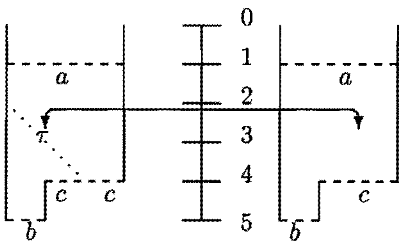
However, not all identities can be covered by T1$_I$.

Example 6.5.4
$a(1)\cdot(\int_{v\in\langle 1,4\rangle} \tau(v)\cdot(b(3)+c(4))+c(4)) \leftrightarrows^{\iota}_{\tau b} a(1)\cdot(b(3)+c(4))$



and

Example 6.5.5
$a(1)\cdot(\int_{v\in\langle 2,4\rangle} \tau(v)\cdot(b(5)+c(4))+c(4)) \leftrightarrows^{\iota}_{\tau b} a(1)\cdot(b(5)+c(4))$

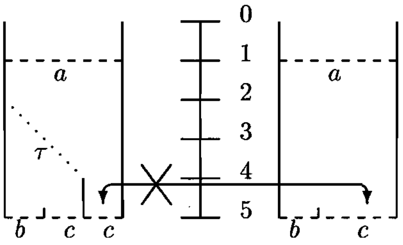


These identities look like an instance of the (untimed) second branching τ-law:

B2 $z \cdot (\tau \cdot (p+q)+p) = z \cdot (p+q)$

When adding time to this law we have to be very careful with the conditions on all the intervals as is shown in the following example. The process terms in this example differ only with the process terms in the previous example, in that respect that in the process term on the left hand side *both* components $b(5)$ and $c(5)$ can idle after 4, that is the upperbound of $\langle 2,4\rangle$.

Example 6.5.6
$a(1) \cdot (\int_{v\in\langle 2,4\rangle} \tau(v) \cdot (b(5)+c(5)) + c(5)) \not\leftrightarrow^{\iota}_{b} a(1) \cdot (b(5)+c(5))$



Consider the next two examples.

Example 6.5.7
The transition system of the left hand side of the second pair has a deadlock at 4 caused by the possible execution of the τ at 4. The transition system of the right hand side of the first pair does not have such a deadlock.

$$a(1) \cdot (\int_{v\in\langle 2,4\rangle} \tau(v) \cdot (b(3)+c(4)) + c(4)) \;\leftrightarrow^{\iota}_{rb}\; a(1) \cdot (b(3)+c(4))$$
$$a(1) \cdot (\int_{v\in\langle 2,4]} \tau(v) \cdot (b(3)+c(4)) + c(4)) \;\not\leftrightarrow^{\iota}_{rb}\; a(1) \cdot (b(3)+c(4))$$

Example 6.5.8 *Consider the following two pairs, of which the first is a bisimilar one. In the transition system of the left hand side of the second pair, the choice for doing the b might be done at 10, while at the right hand side it may be postponed until 11.*

$$a(1) \cdot (\int_{v\in\langle 1,10\rangle} \tau(v) \cdot (\int_{w\in\langle 1,20]} b(w) + \int_{z\in\langle 0,10\rangle} c(z)) + \int_{w\in\langle 1,20]} b(w))$$
$$\leftrightarrow^{\iota}_{rb}$$
$$a(1) \cdot (\int_{w\in\langle 1,20]} b(w) + \int_{z\in\langle 0,10\rangle} c(z))$$

$$a(1) \cdot (\int_{v\in\langle 1,10\rangle} \tau(v) \cdot (\int_{w\in\langle 1,20]} b(w) + \int_{z\in\langle 0,11\rangle} c(z)) + \int_{w\in\langle 1,20]} b(w))$$
$$\not\leftrightarrow^{\iota}_{rb}$$
$$a(1) \cdot (\int_{w\in\langle 1,20]} b(w) + \int_{z\in\langle 0,11\rangle} c(z))$$

The previous examples and the discussion of $\mathrm{T1}_I$ show us that

$$\int_{\alpha}(a(v) \cdot ((\int_{w\in\{b_0,b_1\}}(\tau(w)) \cdot (p+q)+p))$$

is rooted branching bisimilar with

$$\int_{\alpha}(a(v) \cdot (p + b_0 \gg q))$$

under the conditions that

- The interval $\langle\!| b_0, b_1 |\!\rangle$ is not empty.
- After execution of the a not all τ's are deadlocked, that is, $v < b_1$.
- At b_1 the choice for p or q is determined in the process term $p + q$. That is, one of the two summands cannot idle till b_1, while the other summand can. More formally, either $U_{b_1}(p) \wedge \neg(U_{b_1}(q))$ (p can idle until b_1, and q cannot), or $\neg(U_{b_1}(p)) \wedge U_{b_1}(q)$ (p cannot idle untill b_1, while q can).

And finally we obtain the timed branching law B_I, as given in Table 6.2.

$$\boxed{\begin{array}{l} \mathrm{B}_I \quad \alpha \;=\; (\; \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \;\wedge\; v < b_1 \;\wedge \\ \qquad\qquad ((U_{b_1}(p) \wedge \neg(U_{b_1}(q))) \vee (\neg(U_{b_1}(p)) \wedge U_{b_1}(q)))\;) \\ \\ \quad \int_{\alpha\wedge\beta}(a(v) \cdot (\int_{w\in\langle\!| b_0,b_1 |\!\rangle}(\tau(w)) \cdot (p+q) + p)) \;=\; \\ \quad \int_{\alpha\wedge\beta}(a(v) \cdot (p + b_0 \gg q)) \end{array}}$$

Table 6.2: The timed branching law

Unfortunately, the condition α of the law B_I is rather complicated. We can make a case distinction, and give for each case a simpler version of B_I, see Table 6.3. We have the following Theorem.

Theorem 6.5.9 (Soundness) $p, q \in T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I})$

$$\mathrm{BPA}\rho\delta\tau\mathrm{I} + \mathrm{B}_I \vdash p = q \quad\Longrightarrow\quad p \leftrightarrows_{rb} q$$

Proof. Omitted □

In the next chapter we prove the completeness of BPA$\rho\delta$I+ B_I w.r.t. $\leftrightarrows_{rb}$.

6.6 The Embedding of BPA$\delta\tau$ into BPA$\rho\delta\tau$I

Baeten & Bergstra have given an embedding of BPAδ into BPA$\rho\delta$ which interprets every (untimed) atomic action a as $\int_{tt} a(v)$. Let us denote this translation by

$$RT : T(\mathrm{BPA}\delta\tau) \longrightarrow T(\mathrm{BPA}\rho\delta\tau\mathrm{I}).$$

$$\mathrm{B}_I^a \quad \alpha \ = \ (\ \{\!|b_0, b_1|\!\} \neq \emptyset \ \wedge \ v < b_1 \ \wedge U_{b_1}(p)\ )$$

$$\textstyle\int_{\alpha\wedge\beta}(a(v) \cdot (\int_{w\in\{\!|b_0,b_1|\!\}}(\tau(w)) \cdot (p + q \gg b_1) + p)) \ = \ \int_{\alpha\wedge\beta}(a(v) \cdot (p + b_0 \gg q \gg b_1))$$

$$\mathrm{B}_I^b \quad \alpha \ = \ (\ \{\!|b_0, b_1|\!\} \neq \emptyset \ \wedge \ v < b_1 \ \wedge U_{b_1}(q)\ )$$

$$\textstyle\int_{\alpha\wedge\beta}(a(v) \cdot (\int_{w\in\{\!|b_0,b_1|\!\}}(\tau(w)) \cdot (p \gg b_1 + q) + p \gg b_1)) \ = \ \int_{\alpha\wedge\beta}(a(v) \cdot (p \gg b_1 + b_0 \gg q))$$

Table 6.3: Simpler versions of timed branching law

We have

$$\forall p, q \in T(\mathrm{BPA}\delta) \qquad p \mathrel{\underline{\leftrightarrow}} q \quad \Longleftrightarrow \quad RT(p) \mathrel{\underline{\leftrightarrow}}^{\iota} RT(q).$$

However, this does not hold any more after adding the τ and working with branching bisimulation equivalence. For example, in the untimed case we have $a \cdot \tau \mathrel{\underline{\leftrightarrow}}_{rb} a$ but

$$\int_{tt} a(v) \cdot \int_{tt} \tau(v) \quad \not\mathrel{\underline{\leftrightarrow}}^{\iota}_{rb} \quad \int_{tt} a(v)$$

This is caused by a difference in the design decisions of BPAδ respectively BPA$\rho\delta$I. In BPAδ the intuition is that the execution of an action may take some time, see for example [Gla87]. This is exactly the motivation behind the first τ law $p \cdot \tau = p$. But in real time process algebra $\int_{\alpha} a(v)$ executes an a at some point r such that $\models \alpha[r/v]$ and terminates successfully at r. A possible embedding of BPA$\delta\tau$ into BPA$\rho\delta\tau$I is given by interpreting an untimed atomic action a as

$$\int_{tt} a(v) \cdot \int_{tt} \tau(v),$$

expressing that the action a is executed somewhere in time after which it terminates some time later. This translation has been pointed out by Jos Baeten ([Bae92]). Let us denote this translation by RT_{τ}, then we have the following Proposition.

Proposition 6.6.1 $p, q \in T(\mathrm{BPA}\delta\tau)$

$$p \mathrel{\underline{\leftrightarrow}}_{rb} q \quad \Longleftrightarrow \quad RT_{\tau}(p) \mathrel{\underline{\leftrightarrow}}^{\iota}_{rb} RT_{\tau}(q)$$

Proof. Omitted. □

6.7 Branching Bis. in a term semantics

In this Section we focuss on $T^{cl}(\text{BPA}\rho\delta\tau\text{I})$ and we give several alternative characterizations of idle branching bisimulation, by which we obtain, step by step, branching bisimulation in a term semantics.

A first simplification is to weaken the requirement that *all* intermediate states are related. This is done in the next definition of *simple* idle branching bisimilar.

Definition 6.7.1 (Simple Idle Branching Bisimulation)
$\mathcal{R} \subset (T^{cl}(\text{BPA}\rho\delta\tau\text{I}) \times \textit{Time})^2$ *is a* simple idle branching bisimulation *if whenever* $< p, t > \mathcal{R} < q, t >$ *then*

1. $< p, t > \xrightarrow{a(r)} < p', r >$ *(*$a \in A$*) implies that there are* z, q' *and* s *such that*
   - $< q, t > \Longrightarrow < z, s > \xrightarrow{a(r)} < q', r >$*, and*
   - $< p, s > \mathcal{R} < z, s >$ *and* $< p', r > \mathcal{R} < q', r >$.
2. $< p, t > \xrightarrow{\kappa(r)} < p', r >$ *implies that there are* z, q', κ' *and* s *such that*
   - $< q, t > \Longrightarrow < z, s > \xrightarrow{\kappa'(r)} < q', r >$*, and*
   - $< p, s > \mathcal{R} < z, s >$ *and* $< p', r > \mathcal{R} < q', r >$.
3. $< p, t > \xrightarrow{a(r)} \surd$ *(*$a \in A_\tau$*) implies that there are* z *and* s *such that*
   - $< q, t > \Longrightarrow < z, s > \xrightarrow{a(r)} \surd$*, and*
   - $< p, s > \mathcal{R} < z, s >$.
4. *Respectively (1), (2) and (3) with the role of* p *and* q *interchanged.*

For a simple idle branching bisimulation $\mathcal{R}$ we can prove a timed version of the Stuttering Lemma [GW89].

Lemma 6.7.2 (Stuttering Lemma for finite process terms)
If $p, q \in T^{cl}(\text{BPA}\rho\delta\tau\text{I})$ *and*

$$< p, t > \xrightarrow{\iota(r)} < p, r > \text{ and } < q, t > \Longrightarrow < q', r >,$$

and $\mathcal{R}$ *is a simple idle branching bisimulation such that*

$$< p, t > \mathcal{R} < q, t > \text{ and } < p, r > \mathcal{R} < q', r >$$

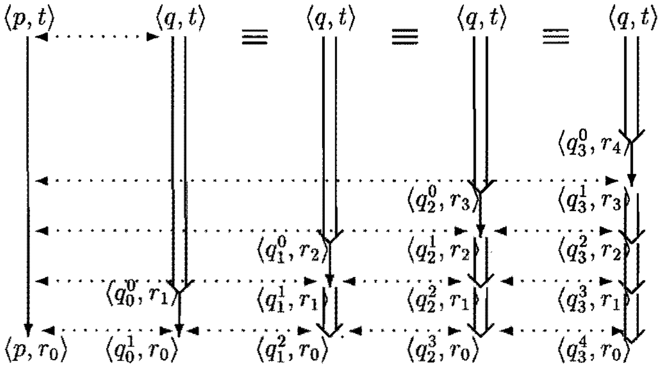
then also $(< p, t > \xrightarrow{\iota(r)} < p, r >)\mathcal{R}(< q, t > \Longrightarrow < q', r >)$.

Proof. We give give only a sketch of the proof.

Take $r_0 = r$. For $< p, t > \xrightarrow{\iota(r_0)} < p, r_0 >$ there are q_0^0 and q_0^1 and there is a r_1 such that $< q, t > \Longrightarrow < q_0^0, r_1 >$ and $< q_0^0, r_1 > \xrightarrow{\kappa(r_0)} < q_0^1, r_0 >$ and $< p, r_0 > \mathcal{R} < q_0^1, r_0 >$.

Since $r_1 < r_0$ we have $< p, t > \xrightarrow{\iota(r_1)} < p, r_1 >$. Hence, there are q_1^0, q_1^1 and r_2 such that $< q, t > \Longrightarrow < q_1^0, r_2 >$ and $< q_1^0, r_2 > \xrightarrow{\kappa(r_1)} < q_1^1, r_1 >$ and $< p, r_1 > \mathcal{R} < q_1^1, r_1 >$. Furthermore, since $< p, r_1 > \xrightarrow{\iota(r_0)} < p, r_0 >$ there is also a $< q_1^2, r_0 >$ such that $< q_1^1, r_1 > \Longrightarrow < q_1^2, r_0 >$ and $< p, r_0 > \mathcal{R} < q_1^2, r_0 >$

If we repeat this argument we get the picture below. Since q is a finite process term, that is a process term without recursion, this argument cannot be repeated infinitely many times and we are done



$\langle p,t\rangle$
$\langle q,t\rangle$
$\equiv$
$\langle q,t\rangle$
$\equiv$
$\langle q,t\rangle$
$\equiv$
$\langle q,t\rangle$
$\langle q_3^0, r_4\rangle$
$\langle q_2^0, r_3\rangle$
$\langle q_3^1, r_3\rangle$
$\langle q_1^0, r_2\rangle$
$\langle q_2^1, r_2\rangle$
$\langle q_3^2, r_2\rangle$
$\langle q_0^0, r_1\rangle$
$\langle q_1^1, r_1\rangle$
$\langle q_2^2, r_1\rangle$
$\langle q_3^3, r_1\rangle$
$\langle p, r_0\rangle$
$\langle q_0^1, r_0\rangle$
$\langle q_1^2, r_0\rangle$
$\langle q_2^3, r_0\rangle$
$\langle q_3^4, r_0\rangle$

□

We denote simple idle branching bisimulation by $\underline{\leftrightarrow}_b^{\iota\text{-}simple}$,

Lemma 6.7.3 $p, q \in T^{cl}(\text{BPA}\rho\delta\tau\text{I})$

$$< p, t > \underline{\leftrightarrow}_b^{\iota\text{-}simple} < q, t > \quad \Longleftrightarrow \quad < p, t > \underline{\leftrightarrow}_b^{\iota} < q, t >$$

Proof.

$\Longrightarrow$ It follows from the Stuttering Lemma.

$\Longleftarrow$ Trivial. □

The next change is, that we consider only sequences of τ-transitions in the premises of the clauses of the definition of a branching bisimulation. We have that $< z_0, t_0 > \xrightarrow{\iota(t_1)} < z_1, t_1 > \xrightarrow{a(t_2)} < z_2, t_2 >$ implies $< z_0, t_0 > \xrightarrow{a(t_2)} < z_2, t_2 >$ as well. Hence if $< q,t > \Longrightarrow < z,s > \xrightarrow{a(r)} < q',r >$, then we can find $< z',s' >$ such that $< q,t > \overset{\tau *}{\Longrightarrow} < z',s' > \xrightarrow{a(r)} < q',r >$ as well.

Here, $\overset{\tau *}{\Longrightarrow}$ denotes that there are only τ-transitions allowed along the sequence.

Definition 6.7.4 (Semi Idle Branching Bisimulation)
$\mathcal{R} \subseteq (T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I}) \times \mathit{Time})^2$ *is a* semi idle branching bisimulation *if whenever* $< p,t > \mathcal{R} < q,t >$ *then*

1. $< p,t > \xrightarrow{a(r)} < p',r >$ *(*$a \in A$*) implies that there are* z, q' *and* s *such that*
   - $< q,t > \overset{\tau *}{\Longrightarrow} < z,s > \xrightarrow{a(r)} < q',r >$,
   - $< p,t > \mathcal{R} < z,s >$ *and* $< p',r > \mathcal{R} < q',r >$.
2. $< p,t > \xrightarrow{\kappa(r)} < p',r >$ *implies that there are* z, q', κ' *and* s *such that*
   - $< q,t > \overset{\tau *}{\Longrightarrow} < z,s > \xrightarrow{\kappa'(r)} < q',r >$,
   - $< p,t > \mathcal{R} < z,s >$ *and* $< p',r > \mathcal{R} < q',r >$.
3. $< p,t > \xrightarrow{a(r)} \surd$ *(*$a \in A_\tau$*) implies that there are* z *and* s *such that*
   - $< q,t > \overset{\tau *}{\Longrightarrow} < z,s > \xrightarrow{a(r)} \surd$,
   - $< p,t > \mathcal{R} < z,s >$.
4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

We denote the resulting equivalence by $\underline{\leftrightarrow}_b^{\iota\text{-}semi}$. And we have the following Lemma:

Lemma 6.7.5 $p, q \in T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I})$, $t \in \mathit{Time}$

$$< p,t > \underline{\leftrightarrow}_b^{\iota\text{-}simple} < q,t > \iff < p,t > \underline{\leftrightarrow}_b^{\iota\text{-}semi} < q,t >$$

Proof. Omitted. □

Next, we give a similar definition in the context of the term semantics. Before doing that, we need a way to express idle transitions in term semantics.

Definition 6.7.6 (shift$_r$)

$$\begin{array}{lcll}
\mathit{shift}_r(a(t)) & = & r \gg a(t) & \\
\mathit{shift}_r(p+q) & = & r \gg (p+q) & \\
\mathit{shift}_r(p \cdot q) & = & \mathit{shift}_r(p) \cdot q & \\
\mathit{shift}_r(t \gg p) & = & r \gg p & \mathit{if}\ r > t \\
 & & t \gg p & \mathit{otherwise}
\end{array}$$

For technical reasons we take $\mathit{shift}_{-\infty}(p) = p$ and $U_{-\infty}(p)$ for every p.

We can express an idle transition in the term semantics as follows

$$p \xrightarrow{\iota(r)} p' \quad \text{abbreviates} \quad U_r(p) \wedge p' \equiv shift_r(p)$$

We redefine $\overset{\tau*}{\Longrightarrow}$, such that is defined on pairs of process terms. Since the course of time is not obvious anymore from the states, we add it to the label of $\Longrightarrow$. As base case we have $p \overset{\tau*}{\Longrightarrow}_{time(p)} p$, and $p \overset{\tau*}{\Longrightarrow}_t q$ with $q \xrightarrow{\tau(r)} q'$ implies $p \overset{\tau*}{\Longrightarrow}_r q'$. For the definition of $time(p)$ we refer to Definition 2.6.4. We have the following proposition, where we take $t \geq -\infty$ for all t.

Proposition 6.7.7 $p \in T^{cl}(\text{BPA}\rho\delta\text{I})$

$$\begin{aligned} & \exists p' \; p \xrightarrow{\iota(r)} p' \\ \Longleftrightarrow \;& \forall t \geq time(p) \quad < strip(p), t > \xrightarrow{\iota(r)} < strip(p), r > \\ \Longleftrightarrow \;& U_r(p) \end{aligned}$$

Proof. Omitted. □

And finally we define term branching bisimulation.

Definition 6.7.8 (Term Branching Bisimulation)
$\mathcal{R} \subseteq T^{cl}(\text{BPA}\rho\delta\tau\text{I})^2$ *is a* term branching bisimulation *if whenever $p\mathcal{R}q$ then*

1. *$p \xrightarrow{a(r)} p'$ $(a \in A)$ implies that there are z, q' and s such that*
   - *$q \overset{\tau*}{\Longrightarrow}_s z \xrightarrow{a(r)} q'$,*
   - *$shift_s(p)\mathcal{R}z$ and $p'\mathcal{R}q'$.*
2. *$p \xrightarrow{\kappa(r)} p'$ implies that there are z, q', κ' and s such that*
   - *$q \overset{\tau*}{\Longrightarrow}_s z \xrightarrow{\kappa'(r)} q'$,*
   - *$shift_s(p)\mathcal{R}z$ and $p'\mathcal{R}q'$.*
3. *$p \xrightarrow{a(r)} \surd$ $(a \in A_\tau)$ implies that there are z and s such that*
   - *$q \overset{\tau*}{\Longrightarrow}_s z \xrightarrow{a(r)} \surd$,*
   - *$shift_s(p)\mathcal{R}z$.*
4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

We denote the resulting equivalence by $\leftrightarrows_b$.

Definition 6.7.9 (p-rooted) *p' is p-rooted, if $p' \equiv p$ or there is an r such that $p \xrightarrow{\iota(r)} p'$.*

Definition 6.7.10 *A term bisimulation $\mathcal{R}$ is rooted w.r.t. to (p, q) if $p\mathcal{R}q$, and $p'\mathcal{R}q'$ implies that p' is p-rooted iff q' is q-rooted.*

Definition 6.7.11 *$p \leftrightarroweq_{rb} q$ if there is a term branching bisimulation $\mathcal{R}$, that is rooted w.r.t. (p, q).*

And finally we have the following proposition.

Proposition 6.7.12 $p, q \in T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I})$

$$p \leftrightarroweq_b q \quad \Longleftrightarrow \quad \forall t \quad < p, t > \ \leftrightarroweq_b^{\iota\text{-}semi} \ < q, t >$$

Proof. Omitted. □

Corollary 6.7.13 $p, q \in T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I})$

$$p \leftrightarroweq_b q \quad \Longleftrightarrow \quad p \leftrightarroweq_b^{\iota} q$$

6.8 Rooted Branch. Bis. Eq. is a Congruence

First we give a timed version of *strongly rooted* branching bisimulation equivalence, by requiring that it behaves from the roots as if it were strong bisimulation. As soon as left and right an action in A_τ has been executed the definition of (unrooted) branching bisimulation is applied.

Definition 6.8.1 (Strongly Rooted Idle Branching Bis. Eq.)
$p \leftrightarroweq_{srb} q$ iff

1. *$p \xrightarrow{a(r)} p'$ with $a \in A_\tau$ implies that there is a q' such that $q \xrightarrow{a(r)} q'$ and $p' \leftrightarroweq_b^{\iota} q'$.*
2. *$p \xrightarrow{\iota(r)} p'$ implies that there is a q' such that $q \xrightarrow{\iota(r)} q'$ and $p' \leftrightarroweq_{srb} q'$.*
3. *$p \xrightarrow{a(r)} \surd$ implies that $q \xrightarrow{a(r)} \surd$.*
4. *Respectively (1),(2) and (3) with the role of p and q interchanged.*

As in the untimed case the definition of strongly rootedness is not stronger than the one of rootedness, in the case of branching bisimulation.

Proposition 6.8.2 $p, q \in T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I})$

$$p \leftrightarroweq_{srb} q \quad \Longleftrightarrow \quad p \leftrightarroweq_{rb} q$$

Proof. Omitted. □

Theorem 6.8.3 *$\leftrightarroweq_{rb}$ is a congruence over $T^{cl}(\mathrm{BPA}\rho\delta\tau\mathrm{I})$.*

Proof. It follows from 6.3.5 and 6.7.13 that $\leftrightarroweq_b$ is an equivalence.

It is easy to show that $\leftrightarroweq_{srb}$ is a congruence, and hence, $\leftrightarroweq_{rb}$ is a congruence as well. □

6.9 Some Additional Notations

If we give a diagram like

$$\begin{tikzcd}
p \arrow[r, "a(r)"] & p' \\
\underline{\leftrightarrow}_b & \underline{\leftrightarrow}_b \\
q \arrow[r, "a(r)"] & q'
\end{tikzcd}$$

we mean that $p \xrightarrow{a(r)} p'$, $q \xrightarrow{a(r)} q'$, $p \underline{\leftrightarrow}_b q$ and $p' \underline{\leftrightarrow}_b q'$ and a diagram like

$$\begin{tikzcd}
p \arrow[r, "\iota(r)"] & p' \\
\underline{\leftrightarrow}_b & \underline{\leftrightarrow}_b \\
q \arrow[r, Rightarrow, "\tau*"] & {}_r\, q'
\end{tikzcd}$$

means that $p \xrightarrow{\iota(r)} p'$, $q \overset{\tau*}{\Longrightarrow}_r q'$ such that $p \underline{\leftrightarrow}_b q$ and $p' \underline{\leftrightarrow}_b q'$.

The first clause of the Definition 6.7.8 can be expressed as well in the following way

$p \xrightarrow{a(r)} p'$ $(a \in A)$ implies that there are z, q' and s such that

$$\begin{tikzcd}
p \arrow[rr, "a(r)"] & & p' \\
\underline{\leftrightarrow}_b & & \\
p \arrow[r, "\iota(s)"] & s \gg p & \underline{\leftrightarrow}_b \\
\underline{\leftrightarrow}_b & \underline{\leftrightarrow}_b & \\
q \arrow[r, Rightarrow, "\tau*"'] & z \arrow[r, "a(r)"] & q'
\end{tikzcd}$$

Finally, we define *term branching bisimulation inclusion*, which will be used in the proof of the main theorem of the next chapter.

Definition 6.9.1 (Term Branching Bisimulation Inclusion)
$p \overset{\subset}{\leftrightarrow}_b q$ whenever

1. *$p \xrightarrow{a(r)} p'$ $(a \in A)$ implies that there are z, q' and s such that*
   - *$q \overset{\tau*}{\Longrightarrow}_s z \xrightarrow{a(r)} q'$,*
   - *$shift_s(p) \overset{\subset}{\leftrightarrow}_b z$ and $p' \underline{\leftrightarrow}_b q'$.*
2. *$p \xrightarrow{\kappa(r)} p'$ implies that there are z, q', κ' and s such that*
   - *$q \overset{\tau*}{\Longrightarrow}_s z \xrightarrow{\kappa'(r)} q'$,*
   - *$shift_s(p) \overset{\subset}{\leftrightarrow}_b z$ and $p' \underline{\leftrightarrow}_b q'$.*
3. *$p \xrightarrow{a(r)} \surd$ $(a \in A_\tau)$ implies that there are z and s such that*

- $q \overset{\tau *}{\Longrightarrow}_s z \xrightarrow{a(r)} \surd$,
- $shift_s(p) \;\underline{\leftrightarrow}_b\; z$.

Note that the definition does not have a "symmetric" part. $\underline{\leftrightarrow}_b$ is the semantic equivalent of summand inclusion in BPA$\rho\delta$I+ B$_I$.

7

Completeness for Branching Bisimulation

7.1 Introduction

In this chapter we prove that the axiom B_I, together with the axiom system BPA$\rho\delta$I, axiomatizes $\underline{\leftrightarrow}_{rb}$ completely.

We define a rewriting that constructs for each process term its *branching basic term.* The main theorem of this chapter is that if two branching basic terms are branching bisimilar, then they are strongly bisimilar as well. From this result the completeness result for branching bisimulation equivalence is reduced to the completeness problem for strong bisimulation equivalence which we have tackled already.

7.2 An Intermezzo on Time Variables

In this chapter we consider prefix normal forms only. We recall that we omit the binding brackets in these terms, i.e. we write $\int_\alpha a(v) \cdot p$ for $\int_\alpha (a(v) \cdot p)$.

In this chapter we rewrite a term like $a(1) \cdot \int_{v\in\langle 2,3\rangle} \tau(v) \cdot \int_{w\in\langle v,5\rangle} b(w)$ to $a(1) \cdot \int_{w\in\langle 2,5\rangle} b(w)$. The difficulty with this rewriting is that in this example the time variable v, of the integral that is to be removed, occurs later on in the term.

Example 7.2.1

$$\text{BPA}\rho\delta\text{I} \vdash a(1) \cdot \int_{v\in\langle 2,3\rangle} \tau(v) \cdot \int_{w\in\langle v,5\rangle} b(w) = a(1) \cdot \int_{w\in\langle 2,5\rangle} b(w)$$

We will give a derivation which starts with $a(1) \cdot \int_{w\in\langle 2,5\rangle} b(w)$

$$\begin{array}{cl}
 & a(1) \cdot \int_{w \in \langle 2,5 \rangle} b(w) \\
\overset{\mathrm{B}_I}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot \int_{w \in \langle 2,5 \rangle} b(w) \\
\overset{\mathrm{RT1}_I}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot (v \gg \int_{w \in \langle 2,5 \rangle} b(w)) \\
\overset{\mathrm{RT2}_I}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot \int_{v \in \langle 2,5 \rangle \wedge v < w} b(w) \\
\overset{\mathrm{C3}}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot (2 < v :\to \int_{w \in \langle 2,5 \rangle \wedge v < w} b(w)) \\
\overset{\mathrm{C2}}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot (2 < v :\to \int_{w \in \langle 2,5 \rangle \wedge 2 < v < w} b(w))
\end{array}$$

The expression $w \in \langle 2,5 \rangle \wedge 2 < v < w$ abbreviates $2 < v \wedge v < w \wedge 2 < w \wedge w < 5$, which has a subcondition $2 < v \wedge v < w \wedge 2 < w$ that can be reduced to $2 < v \wedge v < w$. By which we obtain $2 < v \wedge v < w \wedge w < 5$. We continue.

$$\begin{array}{cl}
\overset{\mathrm{CA}}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot (2 < v :\to \int_{2 < v \wedge v < w \wedge w < 5} b(w)) \\
\overset{\mathrm{C2}}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot (2 < v :\to \int_{v < w \wedge w < 5} b(w)) \\
\overset{\mathrm{C3}}{=} & a(1) \cdot \int_{v \in \langle 2,3 \rangle} \tau(v) \cdot \int_{w \in \langle v,5 \rangle} b(w)
\end{array}$$

□

We will use the following identity

$$\int_{v \in V} a(v) \cdot p = \int_{v \in V} a(v) \cdot p[\inf(V)/v]$$

which is derivable if the variable v occurs only as lower bound of initial integrals of p. To formalize this we define $fv^*(p) \subseteq fv(p)$ where p is supposed to be a prefix normal form that contains only bounds in normal form, with that respect, that $v+0$ is rewritten to v.

$$\begin{array}{rll}
 & fv^*(\int_{v \in \{w,b'\}} a(v)) & = var(b') \\
b \notin TVar & fv^*(\int_{v \in \{b,b'\}} a(v)) & = var(b+b') \\
 & fv^*(\int_{v \in \{w,b'\}} a(v) \cdot p) & = var(b') \cup fv(p) \backslash \{v\} \\
b \notin TVar & fv^*(\int_{v \in \{b,b'\}} a(v) \cdot p) & = var(b+b') \cup fv(p) \backslash \{v\} \\
 & fv^*(p+q) & = fv^*(p) + fv^*(q)
\end{array}$$

Proposition 7.2.2 *Let p be a prefix normal form and b a bound such that $v \notin fv^*(p) \cup var(b)$ then*

$$\mathrm{BPA}\rho\delta\mathrm{I} \vdash \int_{\alpha \wedge b \leq v} a(v) \cdot p = \int_{\alpha \wedge b \leq v} a(v) \cdot p[b/v]$$

Proof. Omitted, the proof is analogous to the derivation in Example 7.2.1. □

A similar, but semantic, proposition is the following.

Proposition 7.2.3 *Let p be a prefix normal form where $v \notin fv^*(p)$ and $r_1, r_2 \leq s$ then*

$$s \gg p[r_1/v] \leftrightarrows s \gg p[r_2/v]$$

Proof. Omitted. □

7.3 Branching Basic Terms

In this section we will present some rewrite rules by which each prefix normal form can be rewritten to its so called *branching basic term.* These terms are constructed such that if they are branching bisimilar, then they are strongly bisimilar as well. To give the reader an idea we will give several process terms and their branching basic terms.

Example 7.3.1

$$\begin{array}{lcl} a(1)\cdot\tau(2)\cdot b(3) & \longrightarrow & a(1)\cdot b(3) \\ a(1)\cdot(\int_{v\in\langle 1,2\rangle}\tau(v)\cdot b(2)+b(2)) & \longrightarrow & a(1)\cdot b(2) \\ a(1)\cdot(\int_{v\in\langle 1,2]}b(v)+\tau(2)\cdot\int_{v\in\langle 2,3\rangle}b(v)) & \longrightarrow & a(1)\cdot\int_{v\in\langle 1,3\rangle}b(v) \end{array}$$

7.3.1 Introducing τ's for each moment of choice

In order to rewrite each process term into a canonical form we have to rewrite all three process terms below into the same form.

Example 7.3.2 BPA$\rho\delta$I + B$_I$ $\vdash$

$$\begin{array}{rlllll} & a(1)\cdot & (\ \tau(2)\cdot & b(3) & + & c(3)) \\ = & a(1)\cdot & (\ \tau(2)\cdot & b(3) & +\ \tau(2)\cdot & c(3)) \\ = & a(1)\cdot & (& b(3) & +\ \tau(2)\cdot & c(3)) \end{array}$$

This example shows us that it is not possible to have as few as possible τ's in a process term. Therefore we add a τ for each moment of choice; in the above example both outermost process terms are rewritten to the process term in the middle.

In case of (non trivial) prefixed integration we have to do some more work. If we consider the process term p where

$$p \simeq \int_{v\in\langle 0,5\rangle} a(v) + \int_{v\in[2,6\rangle} b(v)$$

then we can split $\langle 0,6\rangle$ into $\langle 0,2]$, $\langle 2,5]$ and $\langle 5,6\rangle$, such that the potential of p does not change by idling within one of the resulting intervals.

In order to reflect these intervals of potential at the syntactical level we introduce a τ for each moment of choice. A moment of choice is the upper or lower bound of one of these intervals. (Only for the time stamp 6 we do not add a τ.) Furthermore, we will have only intervals of the form $\langle b,b'\rangle$; a summand of the form $\int_{v\in\langle b,b']}P(v)$ is rewritten to $\int_{v\in\langle b,b'\rangle}P(v)+P(b')$.

Example 7.3.3 *$p\simeq\int_{v\in\langle 0,5\rangle}a(v)+\int_{v\in[2,6\rangle}b(v)$ will be rewritten to*

$$\begin{array}{rl} & \int_{v\in\langle 0,2\rangle}a(v)+a(2)+b(2) \\ + & \tau(2)\cdot(\int_{v\in\langle 2,5\rangle}a(v)+\int_{v\in\langle 2,5\rangle}b(v)+b(5) \\ + & \tau(5)\cdot\int_{v\in\langle 5,6\rangle}b(v)) \end{array}$$

In case of a process term with free time variables such as

$$q \simeq \int_{w\in\langle v,v+1\rangle} a(w) + \int_{w\in\langle v,2v\rangle} b(w)$$

we introduce a partition. Each condition in this partition is an assumption on the ordering of the bounds of the process term q. Therefore each condition determines which τ's have to be added.

Example 7.3.4 *$q \simeq \int_{w\in\langle v,v+1\rangle} a(w) + \int_{w\in\langle v,2v\rangle} b(w)$ will be rewritten to*

$$\begin{array}{lll}
v+1=2v & :\longrightarrow & \int_{w\in\langle v,v+1\rangle} a(w) + \int_{w\in\langle v,v+1\rangle} b(w) \\
v+1<2v & :\longrightarrow & \int_{w\in\langle v,v+1\rangle} a(w) + \int_{w\in\langle v,v+1\rangle} b(w) + b(v+1) \\
 & & \qquad +\tau(v+1)\cdot \int_{w\in\langle v+1,2v\rangle} b(w) \\
v+1>2v & :\longrightarrow & \int_{w\in\langle v,2v\rangle} a(w) + \int_{w\in\langle v,2v\rangle} b(w) + a(2v) \\
 & & \qquad +\tau(2v)\cdot \int_{w\in\langle 2v,v+1\rangle} b(w)
\end{array}$$

7.3.2 Partitioning a process term

As we have seen in Example 7.3.4 we have to consider all possible orderings on the bounds of a process term p. Therefore we need the notion of an *ordered partition* of a finite set S. The sequence $\langle S_1,\ldots,S_n\rangle$ of subsets of S is an ordered partition if $\{S_1,\ldots,S_n\}$ is a partition of S.

Consider a set B of bounds. We construct the set of conditions on B, each defining an ordering on the bounds.

- Construct all possible ordered partitions of B, let us assume that there are m different partitions of B.
- For each ordered partition $\langle B_0,\ldots,B_k\rangle$ we construct a condition α. We take a representative $b_l \in B_l$ for each $l \in \{0,\ldots,k\}$. For each $b \in B_l$ the condition α contains $b = b_l$. Furthermore for $l < l'$ the condition α contains $b_l < b_{l'}$.

In this way we obtain the partition $\{\alpha_j\} = \{\alpha_1,\ldots,\alpha_m\}$ associated to B.

In the following we will denote α by $B_1 < \ldots < B_k$ whenever α is constructed from the ordered partition $\langle B_1,\ldots,B_k\rangle$ of B.

We define the bounds of p, where p is an interval prefix normal form.

$$\begin{array}{lll}
bounds(\int_{w\in\{\!|b,b'|\!\}} P(w) & = & \{b,b'\} \\
bounds(p+q) & = & bounds(p)+bounds(q)
\end{array}$$

The process term p will be partitioned by considering the set of ordered partitions of $bounds(p) \cup \{v\}$, where v is a parameter of the construction.

In case p occurs in a context $\int_{v'\in V} a(v')\cdot p$, then we construct the partition of p that depends on v'.

Assume p is of the form

$$\sum_i \int_{w\in\{\!|_i s_i,u_i|\!\}_i} P_i(w)$$

where each P_i is of the form $a(w)$ or $a(w) \cdot p'$. Consider $\alpha = B_1 < \ldots < B_k$ where $\langle B_1, \ldots, B_n \rangle$ is one of the ordered partitions of $B = \mathit{bounds}(p) \cup \{v\}$.

We construct $\tilde{p}(\alpha, l)$ for $l < n$:

$$\begin{array}{lll}
 & \sum_{i:\alpha \Rightarrow s_i \leq b_l \,\wedge\, b_{l+1} \leq u_i} & \int_{w \in \langle b_l, b_{l+1} \rangle} P_i(w) \\
+ & \sum_{i:\alpha \Rightarrow s_i \leq b_l \,\wedge\, b_{l+1} = u_i} & P_i(w)[b_{l+1}/w] \\
+ & \sum_{i:\alpha \Rightarrow s_i = b_{l+1} = u_i \,\wedge\, \llparenthesis_i = [\,\wedge\, \rrparenthesis_i =]} & P_i(w)[b_{l+1}/w] \\
+ & \sum_{i:\alpha \Rightarrow s_i = b_{l+1} < u_i \,\wedge\, \llparenthesis_i = [} & P_i(w)[b_{l+1}/w]
\end{array}$$

And for $l \geq n$ we take $\tilde{p}(\alpha, l) \simeq \delta$. We have the following proposition.

Proposition 7.3.5 *$l < n$*

$$\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash \tilde{p}(\alpha, l) = b_l \gg p \ggg b_{l+1}$$

Proof. By construction. □

Next, we define

$$\begin{array}{lcll}
v(\alpha) & = & l & \text{such that } v \in B_l \\
u(\alpha, p) & = & l & \text{such that } l \text{ is the smallest index } \geq v(\alpha) \\
 & & & \qquad\qquad \text{with } l' \geq l \text{ implies } \tilde{p}(\alpha, l') \simeq \delta
\end{array}$$

Note that $\alpha \Rightarrow \neg(U_{b_{u(\alpha,p)}}(v \gg p)$. Finally we define $p(\alpha, l)$.

$$\begin{array}{lcll}
p(\alpha, l) & \simeq & \delta & \text{if } l \geq u(\alpha, p) \\
 & \simeq & \tilde{p}(\alpha, l) & \text{if } l = u(\alpha, p) - 1 \\
 & \simeq & \tilde{p}(\alpha, l) + \tau(b_{l+1}) \cdot p(\alpha, l+1) & \text{if } l < u(\alpha, p) - 1
\end{array}$$

Proposition 7.3.6

$$\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash p(\alpha, l) \ggg b_l = \delta$$
$$\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash b_l \gg p(\alpha, l) = p(\alpha, l)$$

Proof. By construction. □

The rewrite rule that partitions p, depending on v, by adding all possible τ's, is

$$\boxed{1. \quad p \quad \longrightarrow_v \quad \sum_{j \in \{1, \ldots, m\}} \{\alpha_j \;:\rightarrow\; p(\alpha_j, v(\alpha_j))\}}$$

where $\{\alpha_j\}$ is the partition associated to $\mathit{bounds}(p) \cup \{v\}$. Finally we have a proposition which states that there is a derivation in $\mathrm{BPA}\rho\delta\mathrm{I}+ \mathrm{B}_I$ that corresponds with the above rewriting.

Proposition 7.3.7 *If $p \longrightarrow_v p'$ by Rule 1 then*

$$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{B}_I \vdash \int_\alpha a(v) \cdot p = \int_\alpha a(v) \cdot p'$$

Proof. p' is of the form $\sum_{j\in\{1,\dots,m\}}\{\alpha_j \ :\rightarrow\ p(\alpha_j, v(\alpha_j))\}$. Take an arbitrary j and take $\beta = \alpha_j$. Let $\{\beta_i \wedge v \in V_i\}$ the v-refinement of β. It is sufficient to show that

$$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{B}_I \vdash \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p = \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p(\alpha, v(\alpha))$$

$$\begin{aligned}
&\mathrm{BPA}\rho\delta\mathrm{I} \vdash\\
&\quad \textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p\\
&= \textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\beta_i \ \wedge\ v\in V_i \ :\rightarrow\ p\}\\
&= \textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\beta_i \ \wedge\ v\in V_i \ :\rightarrow\ \{\alpha \ :\rightarrow\ p\}\}\\
&= \textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\alpha \ :\rightarrow\ p\}
\end{aligned}$$

We use as well $\mathrm{BPA}\rho\delta\mathrm{I}, \neg(U_b(p)) \vdash b \gg p = \delta(b)$, and $\mathrm{BPA}\rho\delta\mathrm{I}, \neg(U(p)) \vdash p \gg b = p$.

First we prove that for l with $v(\alpha)\le l\le u(\alpha,p)$

$$\mathrm{BPA}\rho\delta\mathrm{I} \vdash \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p \ = \ \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_l + p(\alpha,l))$$

by induction on $u(\alpha,p)-l$. There are three cases to consider.

1. $v(\alpha)\le l = u(\alpha,p)$. This case is trivial since $\alpha \Rightarrow \neg(U_{b_l}(p))$. Thus $\mathrm{BPA}\rho\delta\mathrm{I}, \alpha \vdash p \mathrel{\underline{\gg}} b_l = p$, moreover $p(\alpha, u(\alpha,p)) \simeq \delta$. And we have the following derivation.

$$\begin{aligned}
&\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_l + p(\alpha,l))\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\alpha :\rightarrow (p \mathrel{\underline{\gg}} b_l + p(\alpha,l))\}\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\alpha :\rightarrow (p+\delta)\}\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p
\end{aligned}$$

2. $v(\alpha)\le l = u(\alpha,p)-1$.

$$\begin{aligned}
&\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\alpha :\rightarrow (p \mathrel{\underline{\gg}} b_l + b_l \gg p \mathrel{\underline{\gg}} b_{l+1}\}\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\alpha :\rightarrow (p \mathrel{\underline{\gg}} b_l + \tilde{p}(\alpha,l)\}\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot\{\alpha :\rightarrow (p \mathrel{\underline{\gg}} b_l + p(\alpha,l)\}
\end{aligned}$$

3. $v(\alpha)\ \le\ l\ <\ u(\alpha,p)-1$. We use $\mathrm{BPA}\rho\delta\mathrm{I} \vdash (p \mathrel{\underline{\gg}} b) \mathrel{\underline{\gg}} b = p \mathrel{\underline{\gg}} b$ and $\mathrm{BPA}\rho\delta\mathrm{I} \vdash b \gg p \mathrel{\underline{\gg}} b = \delta(b)$.

$$\begin{aligned}
&\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p\\
\overset{\text{ind}}{=}\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_{l+1} + p(\alpha,l+1))\\
\overset{\mathrm{B}_I}{=}\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot((p \mathrel{\underline{\gg}} b_{l+1} + p(\alpha,l+1)) \mathrel{\underline{\gg}} b_{l+1}\\
&\qquad +\tau(b_{l+1})\cdot(p \mathrel{\underline{\gg}} b_{l+1} + p(\alpha,l+1)))\\
\overset{\text{Prop.7.3.6}}{=}\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_{l+1} + \tau(b_{l+1})\cdot p(\alpha,l+1))\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_l + b_l \gg p \mathrel{\underline{\gg}} b_{l+1}\\
&\qquad +\tau(b_{l+1})\cdot p(\alpha,l+1))\\
\overset{\text{Prop.7.3.5}}{=}\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_l + \tilde{p}(\alpha,l)\\
&\qquad +\tau(b_{l+1})\cdot p(\alpha,l+1))\\
=\ &\textstyle\int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \mathrel{\underline{\gg}} b_l + p(\alpha,l))
\end{aligned}$$

We continue

$$\begin{array}{ll} & \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p \\ = & \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(p \ggeq v + p(\alpha, v(\alpha))) \\ = & \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot(v \gg p \ggeq v + v \gg p(\alpha, v(\alpha))) \\ = & \int_{\alpha\wedge\beta_i\wedge v\in V_i} a(v)\cdot p(\alpha, v(\alpha)) \end{array}$$

□

7.3.3 Removing τ's

We have to introduce a rewrite rule that handles the following examples.

Example 7.3.8

$$\begin{array}{ll} & a(1)\cdot(\int_{w\in\langle 1,3\rangle}\tau(w)\cdot(\int_{z\in\langle w,3\rangle}a(z)+b(3))+\int_{w\in\langle 1,3\rangle}a(w)+b(3)) \\ = & a(1)\cdot(\int_{w\in\langle 1,3\rangle}a(w)+b(3)) \end{array}$$

This example shows us that we need a rule for a process term of the form $\int_{w\in\langle b,b'\rangle}\tau(w)\cdot p+q$ where q is a summand of $p[b/w]$.

In the sequel we present rewrite rules of the form $p \longrightarrow_\alpha p'$ that abbreviates $p \longrightarrow \{\alpha :\rightarrow p'\} + \{\neg(\alpha) :\rightarrow p\}$.

3^a $(w \notin fv^*(p))$

$$\int_{w\in\langle b,b'\rangle}\tau(w)\cdot p+q \quad \longrightarrow_{(q \,\underline{\subset}\!\!\leftrightarrow\, p[b/w])} \quad p[b/w]$$

A similar, but more advanced, example is the following process term which has a $\tau(2)$ which does not determine a choice.

Example 7.3.9

$$\begin{array}{ll} & a(1)\cdot(\int_{w\in\langle 1,2\rangle}\tau(w)\cdot(\int_{z\in\langle w,3\rangle}a(z)+b(3))+\int_{w\in\langle 1,2\rangle}a(w) \\ & \qquad +\tau(2)\cdot(\int_{z\in\langle 2,3\rangle}a(z)+b(3))) \\ = & a(1)\cdot(\int_{z\in\langle 1,3\rangle}a(z)+b(3)) \end{array}$$

So, we need an additional rule for process terms of the form $\int_{w\in\langle b,b'\rangle}\tau(w)\cdot p+q+\tau(b')\cdot q'$ where $q+q'$ is a summand of $p[b/w]$.

3^b $(w \notin fv^*(p))$

$$\int_{w\in\langle b,b'\rangle}\tau(w)\cdot p+q+\tau(b')\cdot q' \quad \longrightarrow_{(q+q' \,\underline{\subset}\!\!\leftrightarrow\, p[b/w])} \quad p[b/w]$$

Note that in both of the Rules 3^a and 3^b the summand q may be δ.

Rule 1 rewrites the process term

$$\int_{v\in\langle 0,1]} a(v) + \int_{v\in\langle 1,2\rangle} a(v)$$

into

$$\int_{v\in\langle 0,1\rangle} a(v) + a(1) + \tau(1)\cdot \int_{v\in\langle 1,2\rangle} a(v),$$

though there is no moment of choice at time 1. Hence, we need to rewrite that process term to $\int_{v\in\langle 0,2\rangle} a(v)$. Some more involved examples of τ's which may be removed similarly are given below.

Example 7.3.10

$$\begin{array}{l} \int_{v\in\langle 1,2\rangle} a(v) + a(2) + \tau(2)\cdot \int_{v\in\langle 2,3\rangle} a(v) \\ \hfill \text{must be rewritten to } \int_{v\in\langle 1,3\rangle} a(v) \end{array}$$

$$\begin{array}{l} \int_{v\in\langle 1,2\rangle} a(v) + a(2) + \tau(2)\cdot (\int_{v\in\langle 2,3\rangle} a(v) + b(3)) \\ \hfill \text{must be rewritten to } \int_{v\in\langle 1,3\rangle} a(v) + b(3) \end{array}$$

$$\begin{array}{l} \int_{v\in\langle 1,2]} a(v) + b(2) + \tau(2)\cdot (\int_{v\in\langle 2,3\rangle} a(v) + \int_{v\in\langle 2,3\rangle} b(v)) \\ \hfill \text{must not be rewritten} \end{array}$$

In the sequel we denote an arbitrary $a(b)\cdot p$ or $a(b)$ by $P(b)$. If $b_0 < b_1 < b_2$ and

$$\begin{array}{lcl} p & \simeq & \sum_{i\in I} \int_{v\in\langle b_0,b_1\rangle} P_i(v) + \sum_{j\in J} P'_j(b_1) \\ q & \simeq & \sum_{k\in K} \int_{v\in\langle b_1,b_2\rangle} Q_k(v) + \sum_{l\in L} Q'_l(b_2) \end{array}$$

then the $\tau(b_1)$ can be removed in $p + \tau(b_1)\cdot q$ if for each k there are corresponding indices i and j such that $\int_{v\in\langle b_1,b_2\rangle} Q_k(v)$ can be taken together with $\int_{v\in\langle b_0,b_1\rangle} P_i(v)$ and $P'_j(b_1)$ to $\int_{v\in\langle b_0,b_2\rangle} Q_k(v)$. We have a similar requirement for each j and each k.

This condition is denoted by $p \sim q$ and it is defined as follows:

$$\begin{array}{cclllcll} & \bigwedge_{i\in I} & (& \bigvee_{j\in J} & P_i(b_1) & \underline{\leftrightarrow} & P'_j(b_1) & \\ & & \wedge & \bigvee_{k\in K} & \int_{v\in\langle b_0,b_2\rangle} P_i(v) & \underline{\leftrightarrow} & \int_{v\in\langle b_0,b_2\rangle} Q_k(v) &) \\ \wedge & \bigwedge_{j\in J} & (& \bigvee_{i\in I} & P_i(b_1) & \underline{\leftrightarrow} & P'_j(b_1) & \\ & & \wedge & \bigvee_{k\in K} & P'_j(b_1) & \underline{\leftrightarrow} & Q_k(b_1) &) \\ \wedge & \bigwedge_{k\in K} & (& \bigvee_{i\in I} & \int_{v\in\langle b_0,b_2\rangle} P_i(v) & \underline{\leftrightarrow} & \int_{v\in\langle b_0,b_2\rangle} Q_k(v) & \\ & & \wedge & \bigvee_{j\in J} & P'_j(b_1) & \underline{\leftrightarrow} & Q_k(b_1) &) \end{array}$$

where $z \underline{\leftrightarrow} z'$ denotes the characterizing condition for z and z' (see Lemma 4.6.3). For p, q of the above form we have the following Rule, where $q[b_0/b_1]$ denotes the process term q in which the lower bound b_1 is replaced for b_0. Thus, each summand $\int_{v\in\langle b_1,b_2\rangle} Q_k(v)$ of Q is changed into $\int_{v\in\langle b_0,b_2\rangle} Q_k(v)$.

4	$p + \tau(b) \cdot q$	$\longrightarrow_{(p \sim q)}$	$q[b_0/b_1]$

However, the Rules 3^a,3^b and 4 are only applicable if there are no double τ-summands. For example, the process term

$$\int_{v\in\langle 0,1\rangle} a(v) + a(1) + \tau(1) \cdot \int_{v\in\langle 1,2\rangle} a(v) + \tau(1) \cdot \int_{v\in\langle 1,2\rangle} a(v)$$

can, in its present form, not be rewritten by Rule 4, though we want it to be rewritten (by Rule 4) to $\int_{v\in\langle 0,2\rangle} a(v)$. Hence, we have to take all double τ-summands together before applying either of the Rules $3^a, 3^b$ or 4 which is done by the following Rules 2^a and 2^b.

2^a	$v \notin fv^*(p+q)$ $\int_{v\in\langle b,b'\rangle} \tau(v) \cdot p + \int_{v\in\langle b,b'\rangle} \tau(v) \cdot q$	$\longrightarrow_{(p[b/v] \,\underline{\leftrightarrow}\, q[b/v])}$	$\int_{v\in\langle b,b'\rangle} \tau(v) \cdot p$
2^b	$\tau(b) \cdot p + \tau(b) \cdot q$	$\longrightarrow_{(p \,\underline{\leftrightarrow}\, q)}$	$\tau(b) \cdot p$

7.3.4 The construction of branching basic terms

We combine the rewrite rules to construct branching basic terms. A branching basic term is a term of the form $\sum_i \alpha_i :\rightarrow p_i$, where $\{\alpha_i\}$ is a partition and each p_i is an interval prefix normal form.

We take a prefix normal form p and we perform the steps given below.

(Begin of construction.)

1. We replace each summand $\int_\alpha a(v) \cdot p'$ by $\int_\alpha a(v) \cdot p'_{bb}$, where p'_{bb} is the branching basic term of p'.

   The term p'_{bb} is of the form $\sum_i \alpha_i :\rightarrow p_i$, and by the Lifting Lemma we can rewrite $\int_\alpha a(v) \cdot p'_{bb}$ further to $\sum_i \int_{\alpha\wedge\alpha_i} a(b) \cdot p_i$.

2. Each summand of the form $\int_\alpha a(v) \cdot p$ is rewritten to $\int_\alpha a(v) \cdot p'$, such that $p \longrightarrow_v p'$ by Rule 1.

   Take an arbitrary summand of p', then it is of the form $\gamma :\rightarrow q$, such that q is of the form

$$q_0[+\tau(b_1) \cdot (q_1 + \tau(b_2) \cdot ...\tau(b_n) \cdot q_n)]$$

(where $z[+z']$ denotes a process term which is either of the form z or $z + z'$).

Next, we have a loop, starting for $k = n$, that applies the other rewrite rules for each level k. Each turn we start with a term z_k of the form $\sum_{i \in I_k} \alpha_i :\rightarrow z_k^i$. We take $z_n = tt :\rightarrow q_n$.

(Begin of loop.)
For each $i \in I_k$ we apply the following on z_k^i.

(a) If $k = n$, then we take b_{n+1} such that q_n has a summand $\int_{v \in \langle b_n, b_{n+1} \rangle} P(v)$ or a summand $P(b_{n+1})$.
We remove all summands of the form $\int_{v \in \langle b_k, b_{k+1} \rangle} \delta(v)$ and $\delta(b_{k+1})$.
If, for $k = n$, all summands have been removed then we add a summand $\delta(b_{n+1})$.

(b) Take the double τ-summands together by applying the Rules 2^a and 2^b.

(c) Apply the Rules $3^a, 3^b$, and then the Rule 4.

We rewrite the term that we have obtained further by taking all conditions together in a partition, and we obtain $\{\beta_j :\rightarrow u_k^j\}$.
If $k > 0$ then we lift the partition $\{\beta_j\}$ over the $\tau(b_k)$, and we construct

$$z_{k-1} \simeq \sum_j \beta_j :\rightarrow (q_{k-1} + \tau(b_k) \cdot u_k^j)$$

If $k = 0$, then we are ready. *(End of loop.)*

We do this for each summand of p'. We rewrite the obtained term further by taking all conditions together in a partition, and we obtain

$$\int_\alpha a(v) \cdot \sum_i \alpha_i :\rightarrow p_i$$

We apply the Lifting Lemma again, and we obtain

$$\sum_i \int_{\alpha \wedge \alpha_i} a(v) \cdot p_i$$

Finally, we construct for each $\alpha \wedge \alpha_i$, its v-refinement $\{\alpha_{i,j} \wedge v \in V_{i,j}\}$ and we rewrite the above term to

$$\sum_{i,j} \alpha_{i,j} :\rightarrow \int_{v \in V_{i,j}} a(v) \cdot p_i$$

(End of construction.)

7.3.5 Some properties of branching basic terms

The construction of a branching basic term corresponds with a derivation in the axiom system BPA$\rho\delta$I+ B_I, as is stated by the following proposition.

Proposition 7.3.11 *Let p_{bb} the branching basic term of p then*

$$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{B}_I \vdash \int_\alpha a(v) \cdot p = \int_\alpha a(v) \cdot p_{bb}$$

Proof. We will only give a sketch of the proof. By Proposition 7.3.7 we know that there is a derivation for each application of Rule 1. The Rules 2^a and 2^b are direct instances of the axiom $p + p = p$ and the Rules 3^a,3^b and 4 are instances of the axiom B_I. □

The following two propositions state that branching basic terms are indeed basic terms. First we have a proposition, that says that the time stamps in a branching basic term are always increasing.

Proposition 7.3.12 *If p is a time closed branching basic term, then*

$$p \xrightarrow{a(r)} p' \quad \Longrightarrow \quad \exists p'' \quad p' \equiv r \gg p'' \textit{ and } r \gg p'' \leftrightarrows p''$$

Proof. We only give a sketch of the proof. If $z \longrightarrow_v z'$ by Rule 1, then it is guaranteed that any lower bound b of an initial intervals of z' is in the scope of a condition $v \leq b$. □

Proposition 7.3.13 *If p, q are time closed branching basic terms, then $p \leftrightarrows q$ implies $depth(p) = depth(q)$.*

Proof. We give only a sketch of the proof. Assume $p \leftrightarrows q$ and $depth(p) < depth(q)$, then we motivate that there is a contradiction.

It must be the case that there are p' and q' such that $p' \leftrightarrows q'$, and p' is reachable from p and q' is reachable from q, and q has a summand of the form $\int_{v\in\langle t,t'\rangle} Q(v)$ with $t' \leq t$. For if there is no such q' then p can never be bisimilar with q.

But, in the construction of branching basic terms, especially the first rewrite rule, it is guaranteed that every summand of the form $\int_{\langle b,b'\rangle} P(v)$ is in the scope of a condition α such that $\alpha \Rightarrow b < b'$. Contradiction. □

A process term p is an interval branching basic term if it is a branching basic term such that each summand is of the form $\int_{v\in V} P(v)$. Note, that for such a p there are bounds b, b' such that each interval V is either of the form $\langle b, b'\rangle$ or of the form $[b', b']$. For an interval branching basic term p we define the functions $S(p)$ and $U(p)$ syntactically. $S(p)$ denotes the *start time*; $S(p) = b$ if there p has a summand $\int_{v\in\langle b,b'\rangle} P(v)$. If it does not have such a summand, then it must have a summand $\int_{v\in[b',b']} P(v)$, and we take $S(p) = b'$. $U(p)$ denotes the ultimate delay, as usual, and we take $U(p) = b'$ if p has a summand $\int_{v\in\langle b,b'\rangle} P(v)$ or $\int_{v\in[b',b']} P(v)$.

Lemma 7.3.14 *Let p, q be branching basic terms with $fv(p+q) \subseteq \{v_0\}$ and each for $r_0 \in S$, where $|S| > 1$, we have*

$$p[r_0/v_0] \leftrightarrows q[r_0/v_0]$$

then there is a relation $\mathcal{R}$ which relates subterms of p with subterms of q such that $\mathcal{R}(p', q')$ implies:

- *$S(p') = S(q')$ and $U(p') = U(q')$.*
- *$\forall \sigma \in \Sigma^{cl} : \sigma(p') \leftrightarrows \sigma(q')$.*

Proof. If $fv(p) \subseteq \{v_0\}$ and we write $p \rightsquigarrow_\sigma^S p'$ then we mean that p' is a subterm of p and that there is a a sequence of transitions

$$\sigma(p) \xrightarrow{a_1(r_1)} \sigma(p_1) \ldots \xrightarrow{a_n(r_n)} \sigma(p')$$

where $\sigma(v_i) = r_i$ for some $r_0 \in S$.

Note that since p is also a prefix normal form we may indeed assume that for each transition in such a sequenxe a different variable is bound.

First we prove that there is a relation $\mathcal{R}$ such that $\mathcal{R}(p', q')$ implies

- $S(p') = S(q')$ and $U(p') = U(q')$.
- Forall σ with $p \rightsquigarrow_\sigma^S p'$ and $q \rightsquigarrow_\sigma^S q'$ we have $\sigma(p') \leftrightarrows \sigma(q')$.

First we show that it holds for $\mathcal{R}(p, q)$, and then we assume that we have proven it already for $\mathcal{R}(p', q')$ and we prove it for subterms of p' and q', that have $depth(p') - 1$.

- By assumption $p[r_0/v_0] \leftrightarrows q[r_0/v_0]$ for all r_0 in S. Since for every r_0 in S we have $S(p[r_0/v_0]) = S(q[r_0/v_0])$ and S has more than one element we have $S(p) = S(q)$. Similarly we obtain $U(p) = U(q)$. Finally, by assumption we have for any σ with $p \rightsquigarrow_\sigma^S p$, $q \rightsquigarrow_\sigma^S q$ that $\sigma(p) \leftrightarrows \sigma(q)$.

- Consider a summand $\int_{v \in \langle b, b' \rangle} a(w) \cdot p'' \sqsubseteq p'$ and take σ such that $p \rightsquigarrow_\sigma^S p'$, as p is a branching basic term we know $\sigma(\langle b, b' \rangle)$ can not be empty. Hence there is a finite set

$$\begin{array}{rcl} Q & = & \{ \; q'' \mid \int_{w \in \langle b, b' \rangle} a(w) \cdot q'' \sqsubseteq q' \quad \exists \sigma \in \Sigma^{cl} \; \exists t \in \sigma(\langle b, b' \rangle) \\ & & \qquad\qquad\qquad\qquad \sigma[t/w](p'') \leftrightarrows \sigma[t/w](q'') \; \} \end{array}$$

  Note that $S(p'')$ and $S(q'')$, for $q'' \in Q$, are hyperplanes with dimension smaller or equal to $d+1$, where d is the depth of p' in p, that is the length of the trace $p \rightsquigarrow_\sigma^S p'$.

  For any σ such that $p \rightsquigarrow_{\sigma[t/w]}^S p''$ there is a $q'' \in Q$ such that $q \rightsquigarrow_{\sigma[t/w]}^S q''[t/w]$ where $\sigma[t/w](p'') \leftrightarrows \sigma[t/w](q'')$. In other words, for any such σ there is a $q'' \in Q$ such that $\sigma[t/w]((S(p'')) = \sigma[t/w](S(q''))$, which means that the hyperplane $S(p'')$ is completely covered by the hyperplanes $S(q'')$ for $q'' \in Q$.

  We find a $q'' \in Q$ with $S(p'') = S(q'')$ in finitely many steps. Take an arbitrary $q_0'' \in Q$. At each step $i \geq 0$, there are two cases to consider

- $S(p'') \neq S(q''_i)$. Note, that the intersecting plane of $S(p'')$ and $S(q''_i)$ is one dimension smaller than the one of $S(p'')$. Hence we can find a σ and t such that $\sigma[t/w]$ is "outside" the subplane that has already been covered, i.e. outside the intersections of $S(p'')$ with all $S(q_j)$'s with $j \leq i$:

  We take q''_{i+1} from Q that differs from any q_j with $j \leq i$, such that $\sigma[t/w](p'') \leftrightarroweq \sigma[t/w](q''_{i+1})$.
- $S(p'') = S(q''_i)$ and we are ready.

It can not be the case that for every $q'' \in Q$ we have $S(p'') \neq S(q'')$, for this would imply that an n-dimensional hyperplane can be covered by finitely many $n-1$-dimensional hyperplanes.

Similarly we can deduce that there is a q'' (in Q) such that $U(p'') = U(q'')$ as well.

Now we have to show that $\mathcal{R}(p', q')$ implies

$$\forall \sigma \ : \ \sigma(p') \leftrightarroweq \sigma(q')$$

which we will do by induction on p'. The base case will not be discussed, as it is similar to the case discussed below.

By construction of $\mathcal{R}$ there must be a σ' such that $p \leadsto^S_{\sigma'} p'$, $q \leadsto^S_{\sigma'} q'$ and $\sigma'(p') \leftrightarroweq \sigma'(q')$. Assume

$$\begin{array}{lcl} p' & \simeq & \sum_i \int_{v \in \langle b,b' \rangle} a_i(v) \cdot p_i + \sum_j b_j(b') \cdot p'_j + p'' \\ q' & \simeq & \sum_k \int_{v \in \langle b,b' \rangle} c_k(v) \cdot q_k + \sum_l d_l(b') \cdot q'_l + q'' \end{array}$$

where

$$depth(p'') = depth(p') - 1 = depth(q') - 1 = depth(q'')$$

Take an arbitrary σ and consider a transition $\sigma(p') \xrightarrow{a(r)} z_1$. If it originates from p'' then we have already proven that there is a transition $\sigma(q') \xrightarrow{a(r)} z_2$, originating from q'', such that $z_1 \leftrightarroweq z_2$. So, assume it does not originate from p'', then there are two cases to consider:

- $\sigma(b) < r < \sigma(b')$. Then there is an index i such that $a = a_i$ and

  $$z_1 \equiv \sigma[r/v](v \gg p_i) \leftrightarroweq \sigma[r/v](p_i)$$

  Since $\sigma'(p') \leftrightarroweq \sigma'(q')$ and $\sigma'(\langle b, b' \rangle) \neq \emptyset$ there is a $t \in \sigma'(\langle b, b' \rangle)$ such that

  $$\sigma'(p') \xrightarrow{a(t)} \sigma'[t/v](v \gg p_i) \leftrightarroweq \sigma'[t/v](p_i)$$

  So, there is an index k such that

  $$\sigma'(q') \xrightarrow{a(t)} \sigma'[t/v](v \gg q_k) \leftrightarroweq \sigma'[t/v](q_k)$$

and

$$\sigma'[t/v](p_i) \leftrightarrows \sigma'[t/v](q_k)$$

Hence $\mathcal{R}(p_i, q_k)$, for which we have already proven that

$$\sigma[r/v](p_i) \leftrightarrows \sigma[r/v](q_k))$$

- $r = \sigma(b')$. Similar to the previous case.

□

From this Lemma we obtain a corollary that motivates the usage of the Rule 4.

Corollary 7.3.15 *If we have time closed interval branching basic terms p, q and $t_0 < t_1$ such that*

$$\int_{v \in \langle t_0, t_1 \rangle} a(v) \cdot p \leftrightarrows \int_{v \in \langle t_0, t_1 \rangle} a(v) \cdot q$$

and $\int_{v \in \langle t_0, t_2 \rangle} a(v) \cdot p$, with $t_1 < t_2$, is also a branching basic term, then $\int_{v \in \langle t_0, t_2 \rangle} a(v) \cdot q$ is a branching basic term as well for which

$$\int_{v \in \langle t_0, t_2 \rangle} a(v) \cdot p \leftrightarrows \int_{v \in \langle t_0, t_2 \rangle} a(v) \cdot q.$$

Corollary 7.3.16 *If $fv(p) = \emptyset$ and for all r in S, where $|S| > 1$, we have $p \leftrightarrows q[r/v]$ then $fv(q) = \emptyset$ as well.*

Finally we have a proposition that relates branching bisimulation inclusion with branching basic terms:

Proposition 7.3.17 *Let p be an interval branching basic term with a summand $\int_{v \in V} \tau(v) \cdot p'$ such that $v \notin fv^*(p')$ and $t \in V$ then*

$$t \gg p'[t/v] \;\; \underline{\leftrightarrow}_b \;\; t \gg p$$

Proof. Consider a transition

$$t \gg p'[t/v] \xrightarrow{a(r)} z$$

take an arbitrary $s \in \langle t, r \rangle$, then by Corollary 7.2.3 we have

$$s \gg p'[t/v] \leftrightarrows s \gg p'[s/v]$$

There is a summand $\int_{w \in W} a(v) \cdot p'' \sqsubseteq p'$ such that $z \equiv r \gg p''[t/v][r/w]$. Moreover, since $v \notin fv^*(p')$ we know that $v \notin fv(p'')$. Hence, $z \equiv r \gg p''[r/w]$ and thus we have (for $a \in A_\tau$):

$$\begin{tikzcd}
t \gg p'[t/v] \arrow[rr, "a(r)"] \arrow[d, phantom, "\leftrightarrow"] & & z \\
t \gg p'[t/v] \arrow[r, "\iota(s)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & s \gg p'[t/v] \arrow[r, "a(r)"] \arrow[d, phantom, "\leftrightarrows_b"] & z \\
t \gg p \arrow[r, "\tau(s)"] & s \gg p'[s/v] \arrow[r, "a(r)"] & z
\end{tikzcd}$$

□

7.4 A Theorem for Branching Basic Terms

In this section we prove the main theorem for branching basic terms, saying that two branching basic terms are strongly bisimilar whenever they are branching bisimilar.

Theorem 7.4.1 *If p, q are time closed branching basic terms then*

$$p \mathrel{\underline{\leftrightarrow}_b} q \quad \Longrightarrow \quad p \mathrel{\underline{\leftrightarrow}} q$$

Proof. We prove four Facts for p and q, in the given order.
Take $U(p) = u$.

- Fact 1 $(r < u)$

$$\begin{tikzcd} p \arrow[r, "\tau(r)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & r \gg p'[r/v] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] \\ p \arrow[r, "\iota(r)"] & r \gg p \end{tikzcd} \quad \Longrightarrow \quad v \notin fv^*(p)$$

- Fact 2 $(r < u)$

$$\begin{tikzcd} p \arrow[r, "\tau(r)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & r \gg p'[r/v] \arrow[d, phantom, "\not\underline{\leftrightarrow}_b"] \\ p \arrow[r, "\iota(r)"] & r \gg p \end{tikzcd}$$

- Fact 3 $(r < \min(U(p), U(q))$

$$\begin{tikzcd} p \arrow[r, Rightarrow, "\tau*", "r"'] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & p' \arrow[d, phantom, "\underline{\leftrightarrow}_b"] \\ q \arrow[r, "\iota(r)"] & r \gg q \end{tikzcd} \quad \Longrightarrow \quad p \equiv p'$$

- Fact 4

$$p \mathrel{\underline{\leftrightarrow}_b} q \quad \Longrightarrow \quad p \mathrel{\underline{\leftrightarrow}} q$$

In the proof of Fact 3 we assume that Fact 2 has been proven already for p only. Hence, if $p \mathrel{\underline{\leftrightarrow}_b} q$ and we have proven Fact 2 already for q, then in case of $p \xrightarrow{a(r)} p'$ the condition that there $\exists t, z, q'$ such that

$$\begin{tikzcd} p \arrow[rrr, "a(r)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & & & p' \arrow[dd, phantom, "\underline{\leftrightarrow}_b"] \\ p \arrow[r, "\iota(t)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & t \gg p \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & & \\ q \arrow[r, Rightarrow, "\tau*", "t"'] & z \arrow[rr, "a(r)"] & & q' \end{tikzcd}$$

reduces to $\exists q'$ such that

$$\begin{tikzcd} p \arrow[r, "a(r)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & p' \arrow[d, phantom, "\underline{\leftrightarrow}_b"] \\ q \arrow[r, "a(r)"] & q' \end{tikzcd}$$

If z is a time closed branching basic term with $S(z) = t$ and $U(z) = t'$ and there is an $r \in \langle t, t' \rangle$ then z_r denotes the process term which is constructed from z by replacing each summand $\int_{v \in \langle t,t' \rangle} P(v)$ by $\int_{v \in \langle r,t' \rangle} P(v)$. Note that $r \gg z \underline{\leftrightarrow} z_r$.

- **Proof.** Fact 1
  Assume for $r < u$ that

$$\begin{tikzcd} p \arrow[r, "\tau(r)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & r \gg p'[r/v] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] \\ p \arrow[r, "\iota(r)"] & r \gg p \end{tikzcd}$$

then we have to prove $v \notin fv^*(p')$.

Take an arbitrary $t \in \langle r, u \rangle$ and consider the transition

$$r \gg p \xrightarrow{\tau(t)} t \gg p'[t/v],$$

we have to find a corresponding series of transitions starting from $r \gg p'[r/v]$. Note that p' is smaller than p, thus we may assume that Fact 2 has already been proven for p'.

First we assume that there is a p'' such that

$$\begin{tikzcd} r \gg p \arrow[r, "\tau(t)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & t \gg p'[t/v] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] \\ r \gg p'[r/v] \arrow[r, "\tau(t)"] & t \gg p''[t/v] \end{tikzcd}$$

Since p is a basic term

$$p'[t/v] \underline{\leftrightarrow} t \gg p'[t/v] \underline{\leftrightarrow}_b t \gg p''[t/v] \underline{\leftrightarrow} p''[t/v],$$

by induction $p'[t/v] \underline{\leftrightarrow} p''[t/v]$ and by Proposition 7.3.13 we have $depth(p'[t/v]) = depth(p''[t/v])$. But this cannot be the case since $depth(p'') < depth(p')$, as p'' is a proper subterm of p'.

Hence, it must be the case that

$$\begin{tikzcd} r \gg p \arrow[r, "\tau(t)"] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] & t \gg p'[t/v] \arrow[d, phantom, "\underline{\leftrightarrow}_b"] \\ r \gg p'[r/v] \arrow[r, "\iota(t)"] & t \gg p'[r/v] \end{tikzcd}$$

Let b, b' be the lower bound resp. the upper bound of the initial integrals of p', then every summand of p' is either of the form $\int_{v\in\langle b,b'\rangle} P(v)$ or $P(b')$.

There are two cases to consider:

- The case where $b = v$. Note that

  $$p'[t/v] \leftrightarrows t \gg p'[t/v] \leftrightarrows_b t \gg p'[r/v] \leftrightarrows (p'[r/v])_t.$$

  By induction $p'[t/v] \leftrightarrows (p'[r/v])_t$.
  Take an arbitrary summand $\int_{w\in\langle v,b'\rangle} a(w)\cdot z$ of p'. We will show that $v \notin var(b') \cup fv(z)$.
  Since $p'[t/v] \leftrightarrows (p'[r/v])_t$ there is a z^t such that

  $$\begin{array}{ccc} \int_{w\in\langle t,b'(t)\rangle} a(w)\cdot(z[t/v])) & \sqsubseteq & p'[t/v] \\ \leftrightarrows & & \leftrightarrows \\ \int_{w\in\langle t,b'(r)\rangle} a(w)\cdot(z^t[r/v]) & \sqsubseteq & (p'[r/v])_t \end{array}$$

  Then there must be a summand $\int_{w\in\langle v,b'\rangle} a(w)\cdot z'$ of p' and an infinite subset $S \subseteq \langle r,u\rangle$ such that $\forall t \in S$

  $$\int_{w\in\langle t,b'(t)\rangle} a(w)\cdot(z[t/v]) \leftrightarrows \int_{w\in\langle t,b'(r)\rangle} a(w)\cdot(z'[r/v])$$

  from which we conclude that

  * $b'(t) = b'(r)$ for more than one t, thus it must be the case that $v \notin var(b')$ and thus $b' \in T(\mathcal{S})$.
  * $\forall t \in S$ it holds that $z[t/v] \leftrightarrows z'[r/v]$. Hence, by Corollary 7.3.16, we obtain $v \notin fv(z)$.

  For summands $\int_{w\in\langle v,b'\rangle} a(w)$ we can conclude similarly that v can not occur in b'.

- The case where $b \neq v$. Then since p is a basic term $r \leq b(r)$. $r = b(r)$ (for arbitrary r) implies $b = v$ which case already has been considered. So there are $r < b(r)$ and we may assume as well that we have taken t such that $r < t < b(r)$. Hence

  $$p'[r/v] \leftrightarrows t \gg p'[r/v] \leftrightarrows_b t \gg p'[t/v] \leftrightarrows p'[t/v].$$

  By induction $p'[r/v] \leftrightarrows p'[t/v]$. Since this holds for all $t \in \langle r, b(r)\rangle$ we conclude $v \notin fv(p')$.

For summands $a(b')\cdot p''$ and $a(b')$ of p' we can conclude similarly that v can not occur in b'. Hence, $v \notin fv^*(p')$.

- **Proof.** Fact 2
  We assume for $r < u$ that

$$\begin{tikzcd} p \arrow[r, "\tau(r)"] \arrow[d, phantom, "\leftrightarrow_b"] & r \gg p'[r/v] \arrow[d, phantom, "\leftrightarrow_b"] \\ p \arrow[r, "\iota(r)"] & r \gg p \end{tikzcd}$$

and we will show that this assumption leads to a contradiction. Take $s = S(p)$. First we note $v \notin fv^*(p')$ (by Fact 1) and that there is a $z \not\simeq \delta$ such that

$$p \simeq \int_{v \in \langle s,u \rangle} \tau(v) \cdot p' + z$$

For if $z \simeq \delta$ then Rule 3^a could be applied. Moreover

$$z \simeq \sum_i \int_{v \in \langle s,u \rangle} Z_i(v) + \sum_j Z'_j(u)$$

We will show that the assumption $r \gg p'[r/v] \leftrightarrow_b r \gg p$ leads to the conclusion that either the Rule 3^a or the Rule 3^b is applicable. We will do this in two steps. First we show for each i that $\int_{v \in \langle s,u \rangle} Z_i(v) \;\underline{\leftrightarrow}\; p'[s/v]$. Then we show for each j that either $Z'_j(u) \;\underline{\leftrightarrow}\; p'[s/v]$ or that $Z'_j(u)$ is of the form $\tau(u) \cdot z'$ such that $z' \;\underline{\leftrightarrow}\; p'[s/v]$. Moreover, we show that there is at most one such summand $\tau(u) \cdot z'$, in which case z is of the form $z'' + \tau(u) \cdot z'$ such that $z'' + z' \sqsubseteq p[s/v]$ and Rule 3^b is applicable. If there is no such summand $\tau(u) \cdot z'$ then obviously $z \;\underline{\leftrightarrow}\; p[s/v]$ and Rule 3^a is applicable.

- If $\int_{v \in \langle s,u \rangle} Z_i(v)$ is of the form $\int_{v \in \langle s,u \rangle} a(v)$ then we obtain direct that for each transition

  $$z \xrightarrow{a(t)} \surd \text{ there is a transition } r \gg p'[r/v] \xrightarrow{a(t)} \surd$$

  and thus $r \gg p'[s/v] \xrightarrow{a(t)} \surd$ as well. Hence, $\int_{v \in \langle s,u \rangle} a(v) \;\underline{\leftrightarrow}\; p[s/v]$. Next, assume $\int_{v \in \langle s,u \rangle} Z_i(v)$ is of the form $\int_{v \in \langle s,u \rangle} a(v) \cdot z'$, then

  $$z \xrightarrow{a(t)} t \gg z'[t/v] \quad t \in \langle r, u \rangle$$

  since we assume $r \gg p'[r/v] \leftrightarrow_b r \gg p$ there must be corresponding series of transitions starting from $r \gg p'[r/v]$. Note that we have proven Fact 2 already for $r \gg p'[r/v]$.
  First we assume that $a = \tau$ and that for infinitely many t the transition is matched with and idling:

  $$\begin{tikzcd} r \gg p \arrow[r, "\tau(t)"] \arrow[d, phantom, "\leftrightarrow_b"] & t \gg z'[t/v] \arrow[d, phantom, "\leftrightarrow_b"] \\ r \gg p'[r/v] \arrow[r, "\iota(t)"] & t \gg p'[r/v] \end{tikzcd}$$

  Since $v \notin fv^*(p')$ and p is a basic term

$$p'[t/v] \leftrightarrows t \gg p'[t/v] \leftrightarrows t \gg p'[r/v] \leftrightarrows_b t \gg z'[t/v] \leftrightarrows z'[t/v]$$

By induction $p'[t/v] \leftrightarrows z'[t/v]$, for infinitely many t. Since $v \notin fv^*(p')$ we have as well $v \notin fv^*(z')$ (see Corollary 7.2.2). Hence, p has two double τ-summands, e.g. $\int_{v\in\langle s,u\rangle} \tau(v)\cdot p'$ and $\int_{v\in\langle s,u\rangle} \tau(v)\cdot z'$, and thus Rule 2^a is applicable. Contradiction.

We have obtained that for each transition

$$r \gg z \xrightarrow{a(t)} t \gg z'[t/v],$$

where $r < u$, there is a p'' such that

$$r \gg p'[r/v] \xrightarrow{a(t)} t \gg p''[t/v]$$

and

$$\bullet z'[t/v] \leftrightarrows t \gg z'[t/v] \leftrightarrows_b t \gg p''[t/v].$$

By induction $z'[t/v] \leftrightarrows p''[t/v]$. Hence

$$(3) \quad \forall r \in \langle s,u\rangle \quad \int_{v\in\langle r,u\rangle} a(v)\cdot z' \hookrightarrow p'[r/v]$$

From which we obtain that

$$\int_{v\in\langle s,u\rangle} a(v)\cdot z' \hookrightarrow p'[s/v]$$

For if not, then there must be a transition

$$\int_{v\in\langle s,u\rangle} a(v)\cdot z' \xrightarrow{a(r')} r' \gg z'[r'/v]$$

which cannot be matched by $p'[s/v]$. But this cannot be possible since we may choose r arbitrary close to s in (3).

Concluding, we may say

$$\sum_i \int_{v\in\langle s,u\rangle} Z_i(v) \hookrightarrow p'[s/v]$$

– Now we have to consider $\sum_j Z'_j(u)$. For a transition

$$p \xrightarrow{a(u)} \surd$$

we obtain directly that there is a transition

$$r \gg p'[r/v] \xrightarrow{a(u)} \surd$$

and, hence, $a(u) \sqsubseteq p[s/v]$.

So, assume a summand Z'_j of the form $a(u)\cdot z'$ then

$$r \gg p \xrightarrow{a(u)} u \gg z'$$

and either

- $*$ $a = \tau$ and

$$\begin{tikzcd} r \gg p \arrow[r, "\tau(u)"] \arrow[d, phantom, "\leftrightarroweq_b"] & u \gg z' \arrow[d, phantom, "\leftrightarroweq_b"] \\ r \gg p'[r/v] \arrow[r, "\iota(u)"] & u \gg p'[r/v] \end{tikzcd}$$

Since $v \notin fv^*(p')$ we have $u \gg p'[r/v] \leftrightarroweq p'_u[s/v] \; \underset{\leftrightarrow}{\subset} \; p'[s/v]$ and we have

$$p'_u[s/v] \leftrightarroweq u \gg p'[r/v] \leftrightarroweq_b u \gg z' \leftrightarroweq z'$$

By induction $p'_u[s/v] \leftrightarroweq z'$ and thus $z' \; \underset{\leftrightarrow}{\subset} \; p'[s/v]$.
Note that there is at most one summand $\tau(u) \cdot z'$ such that $u \gg z' \leftrightarroweq_b u \gg p'[r/v]$. For if $\tau(u) \cdot z' + \tau(u) \cdot z'' \sqsubseteq z$ such that $u \gg z'' \leftrightarroweq_b u \gg p'[r/v]$ then we can deduce, using induction, that $z' \leftrightarroweq z''$ and Rule 2^b would be applicable.

- $*$ or there is a p' has a subterm p'' such that

$$\begin{tikzcd} r \gg p \arrow[r, "a(u)"] \arrow[d, phantom, "\leftrightarroweq_b"] & u \gg z' \arrow[d, phantom, "\leftrightarroweq_b"] \\ r \gg p'[r/v] \arrow[r, "a(u)"] & u \gg p''[u/w] \end{tikzcd}$$

and we have

$$z' \leftrightarroweq u \gg z' \leftrightarroweq_b u \gg p''[u/v] \leftrightarroweq p''[u/w]$$

by induction $z' \leftrightarroweq p''[u/v]$ and thus $a(u) \cdot z' \leftrightarroweq a(u) \cdot p''[u/w]$ from which we conclude $a(u) \cdot z' \; \underset{\leftrightarrow}{\subset} \; p'[s/v]$

- **Proof.** Fact 3
We have to prove for $r < \min(U(p), U(q))$

$$\begin{tikzcd} p \arrow[r, Rightarrow, "\tau*"', ""] \arrow[d, phantom, "\leftrightarroweq_b"] & p' \arrow[d, phantom, "\leftrightarroweq_b"] & \\ q \arrow[r, "\iota(r)"] & r \gg q & \end{tikzcd} \Longrightarrow \quad p \equiv p'$$

Since $r < U(p)$ also $p \xrightarrow{\iota(r)} r \gg p$, and by branching bisimulation there is a q' such that $q \overset{\tau*}{\Longrightarrow}_r q'$ and $r \gg p \leftrightarroweq_b q'$). We have, using Proposition 7.3.17,

$$\begin{array}{lllll} r \gg p & \leftrightarroweq_b & q' & \underset{\leftrightarrow_b}{\subset} & r \gg q \\ r \gg q & \leftrightarroweq_b & p' & \underset{\leftrightarrow_b}{\subset} & r \gg p \end{array}$$

by which we obtain that $r \gg p \leftrightarroweq_b p'$ and then we conclude, using Fact 2 and the Stuttering Lemma, that $p \overset{\tau*}{\Longrightarrow}_r p'$ is an empty sequence and $p \equiv p'$.

- **Proof.** Fact 4

$$p \leftrightarroweq_b q \quad \Longrightarrow \quad p \leftrightarroweq q$$

It is sufficient to show that $U(p) = U(q)$, as this implies for any t that $U_t(p) \iff U_t(q)$. Assume $U(p) \leq U(q)$, then we may assume as well that p is of the form

$$z_0 + \tau(u_1) \cdot (z_1 + \tau(u_1) \cdot (...\tau(u_n) \cdot z_n...))$$

for some $n \geq 0$. We take $u_0 = s$, $u_{n+1} = U(q)$. Furthermore we take

$$p_n \overset{def}{=} z_n$$
$$p_i \overset{def}{=} z_i + \tau(u_{i+1}) \cdot p_{i+1} \text{ for } i \in \{0, \ldots, n-1\}$$

Then we have

$$p_i \leftrightarrow_b u_i \gg q$$

and

$$U(q) > U(p) \quad \iff \quad n > 0$$

and we will show that $n > 0$ implies that Rule 4 is applicable from which we obtain that $n = 0$ and, hence, $U(p) = U(q)$.

We have

$$z_{n-1} + \tau(u_n) \cdot z_n \leftrightarrow_b u_{n-1} \gg q \leftrightarrow_b q_{u_{n-1}}.$$

For each $r \in \langle u_{n-1}, u_n\rangle$ we have

$$\begin{array}{llll} z_{n-1} \xrightarrow{a(r)} z' & \Longrightarrow & \exists q' \;\; q_{u_{n-1}} \xrightarrow{a(r)} q' & \text{where } z' \leftrightarrow_b q_{u_{n-1}} \\ q_{u_{n-1}} \xrightarrow{a(r)} q' & \Longrightarrow & \exists z' \;\; z_{n-1} \xrightarrow{a(r)} z' & \text{where } z' \leftrightarrow_b q_{u_{n-1}} \end{array}$$

Hence, using Corollary 7.3.15 and induction, for each summand $\int_{v\in\langle u_{n-1},u_n\rangle} a(v) \cdot z'$ of z_{n-1} there is a corresponding summand $\int_{v\in\langle u_{n-1},u_n\rangle} a(v) \cdot q'$ of $q_{u_{n-1}}$ such that

$$\int_{v\in\langle u_{n-1},u_n\rangle} a(v) \cdot z' \leftrightarrow \int_{v\in\langle u_{n-1},u_n\rangle} a(v) \cdot q',$$

and by Lemma 7.3.14 also

$$\int_{v\in\langle u_{n-1},u_{n+1}\rangle} a(v) \cdot z' \leftrightarrow \int_{v\in\langle u_{n-1},u_{n+1}\rangle} a(v) \cdot q',$$

and vice versa. Similarly for each summand $a(u_n) \cdot z'$ of z_{n-1}, apart from $\tau(u_n) \cdot z_n$, and each summand $\int_{v\in\langle u_n,u_{n+1}\rangle} Z(v)$ of z_n.

Hence $z_{n-1} \sim z_n$ and Rule 4 can be applied.

□

7.5 Completeness for Branching Bis. Eq.

We construct for each basic term its *rooted branching basic term*:

1. We replace every summand $\int_\alpha a(v) \cdot p'$ by $\int_\alpha a(v) \cdot p'_{bb}$, where p'_{bb} is the branching basic term of p'
2. Each p'_{bb} is of the form $\sum_i \alpha_i :\rightarrow p_i$, and we rewrite $\int_\alpha a(v) \cdot p'_{bb}$ further to $\sum_i \int_{\alpha \wedge \alpha_i} a(v) \cdot p_i$.

Lemma 7.5.1 *Let $p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$ and p_{rbb}, q_{rbb} are the rooted branching basic terms of resp. p and q, then*

$$p \mathrel{\underline{\leftrightarrow}}^{\alpha}_{rb} q \quad \Longrightarrow \quad p_{rbb} \mathrel{\underline{\leftrightarrow}}^{\alpha} q_{rbb}$$

Proof. First we note that $\mathrm{BPA}\rho\mathrm{C} + \mathrm{B}_I \vdash p = p_{rbb}, q = q_{rbb}$, by soundness we obtain that forall $\sigma \in \Sigma^{cl}$ we have $\sigma(p) \mathrel{\underline{\leftrightarrow}}_{rb} \sigma(p_{rbb})$ and $\sigma(q) \mathrel{\underline{\leftrightarrow}}_{rb} \sigma(q_{rbb})$. By transitivity of $\mathrel{\underline{\leftrightarrow}}_{rb}$, we obtain that forall $\sigma \in [\alpha]$ we have $\sigma(p_{rbb}) \mathrel{\underline{\leftrightarrow}}_{rb} \sigma(q_{rbb})$.

By the definition of *strongly* rootedness and Theorem 7.4.1, we obtain $\sigma(p_{rbb}) \mathrel{\underline{\leftrightarrow}} \sigma(q_{rbb})$. □

Corollary 7.5.2 (Completeness) $p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$

$$p \mathrel{\underline{\leftrightarrow}}^{\alpha}_{rb} q \quad \Longrightarrow \quad \mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{B}_I, \alpha \vdash p = q$$

Proof. Directly by the previous Lemma and the Completeness for $\mathrel{\underline{\leftrightarrow}}^{\alpha}$ (Theorem 4.6.7).

8

Delay and Weak Bisimulation and Time

8.1 Introduction

In this Chapter we discuss briefly delay bisimulation equivalence and weak bisimulation equivalence.

In the untimed case, delay bisimulation can be found by taking branching bisimulation and relaxing one condition. Next, weak bisimulation is obtained by taking delay bisimulation and allowing τ-transitions afterwards. This is shown in the following figure which is copied from Chapter 1.

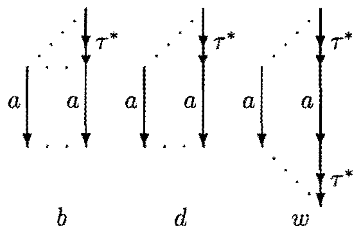


Figure 8.1: Three bisimulations with τ

In Section 8.2 we take the definition of timed branching bisimulation and we derive timed delay bisimulation from it.

Baeten & Bergstra have suggested in [BB91] to interpret τ-transitions as idle transitions. In order to obtain "well behaved" transition systems one has to apply the transitive closure on idle and step transitions. In Section 8.3 we investigate this idea in greater detail and we show that the resulting equivalence coincides with strongly rooted timed delay bisimulation equivalence of section 8.2.

In Section 8.4 we study the axiomatization of timed rooted delay bisimulation.

In Section 8.5 we define timed weak bisimulation equivalence, by taking delay bisimulation and allowing τ-transitions afterwards. However, as this equivalence is not a congruence for ACPρ it will not be studied in detail. In Chapter 11 we discuss delay and weak bisimulation in a so called two phase semantics, in that setting weak bisimulation will be a congruence.

This chapter is based on [Klu92], though Section 8.3 originates from [Klu91a].

8.2 Rooted Delay Bisimulation Equivalence

The first clause of the definition of Idle Branching Bisimulation (see Definition 6.3.3) is

- $< p, t > \xrightarrow{a(r)} < p', r >$ $(a \in A)$ implies that there is a q' such that $(< p, t > \xrightarrow{a(r)} < p', r >)\mathcal{R}(< q, t > \overset{a(r)}{\Longrightarrow} < q', r >)$.

We relax this clause, by removing the condition that the intermediate states must be related, and we obtain the following clause for delay bisimulation.

- $< p, t > \xrightarrow{a(r)} < p', r >$ $(a \in A)$ implies that there is a q' such that $< q, t > \overset{a(r)}{\Longrightarrow} < q', r >$ and $< p', r > \mathcal{R} < q', r >$.

In this way we obtain the following definition for delay bisimulation.

Definition 8.2.1 (Idle Delay Bisimulation)
$\mathcal{R} \subset (T^{cl}(\text{BPA}\rho\delta\text{I}) \times Time)^2$ *is an* idle delay bisimulation *if whenever* $< p, t > \mathcal{R} < q, t >$ *then*

1. $< p, t > \xrightarrow{a(r)} < p', r >$ *$(a \in A)$ implies that there is a q' such that* $< q, t > \overset{a(r)}{\Longrightarrow} < q', r >$ *and* $< p', r > \mathcal{R} < q', r >$.
2. $< p, t > \xrightarrow{\kappa(r)} < p', r >$ *implies that there is a q' such that* $< q, t > \Longrightarrow < q', r >$ *and* $< p', r > \mathcal{R} < q', r >$.
3. $< p, t > \xrightarrow{a(r)} \surd$ *$(a \in A_\tau)$ implies that* $< q, t > \overset{a(r)}{\Longrightarrow} \surd$
4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

We define (rooted) idle delay bisimulation equivalence $\underline{\leftrightarrow}^{\iota}_{(r)d}$ analogous to (rooted) idle branching bisimulation equivalence.

If we define delay bisimulation equivalence in the context of the term semantics, then we identify

$$\int_{v \in \langle 1,2 \rangle} \tau(v) \cdot a(3) + b(3) \text{ and } \int_{v \in \langle 1,3 \rangle} \tau(v) \cdot a(3) + b(3),$$

which are certainly not identified by $\underline{\leftrightarrow}^{\iota}_{d}$, as the idle transition

$$< \int_{v\in\langle 1,3\rangle} \tau(v)\cdot a(3)+b(3),0 > \xrightarrow{\iota(2)} < \int_{v\in\langle 1,3\rangle} \tau(v)\cdot a(3)+b(3),2 >$$

cannot be properly matched by

$$< \int_{v\in\langle 1,2\rangle} \tau(v)\cdot a(3)+b(3),0 > .$$

Hence, we do not define (rooted) delay bisimulation in the context of the term semantics. But, we have another characterization of (rooted) delay bisimulation, that is called *semi delay bisimulation.*.

Definition 8.2.2 (Semi Delay Bisimulation)
$\mathcal{R} \subset (T(\mathrm{BPA}\rho\delta\mathrm{I}) \times Time)^2$ is a semi delay bisimulation *if whenever $< p,t > \mathcal{R} < q,t >$ then*

1. *$< p,t > \overset{a(r)}{\Longrightarrow} < p',r >$ $(a \in A)$ implies that there is a q' such that $< q,t > \overset{a(r)}{\Longrightarrow} < q',r >$, and $< p',r > \mathcal{R} < q',r >$.*
2. *$< p,t > \Longrightarrow < p',r >$ implies that there is a q' such that $< q,t > \Longrightarrow < q',r >$, and $< p',r > \mathcal{R} < q',r >$.*
3. *$< p,t > \overset{a(r)}{\Longrightarrow} \surd$ $(a \in A_\tau)$ implies that $< q,t > \overset{a(r)}{\Longrightarrow} \surd$,*
4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

We define semi delay equivalence $\leftrightarrows_d^{\iota\text{-}semi}$ and rooted demi delay equivalence $\leftrightarrows_{rd}^{\iota\text{-}semi}$ analogously and we have

Lemma 8.2.3

$$< p,t > \leftrightarrows^{\iota}_{(r)d} < q,t > \iff < p,t > \leftrightarrows^{\iota\text{-}semi}_{(r)d} < q,t >$$

Proof. Omitted. □

8.3 Closure Rules and Idle Transitions

In their original paper [BB91] Baeten and Bergstra spend a few words on abstraction, that we work out in more detail in this section. They propose to replace each label τ by ι in the transition systems. To guarantee that the transition systems obey the standard properties of timed transition systems, one has to apply a transitive closure on the idle and step transitions. We can perform this transitive closure by an action rule like:

$$\frac{< p,t > \xrightarrow{\iota(r)} < p',r > \quad < p',r > \xrightarrow{a_\iota(s)} < p'',s >}{< p,t > \xrightarrow{a_\iota(s)} < p'',s >}$$

We require that (strong) bisimulation remains a congruence, also after application of this action rule. This implies, that we may not apply this action rule at the root level, as $\tau(1) \cdot a(2)$ must not be identified with $\tau(1) \cdot a(2) + a(2)$. From the previous discussions on branching bisimulation we know that such an identity is not allowed.

In order to distinguish root states (states that are reachable from the start state by idle transitions only) from internal states, we add a boolean value to each state, initialized on *ff* (false). As soon as an action $a \in A$, or an idling that originates from a τ, has been performed, the boolean value switches to *tt* and remains *tt* throughout the execution of the rest of the process. For example, we have the following transitions:

$$< \tau(2) \cdot p, 0, \mathit{ff} > \xrightarrow{\iota(1)} < \tau(2) \cdot p, 1, \mathit{ff} > \xrightarrow{\iota(2)} < p, 2, \mathit{tt} >$$

In Table 8.1 the action rules for the closure semantics is given.

An idle transition of the form

$$< p, t, \mathit{ff} > \xrightarrow{\iota(r)} < p', t, \mathit{ff} >,$$

so, where the boolean value is *ff* in both states, originates from a "real" idle transition, i.e. one that does not originates from a τ. We use this in certain premises, where we have to distinguish "real" idle transitions from the idle transitions that originate from τ's. For example, in case an alternative composition inherits an idle transition, then the other components are dropped in case the idle transition originates from a τ, otherwise the other components are not dropped. Thus,

$$\begin{array}{ll} < \tau(2) \cdot a(3), 1, \mathit{ff} > \xrightarrow{\iota(2)} < a(3), 2, \mathit{tt} > & \text{and thus} \\ < \tau(2) \cdot a(3) + b(3), 1, \mathit{ff} > \xrightarrow{\iota(2)} < a(3), 2, \mathit{tt} > & \text{as well} \end{array}$$

though

$$\begin{array}{ll} < a(3), 1, \mathit{ff} > \xrightarrow{\iota(2)} < a(3), 2, \mathit{ff} > & \text{and thus} \\ < a(3) + b(3), 1, \mathit{ff} > \xrightarrow{\iota(2)} < a(3) + b(3), 2, \mathit{ff} > & . \end{array}$$

We say that two states $< p, t, b >$ and $< q, t, b >$ are *closure bisimilation equivalent*, denoted by $< p, t, b > \;\underline{\leftrightarrow}^{\iota}_{clos}\; < q, t, b >$, if there is a strong bisimulation $\mathcal{R}$, that relates $< p, t, b >$ and $< q, t, b >$.

Lemma 8.3.1

$$< p, t, \mathit{ff} > \;\underline{\leftrightarrow}^{\iota}_{clos}\; < q, t, \mathit{ff} > \quad \Longleftrightarrow \quad < p, t > \;\underline{\leftrightarrow}^{\iota}_{srd}\; < q, t >$$

Proof. Omitted. □

Here, $\underline{\leftrightarrow}^{\iota}_{srd}$ denotes *strongly rooted idle delay bisimulation equivalence*, which can be defined analogously to strongly rooted idle branching bisimulation equivalence, see Definition 6.8.1.

$$\frac{t < r}{< a(r), t, b > \xrightarrow{a(r)} \surd} \qquad \frac{t < r}{< \tau(r), t, b > \xrightarrow{\iota(r)} \surd}$$

$$\frac{t < r < s}{< a_{\tau\delta}(s), t, b > \xrightarrow{\iota(r)} < a_{\tau\delta}(s), r, b >}$$

$$\frac{< p, t, \mathit{ff} > \xrightarrow{\iota(r)} < p', r, \mathit{ff} >}{< p + q, t, b > \xrightarrow{\iota(r)} < p' + q, r, b >} \qquad \frac{< p, t, \mathit{ff} > \xrightarrow{\iota(r)} < p', r, \mathit{ff} >}{< q + p, t, b > \xrightarrow{\iota(r)} < q + p', r, b >}$$

$$\frac{< p, t, b > \xrightarrow{a_\iota(r)} \surd}{< p + q, t, b > \xrightarrow{a_\iota(r)} \surd} \qquad \frac{< p, t, b > \xrightarrow{a_\iota(r)} \surd}{< q + p, t, b > \xrightarrow{a_\iota(r)} \surd}$$

$$\frac{< p, t, b > \xrightarrow{a_\iota(r)} < p', r, \mathit{tt} >}{< p + q, t, b > \xrightarrow{a_\iota(r)} < p', r, \mathit{tt} >} \qquad \frac{< p, t, b > \xrightarrow{a_\iota(r)} < p', r, \mathit{tt} >}{< q + p, t, b > \xrightarrow{a_\iota(r)} < p', r, \mathit{tt} >}$$

$$\frac{< p, t, b > \xrightarrow{a_\iota(r)} < p', r, b' >}{< p \cdot q, t, b > \xrightarrow{a_\iota(r)} < p' \cdot q, r, b' >} \qquad \frac{< p, t, b > \xrightarrow{a_\iota(r)} \surd}{< p \cdot q, t, b > \xrightarrow{a_\iota(r)} < q, r, \mathit{tt} >}$$

$$\frac{s < r \quad < p, t, \mathit{ff} > \xrightarrow{a_\iota(r)} < p', r, \mathit{tt} >}{< s \gg p, t, b > \xrightarrow{a_\iota(r)} < p', r, \mathit{tt} >} \qquad \frac{r > s \quad < p, t, b > \xrightarrow{a_\iota(r)} \surd}{< s \gg p, t, b > \xrightarrow{a_\iota(r)} \surd}$$

$$\frac{t < r < s}{< s \gg p, t, b > \xrightarrow{\iota(r)} < s \gg p, r, b >} \qquad \frac{s < r \quad < p, t, \mathit{ff} > \xrightarrow{\iota(r)} < p', r, \mathit{ff} >}{s \gg p, t, b > \xrightarrow{\iota(r)} < p', r, b >}$$

Closure Rules

$$\frac{< p, t, \mathit{tt} > \xrightarrow{\iota(t')} < p', t', \mathit{tt} > \quad < p', t', \mathit{tt} > \xrightarrow{a_\iota(r)} < p'', r, \mathit{tt} >}{< p, t, \mathit{tt} > \xrightarrow{a_\iota(r)} < p'', r, \mathit{tt} >} \qquad \frac{< p, t, \mathit{tt} > \xrightarrow{\iota(t')} < p', t', \mathit{tt} > \quad < p', t', \mathit{tt} > \xrightarrow{a_\iota(r)} \surd}{< p, t, \mathit{tt} > \xrightarrow{a_\iota(r)} \surd}$$

$(a \in A,\ a_\iota \in A_\iota,\ a_{\tau\delta} \in A_{\tau,\delta},\ r, t, s \in \mathit{Time})$

Table 8.1: Action Rules for BPA$\rho\delta\tau$ with Closure Rules

8.4 Axioms for Rooted Delay Bis. Eq.

Untimed rooted delay bisimulation equivalence is completely axiomatized by the axioms T1 and T2.

$$\boxed{\begin{array}{lrcl} \text{T1} & p\cdot\tau & = & p \\ \text{T2} & \tau\cdot p & = & \tau\cdot p+p \end{array}}$$

8.4.1 The first τ-axiom

We have seen already in Section 6.2 on timed branching bisimulation that $p\cdot\tau=p$ cannot be transformed straightforwardly to the timed case as $a(1)\cdot\tau(2)$ cannot be identified with $a(1)$.

An option for a generalization is the identity $\text{T1}_?$.

$$\boxed{\begin{array}{l} \text{T1}_? \quad \alpha = (\; \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \;\wedge\; v<b_1 \;\wedge\; U_{b_1}(p)\;) \\ \\ \textstyle\int_{\alpha\wedge\beta}(a(v)\cdot \int_{w\in\langle\!| b_0,b_1|\!\rangle}(\tau(w))\cdot p) \;=\; \int_{\alpha\wedge\beta}(a(v)\cdot b_0 \gg p) \end{array}}$$

However, this identity is derivable from the law T1_I, that has already been discussed in Chapter 6, Section 6.5.

$$\boxed{\begin{array}{l} \text{T1}_I \quad \alpha = (\; \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \;\wedge\; v<b_1 \;\wedge\; U_{b_1}(p)\;) \\ \\ \textstyle\int_{\alpha\wedge\beta}(a(v)\cdot p) \;=\; \int_{\alpha\wedge\beta}(a(v)\cdot (p \gtrsim b_0 + \int_{w\in\langle\!| b_0,b_1|\!\rangle}(\tau(w))\cdot p)) \end{array}}$$

Hence, we consider T1_I as the most appropriate generalization of the untimed T1.

Proposition 8.4.1 $\text{BPA}\rho\delta\text{I}+\text{T1}_I \vdash \text{T1}_?$

Proof. Take α as in $\text{T1}_?$. Without proof we state that $\text{BPA}\rho\delta\text{I} \vdash (b \gg p) \gtrsim b = \delta(b)$.

$$\begin{array}{cl} & \int_{\alpha\wedge\beta}(a(v)\cdot b_0 \gg p) \\ \overset{\text{T1}_I}{=} & \int_{\alpha\wedge\beta}(a(v)\cdot ((b_0\gg p)\gtrsim b_0 + \int_{w\in\langle\!| b_0,b_1|\!\rangle}(\tau(w))\cdot(b_0\gg p)) \\ = & \int_{\alpha\wedge\beta}(a(v)\cdot (\delta(b_0) + \int_{w\in\langle\!| b_0,b_1|\!\rangle}(\tau(w))\cdot p)) \\ = & \int_{\alpha\wedge\beta}(a(v)\cdot \int_{w\in\langle\!| b_0,b_1|\!\rangle}(\tau(w))\cdot p)) \end{array}$$

□

8.4.2 The second τ-axiom

A typical example is

$$\begin{array}{ll} & \int_{w\in\langle 1,2\rangle} \tau(w) \cdot \int_{w\in\langle 1,3]} b(v) \\ \underline{\leftrightarrow}^{\iota}_{rd} & \int_{w\in\langle 1,2\rangle} \tau(w) \cdot \int_{v\in\langle 1,3]} b(v) + \int_{v\in\langle 1,2]} b(v) \end{array}$$

Note, that

$$\begin{array}{ll} & p \equiv \int_{w\in\langle 1,2\rangle} \tau(w) \cdot \int_{v\in\langle 1,3]} b(v) \\ \not\underline{\leftrightarrow}^{\iota}_{rd} & q \equiv \int_{w\in\langle 1,2\rangle} \tau(w) \cdot \int_{v\in\langle 1,3]} b(v) + \int_{v\in\langle 1,3]} b(v), \end{array}$$

as the idle transition $< q, 0 > \xrightarrow{\iota(2)} < q, 2 >$, where $< q, 2 >$ is $< q, 0 >$-rooted, cannot be matched by $< p, 0 >$, due to the rootedness condition. These examples suggest to us the following timed version of the second τ-axiom, T2.

$$\boxed{\mathrm{T2}_I \quad \int_{w\in\llparenthesis b_0,b_1\rrparenthesis}(\tau(w)) \cdot p \;=\; \int_{w\in\llparenthesis b_0,b_1\rrparenthesis}(\tau(w)) \cdot p + b_0 \gg p \gg b_1}$$

In the following example we show how the above pair of terms can be identified within a certain context.

Example 8.4.2
$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{T1}_I + \mathrm{T2}_I \vdash a(1) \cdot (p + c(2)) = a(1) \cdot (q + c(2))$

$$\begin{array}{rl} & a(1) \cdot (\int_{w\in\langle 1,2\rangle} \tau(w) \cdot \int_{w\in\langle 1,3]} b(v) + c(2)) \\ \overset{\mathrm{T1}_I}{=} & a(1) \cdot (\int_{w\in\langle 1,2\rangle} \tau(w) \cdot (\int_{w\in\langle 1,2]} b(v) + \tau(2) \cdot \int_{w\in\langle 2,3]} b(v)) + c(2)) \\ \overset{\mathrm{T2}_I}{=} & a(1) \cdot (\int_{w\in\langle 1,2\rangle} \tau(w) \cdot (\int_{w\in\langle 1,2]} b(v) + \tau(2) \cdot \int_{w\in\langle 2,3]} b(v)) + \\ & \qquad 1 \gg (\int_{w\in\langle 1,2]} b(v) + \tau(2) \cdot \int_{w\in\langle 2,3]} b(v)) \gg 2 + c(2)) \\ = & a(1) \cdot (\int_{w\in\langle 1,2\rangle} \tau(w) \cdot (\int_{w\in\langle 1,2]} b(v) + \tau(2) \cdot \int_{w\in\langle 2,3]} b(v)) + \\ & \qquad \int_{w\in\langle 1,2]} b(v) + \tau(2) \cdot \int_{w\in\langle 2,3]} b(v) + c(2)) \\ \overset{\mathrm{T1}_I}{=} & a(1) \cdot (\int_{w\in\langle 1,2\rangle} \tau(w) \cdot \int_{w\in\langle 1,3]} b(v) + \int_{w\in\langle 1,3]} b(v) + c(2)) \end{array}$$

As in the untimed case we can derive B_I from $\mathrm{T1}_I$ and $\mathrm{T2}_I$.

Proposition 8.4.3

$$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{T1}_I + \mathrm{T2}_I \vdash \mathrm{B}_I$$

Proof. Omitted. □

Theorem 8.4.4 (Soundness)

$$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{T1}_I + \mathrm{T2}_I \vdash p = q \quad \Longrightarrow \quad p \;\underline{\leftrightarrow}_{rd}\; q$$

Proof. Omitted. □

The law $T2_I$ is not sound for *strongly rooted delay bisimulation equivalence.* Therefore, we formulate $T2_I^{sr}$, by applying $T2_I$ in a context.

$$\boxed{\begin{array}{l} \mathrm{T2}_I^{sr} \\ \\ \int_\alpha(a(v)\cdot(\int_{w\in\llparenthesis b_0,b_1\rrparenthesis}(\tau(w))\cdot p+q)) \;\; = \\ \int_\alpha(a(v)\cdot(\int_{w\in\llparenthesis b_0,b_1\rrparenthesis}(\tau(w))\cdot p+b_0 \gg p \mathrel{\underline{\gg}} b_1+q) \end{array}}$$

We do not study whether $T1_I$ and $T2_I$ axiomatize $\underline{\leftrightarrow}^{\iota}_{rd}$ completely, but we think that it can be proven, using the techniques from the previous chapter.

8.4.3 Delay bisimulation without integration coincides with branching bisimulation

If we formulate the axiom $T2_I$ in the context of $T(\mathrm{BPA}\rho\delta\tau)$, then we obtain

$$\tau(t)\cdot p \;=\; \tau(t)\cdot p + t \gg p \mathrel{\underline{\gg}} t.$$

Without proof we state that $t \gg p \mathrel{\underline{\gg}} t$ reduces to $\delta(t)$. Hence, the axiom $T2_I$ does not add any new identities over $T(\mathrm{BPA}\rho\delta\tau)$.

8.5 Weak Bisimulation and Time

The first clause of the definition of Idle Delay Bisimulation (see Definition 8.2.1) is

- $< p,t > \xrightarrow{a(r)} < p',r >$ $(a \in A)$ implies that there is a q' such that $< q,t > \overset{a(r)}{\Longrightarrow} < q',r >$ and $< p',r > \mathcal{R} < q',r >$.

We extend this clause, by adding an additional sequence of τ and idle transitions afterwards, in order to get a corresponding clause for idle delay bisimulation.

- $< p,t > \xrightarrow{a(r)} < p',r >$ $(a \in A)$ implies that there are z, q' and t' such that
  - $< q,t > \overset{a(r)}{\Longrightarrow} < z,r > \overset{\tau *}{\Longrightarrow}_{t'} < q',t' >$,
  - $< p',t' > \mathcal{R} < q',t' >$,
  - $< p',r > \xrightarrow{\iota(t')} < p',t' >$ and $< p',r > \overset{a_\tau(r')}{\not\longrightarrow}$ for all $r' \in \langle r,t']$

  We need the condition that $< p',r > \xrightarrow{\iota(t')} < p',t' >$ as only states with the same time value can be related. Furthermore, the condition $< p',r > \overset{a_\tau(r')}{\not\longrightarrow}$ for all $r' \in \langle r,t']$ is needed, since the weak bisimulation skips all behavior in between r and t' anyway, so we require that there is no behavior in that interval.

We define *weak bisimulation equivalence*, denoted by $\leftrightarrows_w$, and *rooted weak bisimulation equivalence*, denoted by $\leftrightarrows_{rw}$.

Definition 8.5.1 (Idle Weak Bisimulation)
$\mathcal{R} \subset (T^{cl}(\mathrm{BPA}\rho\delta\mathrm{I}) \times Time)^2$ *is an* idle weak bisimulation *if whenever* $< p, t > \mathcal{R} < q, t >$ *then*

1. $< p, t > \xrightarrow{a(r)} < p', r >$ *($a \in A$) implies that there are z, q' and t' such that*
   - $< q, t > \overset{a(r)}{\Longrightarrow} < z, r > \overset{\tau *}{\Longrightarrow}_{t'} < q', t' >$,
   - $< p', t' > \mathcal{R} < q', t' >$,
   - $< p', r > \xrightarrow{\iota(t')} < p', t' >$ *and* $< p', r > \overset{a_\tau(r')}{\not\longrightarrow}$ *for all* $r' \in \langle r, t']$
2. $< p, t > \xrightarrow{\kappa(r)} < p', r >$ *implies that there is a q' and t' such that*
   - $< q, t > \Longrightarrow < q', t' >$,
   - $< p', t' > \mathcal{R} < q', t' >$,
   - $< p', r > \xrightarrow{\iota(t')} < p', t' >$ *and* $< p', r > \overset{a_\tau(r')}{\not\longrightarrow}$ *for all* $r' \in \langle r, t']$
3. $< p, t > \xrightarrow{a(r)} \surd$ *($a \in A_\tau$) implies that there are z and s such that*
   - $< q, t > \Longrightarrow < z, s > \xrightarrow{a(r)} \surd$,
   - $< p, t > \mathcal{R}(< q, t > \Longrightarrow < z, s >)$.
4. *Respectively (1), (2) and (3) with the role of p and q interchanged.*

8.6 The Third τ axiom

First we give a typical example.

Example 8.6.1

$$a(1) \cdot (\tau(2) \cdot b(3) + c(2)) \leftrightarrows_{rw} \quad a(1) \cdot (\tau(2) \cdot b(3) + c(2)) + a(1) \cdot b(3)$$

This identity looks like an instance of the untimed axiom T3

$$a \cdot (\tau \cdot p + q) = a \cdot (\tau \cdot p + q) + a \cdot p.$$

We generalize this axiom to the timed case.

T3$_\rho$ $\quad r < t$

$$a(r) \cdot (\tau(t) \cdot p + q) \;=\; a(r) \cdot (\tau(t) \cdot p + q) + a(r) \cdot \tau(t) \cdot p$$

T3$_I$ $\quad \alpha \;=\; (\; \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \;\wedge\; v < b_1 \;)$

$$\int_{\alpha\wedge\beta}(a(v) \cdot (\int_{w\in\langle\!| b_0,b_1 |\!\rangle}(\tau(w) \cdot p) + q)) \;=\;$$
$$\int_{\alpha\wedge\beta}(a(v) \cdot (\int_{w\in\langle\!| b_0,b_1 |\!\rangle}(\tau(w) \cdot p) + q) + \int_{\alpha\wedge\beta} a(v) \cdot \int_{w\in\langle\!| b_0,b_1 |\!\rangle}(\tau(w) \cdot p))$$

We obtain new identities without τ, such as BPA$\rho\delta$, as well, these are characterized by combining T1$_\rho$ and T3$_\rho$.

Let $r < t < U(p) \;\wedge\; U(q) \leq t$ then

$$\begin{aligned} a(r) \cdot (t \gg p + q) &\overset{\mathrm{T1}_\rho}{=} a(r) \cdot (\tau(t) \cdot p + q) \\ &\overset{\mathrm{T3}_\rho}{=} a(r) \cdot (\tau(t) \cdot p + q) + a(r) \cdot \tau(t) \cdot p \\ &\overset{\mathrm{T1}_\rho}{=} a(r) \cdot (t \gg p + q) + a(r) \cdot (t \gg p) \end{aligned}$$

Theorem 8.6.2 (Soundness) $p, q \in T(\mathrm{BPA}\rho\delta\mathrm{I})$

$$\mathrm{BPA}\rho\delta\tau + \mathrm{T1}_I + \mathrm{T2}_I + \mathrm{T3}_I \vdash p = q \quad\Longrightarrow\quad p \leftrightarroweq_{rw} q$$

Proof. Omitted

8.7 The Extension to ACPρ

In Chapter 1, Section 1.4, we have discussed briefly the extension of untimed delay and weak bisimulation to ACP.

In the timed case we have similar phenomena. Consider for example the pair

$$\int_{v\in\langle 1,2\rangle} \tau(v) \cdot \int_{v\in\langle 1,2\rangle} a(v) \leftrightarroweq^{t}_{rd} \int_{v\in\langle 1,2\rangle} \tau(v) \cdot \int_{v\in\langle 1,2\rangle} a(v) + \int_{v\in\langle 1,2\rangle} a(v)$$

If we take $\gamma(a, b) = c \neq \delta$, then

$$(\int_{v\in\langle 1,2\rangle} \tau(v) \cdot \int_{v\in\langle 1,2\rangle} a(v) + \int_{v\in\langle 1,2\rangle} a(v)) | \int_{v\in\langle 1,2\rangle} b(v)$$

has a summand $\int_{v\in\langle 1,2\rangle} c(v)$. However, according to the operational semantics of this chapter the process term $(\int_{v\in\langle 1,2\rangle} \tau(v) \cdot \int_{v\in\langle 1,2\rangle} a(v) | \int_{v\in\langle 1,2\rangle} b(v)$ has no transitions at all.

Hence, $\leftrightarroweq_{rd}$ is not a congruence for ACPρ. We think that $\leftrightarroweq_{rd}$ is a congruence over ACPρ without the auxiliary operators $\mathbin{||\!\_}$ and $|$, in which case we need a CCS alike expansion theorem for the axiomatization of $\|$. We discuss an expansion theorem for a slightly different setting in Chapter 12. Another way of repairing the above example, is to add action rules to the term semantics, that are similar to the closure rules we have discussed as well. These problems are subject for further research.

For the case of time weak bisimulation the case is even worse, as weak bisimulation is not a congruence for the merge ($\|$). Take $p = d(1) \cdot (a(3) + b(2))$ and $q = d(1) \cdot (a(3) + b(2)) + d(1) \cdot a(3)$ where, $\gamma(b, \bar{b}) = c$. Then we have BPA$\rho\delta$+ T3$_\rho \vdash p = q$. But in a context they can be distinguished. In $\partial_{\{b\}}(p\|\bar{b}(2))$ at time 2 a communication of $b(2)$ with $\bar{b}(2)$ is forced since it is the only option for the whole process not to deadlock at 2. However, $\partial_{\{b\}}(q\|\bar{b}(2))$ has a deadlock at time 2.

$$\begin{array}{lrcl} \mathrm{ACP}\rho \vdash & \partial_{\{b\}}(p\|\bar{b}(2)) & = & a(1) \cdot c(2) \\ \mathrm{ACP}\rho \vdash & \partial_{\{b\}}(q\|\bar{b}(2)) & = & a(1) \cdot c(2) + a(1) \cdot \delta(2) \end{array}$$

This counterexample is due to Jan Bergstra ([Ber92]). Hence, weak bisimulation is not a congruence in ACPρ. Therefore, we think that weak bisimulation is not appropriate for extension with time, at least in the context of ACPρ. This problem is due to a interaction of our operational semantics and weak bisimulation. In our operational semantics every transition takes time and consecutive actions cannot occur at the same point in time.

In Chapter 11 we discuss a two phase semantics for ACPρ, in which rooted weak bisimulation equivalence does not suffer this latter problem. In Chapter 12 we show that weak bisimulation in such a two phase semantics corresponds to other notions of timed weak bisimulation as can be found in the literature [Wan91a],[MT92], [Che93] and [QdFA93].

Part IV

Guarded Recursion

9

Prefixed Integration and Guarded Recursion

9.1 Introduction

In this chapter we generalize the definitions and results of Section 1.5, where we have introduced recursion and guardedness in BPA$\delta\tau$, into the context of BPA$\rho\delta\tau$I.

The main difference with the untimed setting is that we parameterize recursion variables with a time variable. For example, when we define

$$X(v) \overset{def}{=} a(v) \cdot X(v+1)$$

then $X(1)$ is the process that executes an a at time $1, 2, 3, \ldots$. For simplicity we will restrict ourselves to the case where each recursion variable is parameterized by exactly one time variable.

9.2 Some Definitions

We assume a set $RVar$ of *recursion variables*, with typical element X. If $\mathcal{R}$ is a subset of $RVar$ then we denote by $\mathcal{T}(\mathcal{R}, \mathrm{BPA}\rho\delta\tau\mathrm{I})$ the set of process terms over BPA$\rho\delta\tau$I in which the instantiated recursion variables, that are expressions of the form $X(b)$ where b is a bound, from $\mathcal{R}$ may occur as atomic constructs. We put $fv(X(b)) = var(b)$.

A *timed specification* E is a finite collection of declarations of the form

$$\{X_0(v_i) \overset{def}{=} p_0, \ldots, X_n(v_n) \overset{def}{=} p_n\}$$

where $p_i \in (\{X_0, \ldots, X_n\}, \mathrm{BPA}\rho\delta\tau\mathrm{I})$ and $i \neq j$ implies $X_i \neq X_j$. We will restrict ourselves to *time closed* declarations, i.e. $fv(p_i) \subseteq \{v_i\}$. We denote the set $\{X_0, \ldots, X_n\}$ by $rvar(E)$. For $X \in rvar(E)$ we denote the right hand side of the declaration of $X(v)$ in E by $p^E_{X(v)}$. If $X \notin rvar(E)$ then $p^E_{X(v)}$ denotes δ.

We parameterize the action relations of our operational semantics by a specification E. We have two additional action rules, which are given in Table 9.1. We obtain equivalences like $\leftrightarrow_s^E$, $\leftrightarrow_b^E$ and $\leftrightarrow_{rb}^E$ in the obvious way.

$$\frac{p^E_{X(v)}[t/v] \xrightarrow{a}_E p'}{X(t) \xrightarrow{a}_E p'} \qquad \frac{p^E_{X(v)}[t/v] \xrightarrow{a}_E \surd}{X(t) \xrightarrow{a}_E \surd}$$

Table 9.1: Action Rules for Recursion

If p is a process term and there is a time variable v such that $fv(p) \subseteq \{v\}$, then we denote $p[b/v]$ by $p(b)$. If $fv(p) = \{v\}$, then the time variable v can be find easily when we write $p(b)$, and in case $fv(p) = \emptyset$, then it doesn't matter which v we take as $p[b/v]$ can be reduced to p. Moreover, when we write $p(b)$, then we assume implicitly that $|fv(p)| \leq 1$.

Definition 9.2.1 (the notion of a solution) *$p \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$, with $|fv(p)| \leq 1$, is a b-*solution *for X in E modulo $\leftrightarrow$, if $p(b) \leftrightarrow p^E_X(b)$.*

We have similar definitions for $\leftrightarrow_b$ and $\leftrightarrow_{rb}$.

In order to define the notion of guardedness we (re)define the auxiliary boolean function $G^E_{\mathcal{R}}$ [1] on process terms, see Table 9.2.

Proposition 9.2.2 *For all E, $\mathcal{R}$ and $p \in T(RVar, \mathrm{BPA}\rho\delta\tau\mathrm{I})$ there is a boolean expression α, either tt or ff, such that* $\mathrm{A4,5 + G1\text{-}9 + B1,2} \vdash G^E_{\mathcal{R}}(p) = \alpha$

Proof. As in the untimed case, see Proposition 1.5.3. □

Definition 9.2.3 (Guardedness) *The specification E is* guarded *if for all $X \in rvar(E)$* $\mathrm{A4,5 + G1\text{-}9 + B1,2} \vdash G^E_{\emptyset}(X(v)) = tt$.

And, of course, if a specification E is guarded then all process terms over E are guarded as well.

[1]The above definition of $G^E_{\mathcal{R}}(p)$ is rather syntactical. For example, in case

$$X(v) \overset{def}{=} \int_{w \in \langle v, 10\rangle} \tau(w) \cdot X(v+1) + a(v+10)$$

we could say that $X(0)$ is guarded as there are only finitely many unfoldings possible. We have chosen not to do this. It complicates the notion of guardedness since it has to be a conditional expression. Moreover, it is not clear that Proposition 9.2.2 can be proven, take for example

$$Y(v) \overset{def}{=} \int_{w \in \langle 0, 10\rangle} \tau(w) \cdot Y(w) + a(v+10)$$

G1	$G^E_{\mathcal{R}}(\int_\alpha a(v))$	$=$	tt	
G2	$G^E_{\mathcal{R}}(\int_\alpha \tau(v))$	$=$	tt	
G3	$G^E_{\mathcal{R}}(\int_\alpha a(v) \cdot p)$	$=$	tt	
G4	$G^E_{\mathcal{R}}(\int_\alpha \tau(v) \cdot p)$	$=$	$G^E_{\mathcal{R}}(p)$	
G5	$G^E_{\mathcal{R}}(p+q)$	$=$	$G^E_{\mathcal{R}}(p) \wedge G^E_{\mathcal{R}}(q)$	
G6	$G^E_{\mathcal{R}}(X(b))$	$=$	$G^E_{\mathcal{R}\cup\{X\}}(p^E_X)$	if $X \in rvar(E) - \mathcal{R}$
G7	$G^E_{\mathcal{R}}(X(b))$	$=$	ff	otherwise
G8	$G^E_{\mathcal{R}}(X(b) \cdot p)$	$=$	$G^E_{\mathcal{R}\cup\{X\}}(p^E_X \cdot p)$	if $X \in rvar(E) - \mathcal{R}$
G9	$G^E_{\mathcal{R}}(X(b) \cdot p)$	$=$	ff	otherwise

Table 9.2: Axioms for the (boolean) guardedness function

Proposition 9.2.4 *Let E be a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$ then* $\mathrm{A4,5} + \mathrm{G1\text{-}9} + \mathrm{B1,2} \vdash G^E_{\emptyset}(p) = tt$.

Proof. Omitted. □

9.3 Axioms for Recursion and Projection

REC^E	$X(b) = p^E_X[b/v]$
RSP^E_G	$\overline{p}(\overline{b}) = p_{\overline{X}}[\overline{p}/\overline{X}](\overline{b})\ ,\ \overline{q}(\overline{b}) = p_{\overline{X}}[\overline{q}/\overline{X}](\overline{b}) \implies \overline{p}(\overline{b}) = \overline{q}(\overline{b})$

Table 9.3: Additional axioms for recursion

As in the untimed case we have two axioms, REC^E and RSP^E_G, they are given in Table 9.3. And we redefine the projection operator as well, see Table 9.4.

If $\overline{p} = (p_1, \ldots, p_n)$ is a vector of process terms (such that $|fv(p_i)| \leq 1$) and $\overline{b} = (b_1, \ldots, b_n)$ is a vector of bounds, then we denote by $\overline{p}(\overline{b})$ the vector of process terms $(p_1(b_1), \ldots, p_n(b_n))$. For the other notations that are used in Table 9.3 we refer to Section 1.5.3.

PR1	$\pi_0(\int_\alpha a(v))$	$=$	δ
PR2	$\pi_{n+1}(\int_\alpha a(v))$	$=$	$\int_\alpha a(v)$
PR3	$\pi_n(\int_\alpha \tau(v))$	$=$	$\int_\alpha \tau(v)$
PR4	$\pi_0(\int_\alpha a(v) \cdot p)$	$=$	δ
PR5	$\pi_{n+1}(\int_\alpha (a(v) \cdot p))$	$=$	$\int_\alpha (a(v) \cdot \pi_n(p))$
PR6	$\pi_n(\int_\alpha (\tau(v) \cdot p))$	$=$	$\int_\alpha (\tau(v) \cdot \pi_n(p))$
PR7	$\pi_n(p+q)$	$=$	$\pi_n(p) + \pi_n(q)$

$a \in A_\tau,\ n \geq 0$

Table 9.4: Axioms for the projection operator

9.4 The Soundness of the Restricted Recursion Specification Principle

We formulate the notion of a head form as in the untimed case and we have a similar proposition:

Proposition 9.4.1 *Let E be a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$, then there is a p' such that p' is in head form and* $\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{REC}^E + \mathrm{PR1\text{-}7} \vdash \pi_n(p) = p'$

Proof. See the proof of Proposition 1.5.6. □

Lemma 9.4.2 *If E is a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$, then for each n there is a finite process term p', without occurrences of the projection operator, such that*

$$\mathrm{BPA}\rho\delta\mathrm{I} + \mathrm{REC}^E + \mathrm{PR1\text{-}7} \vdash \pi_n(p) = p'$$

Proof. See the proof of Proposition 1.5.7. □

Lemma 9.4.3 *Let E be a guarded specification with $X \in rvar(E)$ and $p \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$ with $|fv(p)| \leq 1$ such that p is a b-solution for $X(v)$, then for all n we have $\pi_n(p(b)) \leftrightarrow_{(rb)} \pi_n(X(b))$.*

Proof. The proof is almost identical to the proof of the untimed version (see Lemma 1.5.8).

Since p is a b-solution for $X(v)$ in E we have $\pi_n(p(b)) \leftrightarrow_{(rb)} \pi_n(p_X[p/X][b/v])$. Consider the derivation between $\pi_n(p_{X(v)}[b/v])$ and $hnf\,(\pi_n(p_{X(v)}[b/v]))$, note that the latter process term is a finite process term, so X does not occur in it. For each identity in this derivation for which REC^E $X(b) = p_{X(v)}[b/v]$ is used we apply $p(b) = p_{X(v)}[p/X][b/v]$ instead. This gives us a derivation between $\pi_n(p_{X(v)}[p/X][b/v])$ and *hnf* $(\pi_n(p_{X(v)}[b/v]))$ and we are ready. □

Lemma 9.4.4 (Projection Lemma) *If E is a guarded specification with $X \in rvar(E)$ such that both $p, q \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$, with $fv(p) \leq 1, fv(q) \leq 1$, are b-solutions for $X(v)$ in E modulo $\underline{\leftrightarrow}_{(rb)}$, then for all n we have $\pi_n(p(b)) \underline{\leftrightarrow}_{(rb)} \pi_n(q(b))$.*

Proof. Immediate from Lemma 9.4.3. □

Recall that $p \overset{\tau*}{\Longrightarrow} p'$ denotes that there is a sequence of timed τ transitions from p to p'.

Proposition 9.4.5 *If E is a guarded specification and $p \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$ then the set*

$$\{p' | p' \text{ is a subterm of } p, \exists\sigma\ p \overset{\tau*}{\Longrightarrow} p'\}$$

is finite.

Proof. By induction on $l(p)$. □

Again we have AIP^E_G:

AIP^E_G	$\forall n\ :\ \pi_n(p) = \pi_n(q) \quad \Longrightarrow$	$p = q$

And we prove

Theorem 9.4.6 (Soundness of AIP$_G$) *If E is a guarded timed specification and $p, q \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$ then*

$$\forall n\ i:\ \pi_n(p) \underline{\leftrightarrow}^E_{(rb)} \pi_n(q) \quad \Longrightarrow \quad p \underline{\leftrightarrow}^E_{rb} q$$

Proof. Let p' be a subterm of p such that p' can be reached from p in more than zero transitions. Similarly we take a subterm q' of q. We define for each m a relation $\sim_m$ such that

$$p' \sim_m q' \quad \Longleftrightarrow \quad \pi_m(p') \underline{\leftrightarrow}_b \pi_m(q')$$

and we put $p' \sim q'$ if for all m we have $p' \sim_m q'$.

We show first that $\sim$ is a branching bisimulation. Take p', q' such that $p' \sim q'$.

- Consider p'' such that $p' \xrightarrow{a(r)} p''$, where $a \in A$, and put

$$\begin{aligned} S_n \;=\; \{\,(z, q^*, v, \sigma) \mid\ & q' \Longrightarrow \sigma(z) \xrightarrow{a(r)} \sigma[r/v](q^*), \\ & p' \sim_{n+1} (q' \Longrightarrow \sigma(z)),\ p'' \sim_n \sigma[r/v](q^*)\} \end{aligned}$$

and

$$S_n^{sym} = \{ (z, q^*, v) \mid \exists\, \sigma (z, q^*, v, \sigma) \in S_n \}$$

Then we have

1. $S_0 \supseteq S_1 \supseteq S_2 \supseteq \ldots$, since $u \sim_{k+1} u'$ implies $u \sim_k u'$. Thus $S_0^{sym} \supseteq S_1^{sym} \supseteq S_2^{sym} \supseteq \ldots$ as well.
2. For all n $S_n \neq \emptyset$ since $p' \sim_{n+1} q'$. Thus $S_n^{sym} \neq \emptyset$ as well.
3. For all n S_n^{sym} is finite, by Proposition 9.4.5.

Hence $\bigcap_{n=0}^{\infty} S_n^{sym} \neq \emptyset$ and we can take a tuple $(z, q'', v) \in \bigcap_{n=0}^{\infty} S_n^{sym}$. Then there is a σ such that $(z, q'', v, \sigma) \in \bigcap_{n=0}^{\infty} S_n$ such that
$q' \Longrightarrow \sigma(z) \xrightarrow{a} \sigma[r/v](q'')$, $p' \sim (q' \Longrightarrow \sigma(z))$ and $p'' \sim \sigma[r/v](q'')$.

The other cases and the rest of the proof can be derived from the above case and the proof of the untimed version of the Theorem (see Theorem 9.4.6). □

Theorem 9.4.7 (Soundness of RSP$_G^E$) *If E is a timed guarded specification with $X \in rvar(E)$ such that both $p, q \in T(rvar(E), \mathrm{BPA}\rho\delta\tau\mathrm{I})$ with $fv(p) \leq 1, fv(q) \leq 1$, are b-solutions for $X(v)$ in E then $p(b) \leftrightarrow_{rb}^{E} q(b)$.*

Proof. Direct by the Projection Lemma and the Soundness of AIP$_G^E$. □

10

Protocol Verification

10.1 Introduction

Process algebra, i.e. untimed process algebra such as ACP [BW90], can be used to prove that the implementation of a protocol meets its specification, for a reference see [Bae90]. The standard example is the alternating bit protocol, see [BW90].

In this chapter we will show that the techniques for protocol verification which are used in untimed process algebra can be used in the timed case as well. We give a verfication of the PAR-protocol (*Positive Acknowledgement with Retransmission*), that has been specified in [BB91]. An earlier version of this verification has been published already in [Klu91a].

We will encounter some new concepts. If we have a process like

$$X(v) \overset{def}{=} \tau(v) \cdot \{(Y(v+2) + \tau(v+1) \cdot X(v+3)\}$$

one would like to be able prove that

$$X(v) = \tau(v) \cdot \{\sum_{n=0}^{\infty} \tau(v+1+3\cdot n) \cdot Y(v+2+3\cdot n)\}$$

The n in the summation of the second process term corresponds with the number of recursion loops in the first process term. This identity cannot be proven within the axiom systems we have seen so far. In the next section we will introduce the so called *Unwind Principle* by which we obtain the above identity.

If one is not interested any more in all internal moments of choice one could also argue that

$$a(1) \cdot (\tau(2) \cdot b(3) + b'(3)) = a(1) \cdot (b(3) + b'(3))$$

In Section 10.4 we introduce a so called τ-*erasing* bisimulation that identifies these two process terms. Of course, this equivalence is not a congruence over ACPρ, as these two terms can be distinguished by the context $\partial_{b,\bar{b}}(\ldots \| \bar{b}(3))$, if $\gamma(b, \bar{b}) = c_b$ and $\gamma(b', \bar{b}) = \delta$. This equivalence allows us to simplify the above identity for $X(v)$ into

$$X(v) = \tau(v) \cdot \{\sum_{n=0}^{\infty} Y(v + 2 + 3 \cdot n)\}$$

The main part of this chapter is devoted to an algebraic reasoning by which we can rewrite the implementation of the PAR protocol into an expression that still contains all external behavior and all possible internal moments of choice. With a little handwaving we use the τ-erasing bisimulation to get rid of all the internal moments of choice, and finally we sketch that the resulting expression is trace included by some more (time-)abstract expression.

10.2 The τ-swap and τ-removal

In the verification of this chapter we make use of two identities. One of them is the τ-swap, that allows to swap the τ from one summand to the other. For example

$$a(1) \cdot (\tau(2) \cdot b(3) + c(3)) \;=\; a(1) \cdot (b(3) + \tau(2) \cdot c(3))$$

The next one is the τ-removal, for example

$$a(1) \cdot (\tau(2) \cdot b(3) + c(2)) \;=\; a(1) \cdot (b(3) + c(2))$$

The formal definitions of these identities are given in Table 10.1.

τ-swap	$\int_{\alpha \wedge U_b(p) \wedge U_b(q)} a(v) \cdot (\tau(b) \cdot p + q) \;=\; \int_{\alpha \wedge U_b(p) \wedge U_b(q)} a(v) \cdot (p + \tau(b) \cdot q)$
τ-removal	$\int_{\alpha \wedge U_b(p) \wedge \neg(U_b(q))} a(v) \cdot (\tau(b) \cdot p + q) \;=\; \int_{\alpha \wedge U_b(p) \wedge \neg(U_b(q))} a(v) \cdot (p + q)$

Table 10.1: The τ-swap and τ-removal

10.3 The Unwind Principle

In Table 10.2 we formulate the so called *unwind principle*, that allows us to unwind a recursive specification infinitely many times. As motivation we give the following pseudo derivation.

$$\begin{array}{lrcl}
 & X(v) & \overset{def}{=} & \tau(v)\cdot\{Y(v+b)+\tau(v+b_0)\cdot X(v+b_1)\} \\
\text{UP} & Y(w) & = & w \gg Y(w), \qquad b_0 < \min(b_1, b) \Longrightarrow \\
 & X(v) & = & \tau(v)\cdot\{\sum_{n=0}^{\infty}\tau(v+b_0+n\cdot b_1)\cdot Y(v+b+n\cdot b_1)\}
\end{array}$$

Table 10.2: The Unwind Principle

$$\begin{array}{rcl}
X(v) & = & \tau(v)\cdot\{Y(v+b)+\tau(v+b_0)\cdot X(v+b_1)\} \\
 & = & \tau(v)\cdot\{\tau(v+b_0)\cdot Y(v+b)+X(v+b_1)\} \\
 & = & \tau(v)\cdot\{\tau(v+b_0)\cdot Y(v+b)+ \\
 & & \quad \tau(v+b_1)\cdot \\
 & & \quad \{Y(v+b+b_1)+\tau(v+b_0+b_1)\cdot X(v+2\cdot b_1)\}\} \\
 & = & \tau(v)\cdot\{\tau(v+b_0)\cdot Y(v+b)+ \\
 & & \quad \{Y(v+b+b_1)+\tau(v+b_0+b_1)\cdot X(v+2\cdot b_1)\}\} \\
 & = & \tau(v)\cdot\{\tau(v+b_0)\cdot Y(v+b)+ \\
 & & \quad \tau(v+b_0+b_1)\cdot Y(v+b+b_1)+X(v+2\cdot b_1)\} \\
 & . & \\
 & . & \\
 & . & \\
 & \overset{\text{UP}}{=} & \tau(v)\cdot\{\sum_{n=0}^{\infty}\tau(v+b_0+n\cdot b_1)\cdot Y(v+b+n\cdot b_1)\}
\end{array}$$

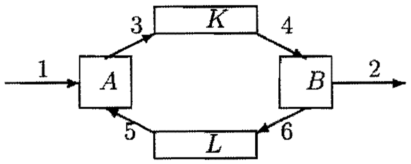
The dots in the derivation below express that this principle is not provable within the theory BPA$\rho\delta$I, or BPA$\rho\delta$I, in a finite derivation.

10.4 A τ Erasing Bisimulation

Definition 10.4.1 (τ-Erasing Bisimulation)
$\mathcal{R} \subseteq (\mathcal{T}\times[0,\infty))^2$ *is an* τ-erasing bisimulation *if whenever* $p\mathcal{R}q$ *then*

1. *$p \overset{a(r)}{\Longrightarrow} p'$ ($a \in A$) implies that there is a q' such that $q \overset{a(r)}{\Longrightarrow} q'$ and $p'\mathcal{R}q'$*
2. *$p \overset{a(r)}{\Longrightarrow} \surd$ ($a \in A_\delta$) implies that $q \overset{a(r)}{\Longrightarrow} \surd$.*
3. *$U_t(p)$ implies that there is a z such that $q \overset{\tau *}{\Longrightarrow} q'$ and $U_t(q')$.*
4. *Respectively (1), (2), (3) with the role of p and q interchanged.*

Recall that $q \overset{\tau *}{\Longrightarrow} q'$ means that there is a τ sequence of length zero or more from q to q'. And we have



Definition 10.4.2 *$p \leftrightarrows_{(r)e} q$ if there is a (rooted) erasing bisimulation $\mathcal{R}$ such that $p\mathcal{R}q$.*

$\leftrightarrows_e$ is an equivalence over BPA$\rho\delta_T$, but not a congruence. $\leftrightarrows_{re}$ is a congruence over BPA$\rho\delta_T$, but not over ACPρ.

We can formulate the following law:

$$\boxed{\begin{array}{ll} \alpha \;=\; (v < b_1 \wedge U_{b_1}(p)) & \\ \mathrm{T}_E \quad \int_{\alpha\wedge\beta} a(v) \cdot (\int_{w \in \langle\!\langle b_0, b_1 \rangle\!\rangle} \tau(w) \cdot p + q) \;=\; \int_{\alpha\wedge\beta} a(v) \cdot (b_0 \gg p + q) & \end{array}}$$

10.5 The Specification and the Implementation of the Protocol

First we define the individual components.

$$\begin{array}{lcl}
A & = & A_1(0,0) \\
A_1(b,v) & = & \sum_{d\in D} \int_{w>v} r_1(d)(w) \cdot A_2(b,d,w) \\
A_2(b,d,v) & = & s_3(db)(v+0.001) \cdot A_3(b,d,v) \\
A_3(b,d,v) & = & \int_{w\in[v+0.001,v+0.01)} r_5(ack)(w) \cdot A_1(1-b,w) + \\
 & & \qquad\qquad time\_out(v+0.01) \cdot A_2(b,d,v+0.01) \\
 & & \\
K & = & \sum_{f\in D\times B} \int_{w>0} r_3(f)(w) \cdot K'(w) \\
K'(v) & = & \{s_4(f)(v+0.002) + error_K(v+0.001)\} \cdot K \\
 & & \\
L & = & \int_{w>0} r_6(ack)(w) \cdot L' \\
L' & = & \{s_5(ack)(v+0.002) + error_L(v+0.001)\} \cdot L \\
 & & \\
B & = & B_1(0) \\
B_1(b) & = & B_2(b) + B_2'(b) \\
B_2(b) & = & \sum_{d\in D} \int_{w>0} r_4(db)(w) \cdot s_2(d)(w+0.001) \cdot B_3(1-b,w) \\
B_2'(b) & = & \sum_{d\in D} \int_{w>0} r_4(d(b-1))(w) \cdot B_3(b,w) \\
B_3(b,v) & = & s_6(ack)(v+0.002) \cdot B_1(b)
\end{array}$$

10.6 Expanding the Definitions

We expand the definitions, for each new configuration a new recursion variable is chosen. In this way we obtain the parameterized recursion variables $X_0 - X_2$, $Y_1 - Y_2$ and Z_1, Z_2.

$$\begin{array}{rcl}
\mathrm{PAR}_{impl} & = & X_0(0,0) \\
 & & \\
X_0(b,v) & = & \partial_H(A_1(b,v) \| K \| L \| B(b)) \\
 & = & \int_{w>v} \sum_{d\in D} r_1(d)(w) \cdot X_1(b,d,w)
\end{array}$$

$$\begin{array}{lll}
 & X_1(b,d,v) & \\
= & \partial_H \ (& A_2(b,d,v) \| \ K \ \| \ L \ \| \ B(b) \ \ ) \\
= & \partial_H \ (& \underline{s_3(db)(v+0.001)} \cdot A_3(b,d,v) \\
 & \| & \sum_{f\in D\times B} \int_{w>0} \underline{r_3(f)(w)} \cdot K' \\
 & \| & L \\
 & \| & B_1(b) \\
 &) & \\
= & c_3(db)(v+0.001) \cdot X_2(b,d,v) &
\end{array}$$

$$\begin{array}{lll}
 & X_2(b,d,v) & \\
= & \partial_H \ (& [\int_{w\in[v,v+0.01)} r_5(ack,w) \cdot A_1(1-b,w) + \\
 & & \quad \underline{time\_out(v+0.01)} \cdot A_3(b,d,v+0.01)] \\
 & \| & [\underline{s_4(db)(v+0.003)} + \underline{error_K(v+0.002)}] \cdot K \\
 & \| & L \\
 & \| & \sum_{d\in D} \int_{w>0} \underline{r_4(db)(w)} \cdot \underline{s_2(d)(w+0.001)} \cdot B(1-b,w) \ + \ B_2'(b) \\
 &) & \\
= & c_4(db)(v+0.003) \cdot s_2(d)(v+0.004) \cdot Z_1(b,d,v) + & \\
 & \quad error_K(v+0.002) \cdot time\_out(v+0.01) \cdot X_1(b,d,v+0.01) &
\end{array}$$

$$\begin{array}{lll}
 & Z_1(b,d,v) & \\
= & \partial_H \ (& A_3(b,d,v+0.001) \\
 & \| & K \\
 & \| & \int_{w>0} \underline{r_6(ack)(w)} \ \cdot \ L' \\
 & \| & \underline{s_6(ack)(v+0.005)} \cdot B(1-b) \\
 &) & \\
= & c_6(ack)(v+0.005) \cdot Z_2(b,d,v) &
\end{array}$$

$$\begin{array}{lll}
 & Z_2(b,d,v) & \\
= & \partial_H \ (& [\int_{w\in[v,v+0.01)} \underline{r_5(ack,w)} \cdot A_1(1-b,w) + \\
 & & \quad \underline{time\_out(v+0.01)} \cdot A_2(b,d,v+0.01)] \\
 & \| & K
\end{array}$$

$$
\begin{array}{lll}
 & \| & [\underline{s_5(ack)(v+0.007)} + \underline{error_L(v+0.006)}] \cdot L \\
 & \| & B_1(1-b) \\
 &) & \\
= & \multicolumn{2}{l}{c_5(ack)(v+0.007) \cdot X_0(1-b, v+0.007) +} \\
 & \multicolumn{2}{l}{\quad error_L(v+0.006) \cdot time\_out(v+0.01) \cdot Y_1(b,d,v+0.01)}
\end{array}
$$

$$
\begin{array}{lll}
 & \multicolumn{2}{l}{Y_1(b,d,v)} \\
= & \partial_H \; (& A_2(b,d,v) \| K \| L \| B(1-b) \;) \\
= & \partial_H \; (& \underline{s_3(db)(v+0.001)} \cdot A_3(b,d,v) \\
 & \| & \sum_{f\in D\times B} \int_{w>0} \underline{r_3(f)(w)} \\
 & & \quad \cdot [s_4(f)(w+0.002) + error_K(w+0.001)] \cdot K \\
 & \| & L \\
 & \| & B_1(1-b) \\
 &) & \\
= & \multicolumn{2}{l}{c_3(db)(v+0.001) \cdot Y_2(b,d,v)}
\end{array}
$$

$$
\begin{array}{lll}
 & \multicolumn{2}{l}{Y_2(b,d,v)} \\
= & \partial_H \; (& [\int_{w\in[v,v+0.01)} r_5(ack,w) \cdot A_1(1-b,w) + \\
 & & \quad \underline{time\_out(v+0.01)} \cdot A_2(b,d,v+0.01)] \\
 & \| & [\underline{s_4(f)(v+0.003)} + \underline{error_K(v+0.002)}] \cdot K \\
 & \| & L \\
 & \| & \sum_{d\in D} \int_{w>0} \underline{r_4(d(1-b))(w)} \cdot \underline{s_2(d)(w+0.001)} \cdot B_2(b,w) + \\
 & & \quad B_2'(1-b) \\
 &) & \\
= & \multicolumn{2}{l}{c_4(db)(v+0.003) \cdot Z_1(b,d,v) +} \\
 & \multicolumn{2}{l}{\quad error_K(v+0.002) \cdot time\_out(v+0.01) \cdot Y_1(b,d,v+0.01)}
\end{array}
$$

10.7 Abstracting from Internal Steps

We apply the renaming operator τ_I which renames every atomic action $a(v)$ to $\tau(v)$ except for the actions $r_1(d)(v)$ and $s_2(d)(v)$.

$$
\begin{array}{lll}
\tau_I(X_0(b,v)) & = & \int_{w>v} \sum_{d\in D} r_1(d)(w) \cdot \tau_I(X_1(b,d,w)) \\
\tau_I(X_1(b,d,v)) & = & \tau(v+0.001) \cdot \tau_I(X_2(b,d,v)) \\
\tau_I(X_2(b,d,v)) & = & \tau(v+0.003) \cdot s_2(d)(v+0.004) \cdot \tau_I(Z_1(b,d,v)) + \\
 & & \tau(v+0.002) \cdot \tau(v+0.01) \cdot \tau_I(X_1(b,d,v+0.01)) \\
\tau_I(Z_1(b,d,v)) & = & \tau(v+0.005) \cdot \tau_I(Z_2(b,d,v)) \\
\tau_I(Z_2(b,d,v)) & = & \tau(v+0.007) \cdot \tau_I(X_0(1-b,v+0.007)) + \\
 & & \tau(v+0.006) \cdot \tau(v+0.01) \cdot \tau_I(Y_1(b,d,v+0.01))
\end{array}
$$

$$
\begin{array}{lll}
\tau_I(Y_1(b,d,v)) & = & \tau(v+0.001) \cdot \tau_I(Y_2(b,d,v)) \\
\tau_I(Y_2(b,d,v)) & = & \tau(v+0.003) \cdot \tau_I(Z_1(b,d,v)) + \\
 & & \tau(v+0.002) \cdot \tau(v+0.01) \cdot \tau_I(Y_1(b,d,v+0.01))
\end{array}
$$

Now we can apply the τ-law and its implied identities (such as the τ-swap and the τ-removal).

$$\begin{aligned}
\tau_I(X_1(b,d,v)) &= \tau(v+0.001) \cdot \tau_I(X_2(b,d,v)) \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ \underline{\tau(v+0.003)} \cdot s_2(d)(v+0.004) \cdot \tau_I(Z_1(b,d,v))\ + \\
&\quad\ \ \tau(v+0.002) \cdot \underline{\tau(v+0.01)} \cdot \tau_I(X_1(b,d,v+0.01))\ \} \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ s_2(d)(v+0.004) \cdot \underline{\tau(v+0.005)} \cdot \tau_I(Z_2(b,d,v))\ + \\
&\quad\ \ \tau(v+0.002) \cdot \tau_I(X_1(b,d,v+0.01))\ \} \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ s_2(d)(v+0.004) \cdot \\
&\quad\quad \{\ \underline{\tau(v+0.007)} \cdot \tau_I(X_0(1-b,t+0.007)) + \\
&\quad\quad\quad \tau(v+0.006) \cdot \underline{\tau(v+0.01)} \cdot \tau_I(Y_1(b,d,v+0.01))\ \}\ + \\
&\quad\quad \tau(v+0.002) \cdot \tau_I(X_1(b,d,v+0.01))\ \} \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ s_2(d)(v+0.004) \cdot \\
&\quad\quad \{\ \tau_I(X_0(1-b,t+0.007)) + \\
&\quad\quad\quad \tau(v+0.006) \cdot \tau_I(Y_1(b,d,v+0.01))\ \} + \\
&\quad\quad \tau(v+0.002) \cdot \tau_I(X_1(b,d,v+0.01))\ \}
\end{aligned}$$

$$\begin{aligned}
\tau_I(Y_1(b,d,v)) &= \tau(v+0.001) \cdot \tau_I(Y_2(b,d,v)) \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ \underline{\tau(v+0.003)} \cdot \tau_I(Z_1(b,d,v)) + \\
&\quad\ \ \tau(v+0.002) \cdot \underline{\tau(v+0.01)} \cdot \tau_I(Y_1(b,d,v+0.01))\ \} \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ \underline{\tau(v+0.005)} \cdot \tau_I(Z_2(b,d,v)) + \\
&\quad\ \ \tau(v+0.002) \cdot \tau_I(Y_1(b,d,v+0.01))\ \} \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ \{\ \underline{\tau(v+0.007)} \cdot \tau_I(X_0(1-b,t+0.007)) + \\
&\quad\quad\quad \tau(v+0.006) \cdot \underline{\tau(v+0.01)} \cdot \tau_I(Y_1(b,d,v+0.01))\ \} + \\
&\quad\quad \tau(v+0.002) \cdot \tau_I(Y_1(b,d,v+0.01))\ \} \\
&= \tau(v+0.001) \cdot \\
&\quad \{\ \tau_I(X_0(1-b,t+0.007)) + \\
&\quad\quad\quad \tau(v+0.006) \cdot \tau_I(Y_1(b,d,v+0.01)) + \\
&\quad\quad\quad \tau(v+0.002) \cdot \tau_I(Y_1(b,d,v+0.01))\ \}
\end{aligned}$$

By applying the *Unwind Principle*:

$$\begin{aligned}
\tau_I(X_1(b,d,v)) = {}& \tau(v+0.001) \cdot \\
& \textstyle\sum_{n=0}^{\infty} \tau(v+0.002+n\cdot 0.01) \cdot \\
& s_2(d)(v+0.004+n\cdot 0.01) \cdot \\
& \{\ \tau_I(X_0(1-b,v+0.007+n\cdot 0.01)) + \\
& \quad \tau(v+0.006+n\cdot 0.01) \cdot \tau_I(Y_1(b,d,v+(n+1)\cdot 0.01))\ \}
\end{aligned}$$

$$\begin{aligned}\tau_I(Y_1(b,d,v)) \quad = \quad & \tau(v+0.001)\cdot \\ & \textstyle\sum_{n=0}^{\infty}\tau(v+0.002+n\cdot 0.01) \\ & \{\tau_I(X_0(1-b,v+0.007+n\cdot 0.01)) + \\ & \quad \tau(v+0.006)\cdot\tau_I(Y_1(b,d,v+0.01))\}\end{aligned}$$

If we abstract from all internal activity we come to the following sequence:

Read the data at port 1	1
It takes n time outs before it is delivered at the sender	2
The data is sent over port 2	3
Either the system is back in its starting position	4a
or another round is needed for the acknowledgement	4b

An "acknowledgement round" is similar, though no read at port 1 an send at port 2 occur. Below we give a more formal presentation of this high level view. We define $Q(b,v)$ and $Q'(b,v)$ as follows. $Q'(b,v)$ is the "acknowledgement round".

$$\begin{aligned}Q(b,v) \quad = \quad & \textstyle\int_{w>v}\sum_{d\in D} r_1(d)(w)\cdot & 1\\ & \textstyle\sum_{n=0}^{\infty}\tau(w+0.002+n\cdot 0.01)\cdot & 2\\ & s_2(d)(w+0.004+n\cdot 0.01)\cdot & 3\\ & \{Q(1-b,w+0.007+n\cdot 0.01)+ & 4a\\ & \quad \tau(w+0.006+n\cdot 0.01)\cdot Q'(b,w+0.006+n\cdot 0.01)\} & 4b\end{aligned}$$

$$\begin{aligned}Q'(b,v) \quad = \quad & \tau(v+0.001)\cdot\textstyle\sum_{n=0}^{\infty}\tau(v+0.006+n\cdot 0.01)\cdot \\ & \{Q(1-b,v+0.007+n\cdot 0.01)+ \\ & \quad \tau(v+0.006+n\cdot 0.01)\cdot Q'(b,v+0.006+n\cdot 0.01)\}\end{aligned}$$

If we take

$$\mathrm{PAR}_{spec}^{int-choice} = Q(0,0)$$

then we can prove

$$\mathrm{ACP}\rho + \mathrm{B}_\rho + \mathrm{RSP} \vdash \mathrm{PAR}_{spec}^{int-choice} = \mathrm{PAR}_{impl}$$

10.8 Some Tougher Methods and Handwavings

The definition of $\mathrm{PAR}_{spec}^{int-choice}$ still contains all internal moments of choice, hence the suffix. At this point in our verification we are not interested any more in these moments and we define

$$\mathrm{PAR}_{spec}^{ext-tim} = P(0,0)$$

where

$$\begin{aligned}
P(b,v) &= \textstyle\int_{w>v}\sum_{d\in D} r_1(d)(w)\cdot \\
&\quad \textstyle\sum_{n=0}^{\infty} s_2(d)(w+0.004+n\cdot 0.01)\cdot \\
&\quad \{P(1-b,w+0.007+n\cdot 0.01)+ \\
&\qquad P'(b,w+0.006+n\cdot 0.01)\} \\
\\
P'(b,v) &= \textstyle\sum_{n=0}^{\infty}\{P(1-b,v+0.007+n\cdot 0.01)+ \\
&\qquad\qquad P'(b,v+0.006+n\cdot 0.01)\}
\end{aligned}$$

We have obtained a description that contains all possible timings which are observable by external actions. If we allow ourselves the freedom to do some handwaving by which we can apply the axiom E infinitely many times then we obtain that $\mathrm{PAR}_{spec}^{int-choice} = \mathrm{PAR}_{spec}^{ext-tim}$.

Finally we define

$$S(b,v) = \int_{w_0>v}\sum_{d\in D} r_1(d)(w_0)\cdot\int_{w_1>w_0} s_2(d)(w_1)\cdot S(b,w_1)$$

We take $\mathrm{PAR}_{spec} = S(0,0)$, and we argue that all traces of $\mathrm{PAR}_{spec}^{ext-tim}$ are traces of PAR_{spec} as well.

$$\mathrm{PAR}_{spec}^{ext-tim} \leq_{trace} \mathrm{PAR}_{spec}$$

Of course the reasoning at the end of this chapter is not very precise and formal. If protocol verification in real time process algebra is to be used, then we expect that the *Unwind Principle*, τ-erasing bisimulation, dealing with infinite sums and preorders, can be of use. However, these concepts have to be studied in much more detail, which is subject for further research.

Part V

Urgent Actions and Related Work

11

Real Time ACP with Urgent Actions

11.1 Introduction

In this chapter we introduce a variant of Real Time ACP by introducing so-called *urgent actions* in a relative time setting. Urgent actions are actions that may be executed consecutively at the same point in time. Note that this is not the case in ACPρ where $a(1) \cdot b(1)$ equals $a(1) \cdot \delta$. We refer to this variant by ACPur, the u stands for urgent actions and the r for relative time. The motivation for this variant is that other timed process calculi have also urgent actions in a relative time setting. In the next chapter we will show how several other time calculi can be expressed in ACPur.

This chapter can, in principle, be read independently of the previous ones. For some definitions, however, we refer explicitly to the sections where they can be found.

ACPur consists of the ingredients given below.

- Relative time; the time stamps of the actions are interpreted relatively to the time of execution of the previous action, or 0 in case of an initial action. Relative time has already been introduced in real time ACP by Baeten and Bergstra in [BB91], where they use square brackets.

- Urgent actions; we assume that for each (symbolic) action a we have its urgent variant, denoted by $\tilde{a}$. The difference between $\tilde{a}[t]$ and $a[t]$ is that the first action idles until t (t included) after which it executes the $\tilde{a}$ action without taking any time. $a[t]$, however, idles till t (t excluded) and the execution of the a action coincides with the proceeding to point 1 in time. Another example, $\tilde{b}[0]$ means that $\tilde{b}$ has to be executed immediately, while $b[0]$ equals a deadlock at time 0, since it cannot do the b nor can it idle.

- Prefixed multiplication; as constants we have timed deadlocks only.

- The standard operators $+, \|, \mathbin{\|\!\_}, |, \partial_H$ and ρ_f. The operator ρ_f has not been discussed in the previous chapters on parallelism and synchronization, though it is a standard operator from the literature. The symbol f denotes a mapping from atomic actions to atomic actions, and the operator ρ_f takes a process term p and applies this mapping f on all the atomic actions of p.

- Two new operators.

  - The *shift*-operator, $(b)\cdot p$. If $r > 0$ then $(r)\cdot p$ is the process that becomes p after idling r time units, if $r < 0$ then $(r)\cdot p$ is the process which is reached after p has idled r time units, and finally, $(0)\cdot p$ equals p. For example:

    $$\begin{array}{rcl} (2)\cdot(\int_{v\in[5,6]}\tilde{a}[v]\cdot\tilde{b}[v+3]) & = & \int_{v\in[7,8]}\tilde{a}[v]\cdot\tilde{b}[v+1] \\ (-2)\cdot(\int_{v\in[5,6]}\tilde{a}[v]\cdot\tilde{b}[v+3]) & = & \int_{v\in[3,4]}\tilde{a}[v]\cdot\tilde{b}[v+5] \end{array}$$

    It is used, among other places, in the axiomatization of the left merge, for example

    $$\tilde{a}[1]\mathbin{\|\!\_}\tilde{b}[3] = \tilde{a}[1]\cdot(-1)\cdot\tilde{b}[3] = \tilde{a}[1]\cdot\tilde{b}[2].$$

    The notation $(b)\cdot p$ originates from Moller & Tofts [MT90], for a time element t they have $(t).p$. The shift operator can also be found in the work of Chen [Che93]. The axioms of $(b)\cdot p$ in this chapter resemble the ones of Chen, though Chen does not give action rules for it.

    Hennessy & Regan [HR90] have introduced a similar construct in discrete time, they denote $(1)\cdot p$ by $\sigma(p)$. Baeten & Bergstra [BB92] have also defined a shift operator, for which they use the notation of Hennesy & Regan. They denote $(r)\cdot p$ by $\sigma_r(p)$. We have chosen to use the notation $(b)\cdot p$ of Moller & Tofts and not the notation $\sigma_b(p)$ of Baeten & Bergstra, as the σ denotes already an arbitrary substitution.

  - In the next chapter we encounter several timed process calculi that assume *maximal progress*, i.e., a process cannot idle any more after the internal action τ has been enabled.

    For maximal progress in the context of real time process algebra we refer to Wang. Nicollin & Sifakis [NS91] have introduced *action urgency*, and they can consider maximal progress of an instantiation, namely τ-urgency. Bolognesi & Lucidi [BL91] have defined an *urgency* operator $\rho(H)p$ in a discrete time context. The operator $\rho(H)$ makes all actions in H urgent in its argument. We define this operator as well in our more general setting. Since we associate the symbol ρ already with renaming, we denote the urgency operator by $\mathcal{U}_H$. So, as soon as a the action $\tilde{a}$, with $a \in H$, is enabled in p then $\mathcal{U}_H(p)$ cannot idle any more.

    For example

$$\begin{aligned}
\mathcal{U}_{\{a\}}(\textstyle\int_{v\in[2,3]}\tilde{a}[v]\cdot p + \int_{v\in\langle 1,4]}\tilde{b}[v]\cdot q) &= \\
&\textstyle\int_{v\in[2,2]}\tilde{a}[v]\cdot p + \int_{v\in\langle 1,2]}\tilde{b}[v]\cdot q \\
\mathcal{U}_{\{a\}}(\textstyle\int_{v\in\langle 2,3]}\tilde{a}[v]\cdot p) &= \tilde{\delta}[2]
\end{aligned}$$

The second identity shows us that the urgency operator may introduce a deadlock. This observation can also be found in a paper of Jeffrey [Jef91c], in that paper deadlocks are called time-stops.

The urgency operator can be axiomatized like the priority operator of Baeten and Bergstra [BB93a], though we will use a slightly different axiomatization.

The semantics for ACP*ur* has a so-called *two phase* pattern [NS91]; it has time phases and action phases. In a time phase all components agree in synchronizing in idling, that is, the whole process idles for a finite or infinite amount of time such that for each pair of points in the time interval there is a connecting idle transition. In an action phase the components execute their actions, either independently or by synchronization. An action phase does not take time, the behavior of the different components in an action phase is very similar to the behavior in untimed ACP. In short, a two phase semantics has timed transitions, which increase the time, and (untimed) action transitions. Other examples of two phase semantics can be found in the Timed CCS calculi of Wang, Moller & Tofts and Chen, and it can also be found in ATP of Nicollin & Sifakis.

We axiomatize ACP*ur* by adapting the axioms from ACPρI. Furthermore, we present branching, delay and weak bisimulation in the context of ACP*ur* and we discuss the differences between branching bisimulation in a two phase semantics with the one of Chapter 6.

11.2 Syntax Definitions

We take the definitions of bounds and intervals of Chapter 4, Section 4.2. We recall that a bound is a time expression, that is a linear expression over time variables. The set of bounds is denoted by *Bound*, a typical bound is denoted by b. The symbols $-\infty$ are ∞ are not part of *Bound*, we denote $\mathit{Bound} \cup \{-\infty, \infty\}$ by $\mathit{Bound}_{-\infty,\infty}$.

A condition is a boolean expression over time variables. Atomic conditions are of the form $tt, \mathit{ff}, b_0 < b_1$ and $b_0 = b_1$. Furthermore, we have the operators $\wedge, \vee$ and $\neg$. The set of conditions is denoted by *Cond*, a typical condition is denoted by α. We extend *Cond* to Cond_U by allowing conditions of the form $U_b(p)$ that corresponds with the ultimate delay; the condition $U_b(p)$ reduces to true if p can idle till b. In Chapter 4 we have defined a predicate $\models$ on conditions, intuitively $\models \alpha$ whenever α reduces to tt.

The set of variables that occur in a bound b or a condition α is denoted by $var(b)$ and $var(\alpha)$ respectively. A substitution is a mapping from time variables to bounds. The set of substitutions is denoted by Σ, a typical substitution is denoted by σ. The set of time closed substitutions, that is the set of σ such that for any time variable v we have $var(\sigma(v)) = \emptyset$ is denoted by Σ^{cl}.

We have $\llparenthesis$ ranging over $\{\langle, [\}$ and $\rrparenthesis$ ranging over $\{\rangle,]\}$. An interval, typically V or W, is an expression of the form $\llparenthesis b_0, b_1 \rrparenthesis$ where $b_0, b_1 \in Bound_{-\infty,\infty}$. We have the additional requirement that $b_1 \neq -\infty$, and that $b_1 = \infty$ implies that $\rrparenthesis = \rangle$. Similarly, $b_0 \neq \infty$ and finally $b_0 = -\infty$ implies that $\llparenthesis = \langle$. We introduce the abbreviation $V+b$ that expresses that b is added to the bounds of V.

$$\llparenthesis b_0, b_1 \rrparenthesis + b \overset{abb}{=} \llparenthesis b_0 + b, b_1 + b \rrparenthesis$$

Furthermore, we abbreviate $V + (-1) \cdot b$ by $V - b$.

With respect to conditions we allow ourselves several abbreviations and expressions that denote conditions. For example, $b < \infty$ denotes tt and $v \in [b_0, b_1\rangle$ (where $[b_0, b_1\rangle$ is an interval) abbreviates $b_0 \leq v \wedge v < b_1$. So, $v \in [b, \infty\rangle$ abbreviates $b < v \wedge v < \infty$, which denotes in turn $b < v \wedge tt$, which can finally be reduced to $b < v$.

The set of process terms over ACPur is denoted by $T(\text{ACP}ur)$, and it is defined by the following BNF sentence, where $a \in A_\delta$, $b \in Bound$, $\alpha \in Cond_U$, $\sigma \in \Sigma$, $H \subseteq A$, and furthermore f is a mapping from A to A.

$$\begin{array}{rl} p ::= & \int_\alpha \tilde{\delta}[v] \mid \int_\alpha \tilde{a}[v] \cdot p \mid p+p \mid \alpha :\rightarrow p \mid \sigma(p) \mid (b) \cdot p \mid p \gg V \\ & p \| p' \mid p \llfloor p' \mid p | p' \mid \partial_H(p) \mid \rho_f(p) \mid \mathcal{U}_H(p) \mid p \lhd_H p \end{array}$$

The operator $p \gg V$ can be considered as a generalization of $p \gg b$ and $b \gg p$; the process term $p \gg V$ behaves as p, restricted to the interval V. We have an auxiliary operator $\lhd_H$ that is used in the axiomatization of $\mathcal{U}_H$.

Process terms of the form $\int_\alpha \tilde{\delta}[v] \cdot p$ are redundant, as they are equal to $\int_\alpha \tilde{\delta}[v]$. Hence, they can be removed from the set of process terms without any problems. We allow these terms as they simplify the axiomatization; without process terms $\int_\alpha \tilde{\delta}[v] \cdot p$ one has to give axioms for the case $\int_\alpha \tilde{a}[v] \cdot p$ (where $a \in A$), and also for the case $\int_\alpha \tilde{\delta}[v]$.

In examples we allow ourselves to abbreviate $\int_\alpha \tilde{a}[v] \cdot \int_{tt} \tilde{\delta}[w]$ by $\int_\alpha \tilde{a}[v]$. So, $\tilde{a}[1]$ abbreviates $\int_{v=1} \tilde{a}[v] \cdot \int_{tt} \tilde{\delta}[w]$. We allow ourselves the following abbreviations:

$$\begin{array}{rcl} \tilde{\delta} & \overset{abb}{=} & \int_{ff} \tilde{\delta}[v] \\ \tilde{a}[b] & \overset{abb}{=} & \int_{v=b} \tilde{a}[v] \end{array}$$

We have also

$$U_b(\llparenthesis b_0, b_1 \rrparenthesis) \overset{abb}{=} \llparenthesis b_0, b_1 \rrparenthesis \neq \emptyset \wedge b \in \langle -\infty, b_1 \rrparenthesis$$

So, for some $t \in Time$ the expression $U_t(\langle 1, 3 \rangle)$ abbreviates $\langle 1, 3 \rangle \neq \emptyset \wedge t \in \langle -\infty, 3 \rangle$, which denotes in turn the condition $\langle 1, 3 \rangle \neq \emptyset \wedge t < 3$, that reduces finally to $t < 3$. This latter condition can be reduced to tt or ff, depending on t.

In $\int_\alpha \tilde{a}[v] \cdot p$ all free occurrences of the time variable v become bound by the integral $\int_\alpha$. We denote the set of free time variables of a process term p by $fv(p)$.

The formal definition of $fv(p)$ is omitted, as it corresponds closely to the definition of free variables for terms in $T(\mathrm{ACP}\rho\mathrm{I})$, as given in Subsection 4.3.3 and Section 5.1. Recall that a term without free time variables is *time closed*, otherwise it is *time open*.

11.3 A Two Phase Operational Semantics

In Table 11.1 and Table 11.2 we give the action rules for the two phase semantics for ACPur. The inference rules for the evaluation of time closed conditions, and the action rules for time closed terms with substitutions, have been omitted.

The rule

$$\frac{\models t > 0 \qquad \models U_t(V)}{\int_{v\in V} \tilde{a}[v]\cdot p \xrightarrow{t} \int_{v\in V-t} \tilde{a}[v]\cdot p[v+t/v]}$$

says that the bounds of an interval can be considered as timers which decrease in time. As soon as the timer of the lower bound becomes 0 it remains 0. Whenever $0 \in V$ then $\int_{v\in V} \tilde{a}[v]\cdot p$ can execute an a and evolve into $p[o/v]$, which is expressed by the rule

$$\frac{\models 0 \in V}{\int_{v\in V} \tilde{a}[v]\cdot p \xrightarrow{\tilde{a}} p[0/v]}$$

Consider the process term $\int_{v\in\langle 4,6\rangle} \tilde{a}[v]\cdot\tilde{b}[v+1]$. We expect that if $\tilde{a}$ is executed after idling 5 time units, then the process will evolve into $\tilde{b}[6]$. In order to obtain this formally we have to substitute $v+t$ for v in the first rule above and 0 for v in the second rule above.

$$\int_{v\in\langle 4,6\rangle} \tilde{a}[v]\cdot\tilde{b}[v+1] \xrightarrow{5} \int_{v\in[0,1\rangle} \tilde{a}[v]\cdot\tilde{b}[v+6] \xrightarrow{\tilde{a}} \tilde{b}[6]$$

Similar rules can be found in the work of Wang, Chen and others.

The process term $\mathcal{U}_H(p)$ can idle t time units if it cannot perform an action $\tilde{a}$, with $a \in H$, before t. This latter requirement is expressed by the negative premise $\forall a \in H\ \forall r < t\ p \overset{\tilde{a}[r]}{\not\longrightarrow}$. For this reason, we have to introduce an auxiliary transition relation $p \xrightarrow{\tilde{a}[r]} p'$, that is defined easily by the last rule in Table 11.2.

The definition of a bisimulation and of bisimulation equivalence is completely straightforward.

Definition 11.3.1 (Bisimulation)
$\mathcal{R} \subset T^{cl}(\mathrm{ACP}ur) \times T^{cl}(\mathrm{ACP}ur)$ *is a* bisimulation *if whenever* $p\mathcal{R}q$ *then*

1. $p \xrightarrow{\mu} p'$ $(\mu \in A \cup \langle 0,\infty\rangle)$ *implies* $\exists q'$ *such that* $q \xrightarrow{\mu} q'$ *and* $p'\mathcal{R}q'$.

2. $q \xrightarrow{\mu} q'$ $(\mu \in A \cup \langle 0,\infty\rangle)$ *implies* $\exists p'$ *such that* $p \xrightarrow{\mu} p'$ *and* $p'\mathcal{R}q'$.

$$\frac{\models t > 0 \quad \models U_t(V)}{\int_{v\in V} \tilde{a}_\delta[v] \cdot p \xrightarrow{t} \int_{v\in V-t} \tilde{a}_\delta[v] \cdot p[v+t/v]}$$

$$\frac{\models 0 \in V}{\int_{v\in V} \tilde{a}[v] \cdot p \xrightarrow{\tilde{a}} p[0/v]} \qquad \frac{\models t > 0 \quad \models U_t(V)}{\int_{v\in V} \tilde{\delta}[v] \xrightarrow{t} \int_{v\in V-t} \tilde{\delta}[v]}$$

$$\frac{\models \alpha = \beta \quad \int_\beta P[v] \xrightarrow{\mu} p'}{\int_\alpha P[v] \xrightarrow{\mu} p'}$$

$$\frac{p \xrightarrow{t} p' \quad q \xrightarrow{t} q'}{p + q \xrightarrow{t} p' + q'} \qquad \frac{p \xrightarrow{t} p' \quad q \not\xrightarrow{t}}{p + q \xrightarrow{t} p' \,,\; q + p \xrightarrow{t} p'}$$

$$\frac{p \xrightarrow{\tilde{a}} p'}{p + q \xrightarrow{\tilde{a}} p' \,,\; q + p \xrightarrow{\tilde{a}} p'} \qquad \frac{\models \alpha \quad p \xrightarrow{\mu} p'}{\alpha :\!\longrightarrow p \xrightarrow{\mu} p'}$$

$$(t + r) \cdot p \xrightarrow{t} (r) \cdot p \qquad \frac{p \xrightarrow{\mu} p'}{(0) \cdot p \xrightarrow{\mu} p'}$$

$$\frac{p \xrightarrow{r} p' \quad p' \xrightarrow{\mu} p''}{(-r) \cdot p \xrightarrow{\mu} p''} \qquad \frac{(t) \cdot p \xrightarrow{\mu} p' \quad \models r = t}{(r) \cdot p \xrightarrow{\mu} p'}$$

$$\frac{p \xrightarrow{t} p' \quad \models U_t(V)}{p \gg V \xrightarrow{t} p' \gg V - t} \qquad \frac{p \xrightarrow{a} p' \quad \models 0 \in V}{p \gg V \xrightarrow{a} p'}$$

$$(a \in A,\; a_\delta \in A_\delta,\; \mu \in A \cup \langle 0, \infty \rangle)$$

Table 11.1: Two phase action rules for BPA$ur\delta$

$$\frac{p \xrightarrow{t} p' \quad q \xrightarrow{t} q'}{p \Box q \xrightarrow{t} p' \Box q'} \qquad (\Box \in \{\|, \lfloor\!\lfloor, |\})$$

$$\frac{p \xrightarrow{\tilde{a}} p'}{p \| q \xrightarrow{\tilde{a}} p' \| q} \qquad \frac{\begin{array}{c} c = \gamma(a, b) \neq \delta \\ p \xrightarrow{\tilde{a}} p' \quad q \xrightarrow{\tilde{b}} q' \end{array}}{p \| q \xrightarrow{\tilde{c}} p' \| q'}$$

$$\frac{p \xrightarrow{\tilde{a}} p'}{p \lfloor\!\lfloor q \xrightarrow{\tilde{a}} p' \| q} \qquad \frac{\begin{array}{c} c = \gamma(a, b) \neq \delta \\ p \xrightarrow{\tilde{a}} p' \quad q \xrightarrow{\tilde{b}} q' \end{array}}{p | q \xrightarrow{\tilde{c}} p' \| q'}$$

$$\frac{p \xrightarrow{t} p'}{\partial_H(p) \xrightarrow{t} \partial_H(p')} \qquad \frac{p \xrightarrow{t} p'}{\rho_f(p) \xrightarrow{t} \rho_f(p')}$$

$$\frac{p \xrightarrow{\tilde{a}} p' \quad a \notin H}{\partial_H(p) \xrightarrow{\tilde{a}} \partial_H(p')} \qquad \frac{p \xrightarrow{\tilde{a}} p'}{\rho_f(p) \xrightarrow{\widetilde{f(a)}} \rho_f(p')}$$

$$\frac{p \xrightarrow{t} p' \quad \forall a \in H \; \forall r < t \; p \overset{\tilde{a}[r]}{\not\longrightarrow}}{\mathcal{U}_H(p) \xrightarrow{t} \mathcal{U}_H(p')} \qquad \frac{p \xrightarrow{\tilde{a}} p' \quad a \notin H}{\mathcal{U}_H(p) \xrightarrow{\tilde{a}} \mathcal{U}_H(p')}$$

$$\frac{\begin{array}{c} p \xrightarrow{t} p' \quad q \xrightarrow{t} q' \\ \forall a \in H \; \forall r < t \; q \overset{\tilde{a}[r]}{\not\longrightarrow} \end{array}}{p \lhd_H q \xrightarrow{t} p' \lhd_H q'} \qquad \frac{\begin{array}{c} p \xrightarrow{t} p' \quad q \overset{t}{\not\longrightarrow} \\ \forall a \in H \; \forall r < t \; q \overset{\tilde{a}[r]}{\not\longrightarrow} \end{array}}{p \lhd_H q \xrightarrow{t} \mathcal{U}_H(p')}$$

$$\frac{p \xrightarrow{\tilde{a}} p'}{p \lhd_H q \xrightarrow{\tilde{a}} \mathcal{U}_H(p')}$$

$$\frac{p \xrightarrow{t} p' \quad p' \xrightarrow{\tilde{a}} p''}{p \xrightarrow{\tilde{a}[t]} p''} \qquad \frac{p \xrightarrow{\tilde{a}} p'}{p \xrightarrow{\tilde{a}[0]} p'}$$

Table 11.2: Two phase action rules for ACP*ur*

Bisimulation equivalence is now defined by

Definition 11.3.2 (Bisimulation Equivalence) $p, q \in T^{cl}(\mathrm{ACP}ur)$
$p \leftrightarrows q$ iff there is a bisimulation $\mathcal{R}$ relating p and q.

We generalize $\leftrightarrows$ to time open terms by taking $p \leftrightarrows q$ if for any $\sigma \in \Sigma^{cl}$ we have $\sigma(p) \leftrightarrows \sigma(q)$. Without proof we state that $\leftrightarrows$ is a congruence over $T^{cl}(\mathrm{ACP}ur)$, the proof is similar to the one in Chapter 4. First one has to give a semantics analogous to the so-called Σ-semantics of Chapter 4, and one has to prove that both bisimulation equivalences coincide. In the operational semantics we have also the auxiliary transition relation $\xrightarrow{\tilde{a}[r]}$; the path format result of Baeten & Verhoef tells us that bisimulation equivalence in which related states must also have corresponding $\xrightarrow{\tilde{a}[r]}$ is a congruence. For the negative premises we use the generalized format that is discussed by Verhoef in [Ver93a]. So, one has to show as well that this extra requirement does not change the bisimulation equivalence.

11.4 The Axiom System ACPur

In the Tables 11.3 and 11.4 the axioms for ACPur are given. The axioms for substitutions and α-conversion are not given here, they can easily be obtained by adapting the axioms of Table 4.5.1.

Most of the axioms are adapted ones from BPA$\rho\delta$I and ACPρI. In BPA$\rho\delta$I neither $\int_{v\in\langle 1,2\rangle} \delta(v)$ nor $\int_{v\in\langle 1,2]} \delta(v)$ can reach time 2. In other words we have:

$$\begin{array}{lcl} U_v(\int_{v\in\langle 1,2\rangle} \delta(v)) & = & v < \sup(\langle 1,2\rangle) & = & v < 2 \\ U_v(\int_{v\in\langle 1,2]} \delta(v)) & = & v < \sup(\langle 1,2]) & = & v < 2 \end{array}$$

In ACPur, however, $\int_{v\in\langle 1,2\rangle} \tilde{\delta}[v]$ cannot reach time 2, while $\int_{v\in\langle 1,2]} \tilde{\delta}[v]$ can. And in ACPur we have

$$\begin{array}{lcl} U_v(\int_{v\in\langle 1,2\rangle} \tilde{\delta}[v]) & = & U_v(\langle 1,2\rangle) & = & v \in \langle -\infty, 2\rangle & = & v < 2 \\ U_v(\int_{v\in\langle 1,2]} \tilde{\delta}[v]) & = & U_v(\langle 1,2]) & = & v \in \langle -\infty, 2] & = & v \leq 2 \end{array}$$

We use $U_b(V)$, already defined in Section 11.2. Furthermore we take $U_b^0(V) \overset{abb}{=} U_b(V) \vee b = 0$. In the axioms we use also the following abbreviations.

$$\begin{array}{lcl} \langle\!| b_0, b_1 |\!\rangle_{\geq 0} \neq \emptyset & \overset{abb}{=} & \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \wedge b_1 \geq 0 \\ b \leq \inf(\langle\!| b_0, b_1 |\!\rangle_{\geq 0}) & \overset{abb}{=} & \langle\!| b_0, b_1 |\!\rangle \neq \emptyset \wedge \\ & & (\ b_0 < 0 \Rightarrow b \leq 0 \\ & & \vee\ b_0 \geq 0 \Rightarrow b \leq b_0) \\ b \in \langle\!| b_0, b_1 |\!\rangle_{\geq 0} + b' & \overset{abb}{=} & (\ b_0 < 0 \Rightarrow b \in [b', b_1 + b' |\!\rangle \\ & & \vee\ b_0 \geq 0 \Rightarrow b \in \langle\!| b_0 + b', b_1 + b' |\!\rangle) \end{array}$$

Where $\infty + b$ and $-\infty + b$ abbreviate ∞ and $-\infty$ respectively.

Instead of these abbreviations it is also possible to introduce an operation like $_{\geq 0}$ as an abbreviation on intervals:

$$\langle\!\langle b_0, b_1 \rangle\!\rangle_{\geq 0} \overset{abb}{=} \left\{ \begin{array}{ll} & b_0 < 0 :\longrightarrow [0, b_1\rangle\!\rangle \\ \cup & b_0 \geq 0 :\longrightarrow \langle\!\langle b_0, b_1 \rangle\!\rangle \end{array} \right.$$

In this case, one has to introduce so-called *conditional intervals*, that are expressions of the form $\cup_i \alpha_i :\longrightarrow V_i$, where $\{\alpha_i\}$ is a partition and each V_i is an interval. Then, for every abbreviation concerning intervals, like $b \in V$ and $V = \emptyset$, one have has to give a corresponding abbreviation. For example, $v \in \cup_i \alpha_i :\longrightarrow V_i$ abbreviates $\vee_i \alpha_i \wedge v \in V_i$. Similarly, one can introduce $\inf(V)$ as a *conditional bound*, that is a bound of the form $\sum_i \alpha_i :\longrightarrow b_i$, where again, $\{\alpha_i\}$ is a partition and each V_i is a bound. For technical reasons we have chosen not to deal with conditional bounds and intervals.

In Table 11.4 we have also the following abbreviations, $\int_{U_v(\alpha,q)} \tilde{\delta}[v]$ expresses the idle behavior of $(\int_\alpha \tilde{a}[v] \cdot p) \mathbin{\lfloor\!\lfloor} q$, and $\int_{U_v(\alpha,\beta)}$ expresses the idle behavior of $(\int_\alpha \tilde{a}[v] \cdot p) | (\int_\beta \tilde{b}[v] \cdot q)$. We have the following abbreviations.

$$\begin{array}{lcl} \int_{U_v(\alpha,q)} \tilde{\delta}[v] & \overset{abb}{=} & \int_{U_v(\int_\alpha \tilde{\delta}[v]) \wedge U_v(q)} \tilde{\delta}[v] \\ \int_{U_v(\alpha,\beta)} \tilde{\delta}[v] & \overset{abb}{=} & \int_{U_v(\int_\alpha \tilde{\delta}[v]) \wedge U_v(\int_\beta \tilde{\delta}[v])} \tilde{\delta}[v] \end{array}$$

In axiom $CM3_{ur}$ we have $\int_{\alpha \wedge U_v^0(q)}$ and not $\int_{\alpha \wedge U_v(q)}$, as p can execute immediate actions (i.e., at time 0) in the context $p \mathbin{\lfloor\!\lfloor} \tilde{\delta}$, where $U_v(\tilde{\delta}) = U_v(\int_{ff} \tilde{\delta}[w]) = ff$.

The only difficulty of Table 11.4 is the axiomatization of the urgency operator, $\mathcal{U}_H(p)$, and its auxiliary operator $p \lhd_H q$. All the behavior of the process term $p \lhd_H q$ originates from p. The process $p \lhd_H q$ can execute an action if p can do so and q cannot execute an action $\tilde{a}$, with $a \in H$, at an earlier point in time. So, the right argument of $\lhd_H$ limits the behavior of the left argument.

The axioms UR1-UR5 can be considered as a kind of algorithm. First we rewrite $\mathcal{U}_H(p)$ to $p \lhd_H p$ by axiom UR1. Then $\lhd_H$ is distributed over all summands of its left argument, by as many as possible applications of UR2. Then, we take a summand of its right argument; if it is an $\tilde{a}$ summand, with $a \in H$, then $\int_\beta \tilde{b}[v] \cdot p$ can execute its $\tilde{b}$ action no later then the a action is enabled by $\alpha \wedge v \in V$ (see axiom UR3). If it is an a-summand, with $a \notin H$, then the summand is simply skipped (see axiom UR4). In this way we compare each summand of the right argument with every summand of the left argument. We are ready if the right component has become δ, and finally we apply $\mathcal{U}_H$ on a smaller depth.

A1		$p + q$	$=$	$q + p$
A2		$(p + q) + z$	$=$	$p + (q + z)$
A3$_{ur}$		$\int_\alpha \tilde{a}[v] \cdot p + \int_\beta \tilde{a}[v] \cdot p$	$=$	$\int_{\alpha \vee \beta} \tilde{a}[v] \cdot p$
A6		$p + \tilde{\delta}$	$=$	p
A6$_{ur}$	$v \notin fv(p)$	$p + \int_\alpha \tilde{\delta}[v]$	$=$	$p + \int_{\alpha \wedge \neg(U_v(p))} \tilde{\delta}[v]$
A7$_{ur}$		$\int_\alpha \tilde{\delta}[v] \cdot p$	$=$	$\int_\alpha \tilde{\delta}[v]$
RT0ur		$\int_\alpha \tilde{a}[v] \cdot p$	$=$	$\int_{\alpha \wedge v \geq 0} \tilde{a}[v] \cdot p$
RT1ur		$\int_{ff} \tilde{a}[v] \cdot p$	$=$	$\tilde{\delta}$
RT11$_{ur}$		$(b) \cdot (p + q)$	$=$	$(b) \cdot p + (b) \cdot q$
RT11$_{ur}$		$(b) \cdot (\alpha :\to p)$	$=$	$\alpha :\to ((b) \cdot p) + \neg\alpha :\to \tilde{\delta}[b]$
RT13$_{ur}$	$v \notin var(b)$	$(b) \cdot \int_{v \in V} \tilde{a}[v] \cdot p$	$=$	$\int_{v \in V_{\geq 0} + b} \tilde{a}[v] \cdot p[v - b/v] + \tilde{\delta}[b]$
RT14$_{ur}$		$(p + q) \gg V$	$=$	$p \gg V + q \gg V$
RT15$_{ur}$	$v \notin var(V)$	$(\int_\alpha \tilde{a}[v] \cdot p) \gg V$	$=$	$\int_{\alpha \wedge v \in V} \tilde{a}[v] \cdot p$
C1		$\alpha :\to (p + q)$	$=$	$\alpha :\to p + \alpha :\to q$
C2	$v \notin var(\alpha)$	$\alpha :\to (\int_\beta \tilde{a}[v] \cdot p)$	$=$	$\int_{\alpha \wedge \beta} \tilde{a}[v] \cdot p$
C3		$\int_\alpha \tilde{a}[v] \cdot p$	$=$	$\int_\alpha \tilde{a}[v] \cdot (\alpha :\to p)$
U1		$U_b(p + q)$	$=$	$U_b(p) \vee U_b(q)$
U2		$U_b(\alpha :\to p)$	$=$	$\alpha \wedge U_b(p)$
U3		$U_b(\int_{v \in V} \tilde{a}[v] \cdot p)$	$=$	$U_b(V)$

$a \in A_\delta,\ b \in Bound$

Table 11.3: Axioms for BPA$ur\delta$

CM1	$p \parallel q$	$=$	$p \mathbin{\lfloor\!\lfloor} q + q \mathbin{\lfloor\!\lfloor} p + p \vert q$
CM3$_{ur}$ $v \notin fv(q)$	$(\int_\alpha \tilde{a}[v] \cdot p) \mathbin{\lfloor\!\lfloor} q$		
	$=$		$\int_{\alpha \wedge U_v^0(q)} \tilde{a}[v] \cdot (p \mathbin{\lfloor\!\lfloor} (-v) \cdot q) + \int_{U_v(\alpha,q)} \tilde{\delta}[v]$
CM4	$(p_1 + p_2) \mathbin{\lfloor\!\lfloor} q$	$=$	$p_1 \mathbin{\lfloor\!\lfloor} q + p_2 \mathbin{\lfloor\!\lfloor} q$
CM7$_{ur}$	$(\int_\alpha \tilde{a}[v] \cdot p) \vert (\int_\beta \tilde{b}[v] \cdot q)$		
		$=$	$\int_{\alpha \wedge \beta} \gamma\widetilde{(a,b)}[v] \cdot (p \parallel q) + \int_{U_v(\alpha,\beta)} \tilde{\delta}[v]$
CM8	$(p_1 + p_2) \vert q$	$=$	$p_1 \vert q + p_2 \vert q$
CM9	$p \vert (q_1 + q_2)$	$=$	$p \vert q_1 + p \vert q_2$
D1$_{ur}$ $a \notin H$	$\partial_H(\int_\alpha \tilde{a}[v] \cdot p)$	$=$	$\int_\alpha \tilde{a}[v] \cdot \partial_H(p)$
D2$_{ur}$ $a \in H$	$\partial_H(\int_\alpha \tilde{a}[v] \cdot p)$	$=$	$\int_\alpha \tilde{\delta}[v]$
D3	$\partial_H(p + q)$	$=$	$\partial_H(p) + \partial_H(q)$
RN2$_{ur}$	$\rho_f(\int_\alpha \tilde{a}[v] \cdot p)$	$=$	$\int_\alpha \widetilde{f(a)}[v] \cdot \rho_f(p)$
RN3	$\rho_f(p + q)$	$=$	$\rho_f(p) + \rho_f(q)$
UR1	$\mathcal{U}_H(p)$	$=$	$p \lhd_H p$
UR2	$(p + q) \lhd_H z$	$=$	$p \lhd_H z + q \lhd_H z$
UR3 $a \in H,\ v, w \notin var(\alpha) \cup var(V)$			
	$(\int_\beta \tilde{b}[v] \cdot p) \lhd_H (\int_{\alpha \wedge w \in V} \tilde{a}[w] \cdot q + z)$		
	$= (\int_{\beta \wedge ((\alpha \wedge V_{\geq 0} \neq \emptyset) \Rightarrow v \leq \inf(V_{\geq 0}))} \tilde{b}[v] \cdot p) \lhd_H z$		
UR4 $a \notin H$	$(\int_\beta \tilde{b}[v] \cdot p) \lhd_H (\int_\alpha \tilde{a}[w] \cdot q + z)$		
		$=$	$(\int_\beta \tilde{b}[v] \cdot p) \lhd_H z$
UR5	$(\int_\beta \tilde{b}[v] \cdot p) \lhd_H \delta$	$=$	$\int_\beta \tilde{b}[v] \cdot \mathcal{U}_H(p)$

$a, b \in A_\delta$

Table 11.4: Additional axioms for ACP*ur*

11.5 Branching Bisimulation

We have already discussed that the behavior of process terms with urgent actions at one point in time is like the behavior of untimed processes. This implies that for the definition of branching bisimulation in a two phase semantics, the clause for $p \xrightarrow{a} p'$ ($a \in A_\tau$) can be taken from the definition of untimed branching bisimulation. Moreover, the clause for $p \xrightarrow{t} p'$ can be derived from Definition 6.3.3; if p is related with q, then one has to find a q' such that $q \overset{t}{\Longrightarrow} q'$ (q evolves into q' within t time units, by idling and executing internal actions) such that every state along $p \xrightarrow{t} p'$ can be related with a corresponding state along $q \overset{t}{\Longrightarrow} q'$. We denote this correspondence by $(p \xrightarrow{t} p')\mathcal{R}(q \overset{t}{\Longrightarrow} q')$, which is formalized by the following definition.

Definition 11.5.1 *$(p \xrightarrow{t} p')\mathcal{R}(q \overset{t}{\Longrightarrow} q')$ denotes that there are $t_0, \ldots, t_n$ and $q_0, q'_0, \ldots, q_n, q'_n$ such that*

$$q \equiv q_0 \xrightarrow{t_0} q'_0 \Longrightarrow q_1 \ldots q_n \xrightarrow{t_n} q'_n \equiv q'$$

such that $t_0 + \ldots + t_n = t$ and for every i in $\{0, \ldots, n\}$ we take $s_i = 0$ in case $i = 0$ and $s_i = t_0 + \ldots + t_{i-1}$ otherwise, such that $\forall s \in [s_i, s_i + t_i]$ we have $p_i^s \mathcal{R} q_i^s$ where

$$p \xrightarrow{s_i+s} p_i^s \text{ and } q_0 \xrightarrow{t_0} \ldots q_i \xrightarrow{s} q_i^s$$

We define branching bisimulation in a two phase semantics as follows.

Definition 11.5.2 *$\mathcal{R} \subseteq T^{cl}(\text{ACP}ur\tau) \times T^{cl}(\text{ACP}ur\tau)$ is a* two phase branching *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{\tilde{a}} p'$ then either $a = \tau$ and $p'\mathcal{R}q$ or $\exists z, q'$ such that $q \Longrightarrow z \xrightarrow{\tilde{a}} q'$, $p\mathcal{R}z$ and $p'\mathcal{R}q'$.*
2. *If $p \xrightarrow{t} p'$ then $\exists q'$ such that $q \overset{t}{\Longrightarrow} q'$ and $(p \xrightarrow{t} p')\mathcal{R}(q \overset{t}{\Longrightarrow} q')$.*
3. *Respectively (1) and (2) with the role of p and q interchanged.*

For $p, q \in T^{cl}(\text{ACP}ur)$ we say that p and q are (two phase) *branching bisimilar equivalent*, denoted by $p \leftrightarrow_b q$, if there is a two phase branching bisimulation that relates p and q. As usual $\leftrightarrow_b$ is not a congruence, therefore we have the following definitions.

Definition 11.5.3 (p-rooted) *A process term p' is* p-rooted *if $p' \equiv p$ or $p \xrightarrow{t} p'$ for some t.*

Definition 11.5.4 (Rootedness) *A bisimulation $\mathcal{R}$ is rooted w.r.t. p and q if $p'\mathcal{R}q'$ implies that p' is p-rooted iff q' is q-rooted*

We say that p and q are *rooted branching bisimilar*, denoted by $p \leftrightarrow_{rb} q$, if there is a branching bisimulation that is rooted w.r.t. p and q. Without proof we state that $\leftrightarrow_{rb}$ is a congruence.

11.6 A Law for Branching Bisimulation

A very easy example of rooted branching bisimilar process terms is given by the following process terms

$$\begin{array}{ll} & \tilde{a}[1] \cdot (\tilde{\tau}[2] \cdot \tilde{b}[1] + \tilde{b}[3]) \\ \leftrightarrow & \tilde{a}[1] \cdot (\tilde{\tau}[2] \cdot (-2) \cdot \tilde{b}[3] + \tilde{b}[3]) \\ \leftrightarrow_{rb} & \tilde{a}[1] \cdot \tilde{b}[3] \end{array}$$

We generalize this example to the following identity:

$$\tilde{a}[t] \cdot (\tilde{\tau}[r] \cdot \sigma_{-r}(p) + p) \leftrightarrow_{rb} \tilde{a}[t] \cdot p$$

where $r \geq 0$.

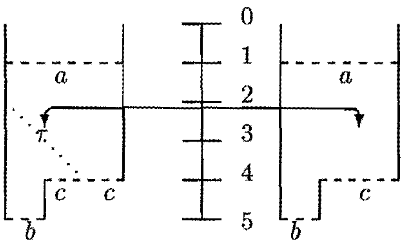
Note that we obtain the following identity as well:

$$\begin{array}{ll} & \tilde{a}[1] \cdot (\tilde{\tau}[2] \cdot \tilde{b}[1] + \tilde{b}[3] + \tilde{d}[1]) \\ \leftrightarrow & \tilde{a}[1] \cdot (\tilde{\tau}[2] \cdot (\tilde{b}[1] + \tilde{\delta}) + \tilde{b}[3] + \tilde{d}[1]) \\ \leftrightarrow & \tilde{a}[1] \cdot (\tilde{\tau}[2] \cdot ((-2) \cdot (\tilde{b}[3]) + (-2) \cdot (\tilde{d}[1])) + \tilde{b}[3] + \tilde{d}[1]) \\ \leftrightarrow & \tilde{a}[1] \cdot (\tilde{\tau}[2] \cdot (-2) \cdot (\tilde{b}[3] + \tilde{d}[1]) + \tilde{b}[3] + \tilde{d}[1]) \\ \leftrightarrow_{rb} & \tilde{a}[1] \cdot (\tilde{b}[3] + \tilde{d}[1]) \end{array}$$

Let us now consider integration by copying Example 6.5.5.

Example 6.5.5

$a(1) \cdot (\int_{v \in \langle 2,4 \rangle} \tau(v) \cdot (b(5) + c(4)) + c(4)) \leftrightarrow_{rb} a(1) \cdot (b(5) + c(4))$



We see that in cases where there are no variable dependencies in absolute time there are such dependencies in relative time. This example suggests us the following variant of the branching bisimulation law. We give a small example to recall the need of $b_0 \gg q$. Here, $b_0 \gg q$ abbreviates $q \gg \langle b_0, \infty \rangle$.

Example 11.6.1

$$\begin{array}{ll} & \tilde{a}[1] \cdot (\int_{v \in \langle 1,3 \rangle} \tilde{\tau}[v] \cdot (\tilde{b}[4 - v] + \tilde{c}[3 - v]) + \tilde{b}[4]) \\ & \quad \textit{for every } t \in \langle 1, 3 \rangle \textit{ we have } (\tilde{d}[1 - t] \leftrightarrow \tilde{\delta}) \\ \leftrightarrow & \tilde{a}[1] \cdot (\int_{v \in \langle 1,3 \rangle} \tilde{\tau}[v] \cdot (\tilde{b}[4 - v] + \tilde{c}[3 - v] + \tilde{d}[1 - v]) + \tilde{b}[4]) \\ \leftrightarrow & \tilde{a}[1] \cdot (\int_{v \in \langle 1,3 \rangle} \tilde{\tau}[v] \cdot (-v) \cdot (\tilde{b}[4] + \tilde{c}[3] + \tilde{d}[1]) + \tilde{b}[4]) \\ \not\leftrightarrow_{rb} & \tilde{a}[1] \cdot (\tilde{b}[4] + \tilde{c}[3] + \tilde{d}[1]) \end{array}$$

$$
\begin{array}{ll}
\mathrm{B}_{ur} & w \notin fv(p+q) \\
 & \alpha \;=\; (\; \llparenthesis b_0, b_1 \rrparenthesis \neq \emptyset \;\wedge \\
 & \qquad ((U_{b_1}(p) \wedge \neg(U_{b_1}(q))) \;\vee\; (\neg(U_{b_1}(p)) \wedge U_{b_1}(q))) \;) \\
 & \\
 & \int_{\alpha\wedge\beta} \tilde{a}[v] \cdot (\int_{w \in \llparenthesis b_0, b_1 \rrparenthesis} \tilde{\tau}[w] \cdot (-w) \cdot (p+q) + p) \\
 & = \\
 & \int_{v \in V} \tilde{a}[v] \cdot (p + b_0 \gg q)
\end{array}
$$

Table 11.5: The branching law for relative time with urgent actions

The condition

$$(U_{b_1}(p) \wedge \neg(U_{b_1}(q))) \;\vee\; (\neg(U_{b_1}(p)) \wedge U_{b_1}(q))$$

expresses that at b_1 one of the two components of $p+q$ is still "active", while the other component has already been dropped from the computation. For furter details we refer to Section 6.5.

We will not discuss the soundness and completeness of ACPur + B_{ur} w.r.t. $\underline{\leftrightarrow}_{rb}$, but we think that the techniques of Chapter 7 suffice to obtain it.

11.7 Branching Bisimulation with and without Urgent Actions

In BPA$\rho\delta\tau$, BPA with absolute time and without urgent actions, we identify the following two processes by branching bisimulation:

$$b(2) + \tau(2) \cdot c(3) \;\underline{\leftrightarrow}_b\; b(2) + c(3)$$

In relative time this identity looks like

$$b[2] + \tau[2] \cdot c[1] \;\underline{\leftrightarrow}_b\; b[2] + c[3]$$

The point is that at time 2 a choice is made; either the b is executed or it is decided to idle further in which case the c will be executed one time unit later.

However, in the two phase branching bisimulation semantics we have:

$$\tilde{b}[2] + \tilde{\tau}[2] \cdot \tilde{c}[1] \;\;\not\underline{\leftrightarrow}_b\; \tilde{b}[2] + \tilde{c}[3].$$

After idling two time units the processes have evolved into

$$\tilde{b}[0] + \tilde{\tau}[0] \cdot \tilde{c}[1] \text{ and } \tilde{b}[0] + \tilde{c}[1]$$

respectively. We cannot match the transition

$$\tilde{b}[0] + \tilde{\tau}[0] \cdot \tilde{c}[1] \xrightarrow{\tilde{\tau}} \tilde{c}[1]$$

of the left hand side somehow with a transition on the right hand side, since $\tilde{b}[0]+\tilde{c}[1]$ does not have a $\tilde{\tau}$ transition, and we cannot relate $\tilde{c}[1]$ with $\tilde{b}[0] + \tilde{c}[1]$ either.

Hence, to reobtain this identity we propose to add a $\tilde{\tau}$ transition

$$\tilde{b}[0] + \tilde{c}[1] \xrightarrow{\tilde{\tau}} \tilde{c}[1]$$

as well, which expresses the decision *not* to execute the $\tilde{b}$ but to continue with idling. Then, we can conclude that the two processes are indeed branching bisimilar.

In order to obtain this $\tilde{\tau}$ transition we add a new operator $\iota(p)$. $\iota(p)$ is like p, it only blocks the immediate actions of p. Its action rule and its axioms are given in Table 11.6.

$\dfrac{p \xrightarrow{t} p'}{\iota(p) \xrightarrow{t} p'}$	$\iota(p+q) = \iota(p) + \iota(q)$ $\iota(\int_\alpha \tilde{a}[v] \cdot p) = \int_{\alpha \wedge v>0} \tilde{a}[v] \cdot p$

Table 11.6: The action rule and axioms for $\iota(p)$

By adding the action rule of Table 11.7 we obtain for each moment of choice a $\tilde{\tau}$ transition which expresses the decision to idle further.

$$\frac{p \xrightarrow{\tilde{a}} p' \quad p \xrightarrow{t} p''}{p \xrightarrow{\tilde{\tau}} \iota(p)}$$

Table 11.7: Action rule that adds a $\tilde{\tau}$ transition for each moment of choice

We denote the (rooted) branching bisimulation equivalence which we obtain by adding the action rule of Table 11.7 by $\underline{\leftrightarrow}^*_{(r)b}$.

B^*_{urp} $\quad U_t(q) = tt$	
$\tilde{a}[r] \cdot (\tilde{\tau}[t] \cdot p + q) = \tilde{a}[r] \cdot (\tilde{\tau}[t] \cdot p + \tilde{\tau}[t] \cdot \iota((-t) \cdot q) + q \gg [0,t])$	

We will not discuss the equivalence $\underline{\leftrightarrow}^*_{(r)b}$ any more in this thesis, though we think that it is an interesting question whether or not process terms like $\tilde{a}[1] \cdot (\tilde{b}[2] + \tilde{\tau}[2] \cdot \tilde{c}[1])$ and $\tilde{a}[1] \cdot (\tilde{b}[2] + \tilde{c}[3])$ should be identified.

It appears that the choice for a two phase semantics interacts in a subtle way with the notion of branching bisimulation.

11.8 Delay and Weak Bisimulation

As usual we obtain the definition of a delay bisimulation from the definition of a branching bisimulation by omitting the requirement that the intermediate states are related as well. So, in the first clause we omit the requirement that $p\mathcal{R}z$ and in the second clause we relax the requirement $(p \xrightarrow{t} p')\mathcal{R}(q \overset{t}{\Longrightarrow} q')$ to simply $p'\mathcal{R}q'$.

Definition 11.8.1 *$\mathcal{R} \subseteq T^{cl}(\mathrm{ACP}ur\tau) \times T^{cl}(\mathrm{ACP}ur\tau)$ is a* two phase delay *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{\tilde{a}} p'$ then either $a = \tau$ and $p'\mathcal{R}q$ or $\exists z, q'$ such that $q \Longrightarrow z \xrightarrow{\tilde{a}} q'$ and $p'\mathcal{R}q'$.*
2. *If $p \xrightarrow{t} p'$ then $\exists q'$ such that $q \overset{t}{\Longrightarrow} q'$ and $p'\mathcal{R}q'$.*
3. *Respectively (1) and (2) with the role of p and q interchanged.*

Moreover, we obtain the definition of a weak bisimulation by allowing a sequence of $\tilde{\tau}$'s "afterwards" in the first clause:

Definition 11.8.2 *$\mathcal{R} \subseteq T^{cl}(\mathrm{ACP}ur\tau) \times T^{cl}(\mathrm{ACP}ur\tau)$ is a* two phase weak *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{\tilde{a}} p'$ then either $a = \tau$ and $p'\mathcal{R}q$ or $\exists z, z', q'$ such that $q \Longrightarrow z \xrightarrow{\tilde{a}} z' \Longrightarrow q'$ and $p'\mathcal{R}q'$.*
2. *If $p \xrightarrow{t} p'$ then $\exists q'$ such that $q \overset{t}{\Longrightarrow} q'$ and $p'\mathcal{R}q'$.*
3. *Respectively (1) and (2) with the role of p and q interchanged.*

We define (rooted) delay, denoted by $\leftrightarrow_{(r)d}$, and (rooted) weak bisimulation equivalence, denoted by $\leftrightarrow_{(r)w}$, along the standard way.

Without proof we state that $\leftrightarrow_{rd}$ and *rwbis* are congruences for ACPur.

11.9 Laws for Rooted Delay and Weak Bisimulation

We can construct the laws for rooted delay branching bisimulation with relative time and urgent actions by adapting the laws T1$_I$ and T2$_I$ from Section 8.4, and we obtain the laws T1$_{ur}$ and T2$^a_{ur}$.

In BPA$\rho\delta\tau$ with delay bisimulation we have the following identity:

$$\int_{v\in[1,2)} \tau(v) \cdot \int_{v\in[1,2]} a(w) \leftrightarrow^{t}_{rd} \int_{v\in[1,2)} \tau(v) \cdot \int_{v\in[1,2]} a(w) + \int_{v\in(1,2]} a(w)$$

In the context of urgent actions (and relative time) we have

$$\int_{v\in[1,2)} \tilde{\tau}[v] \cdot \int_{v\in[0,2-v]} \tilde{a}[w] \not\leftrightarrows_{rd} \int_{v\in[1,2)} \tilde{\tau}[v] \cdot \int_{v\in[1,2-v]} \tilde{a}[w] + \int_{v\in\langle 1,2]} \tilde{a}[w]$$

as the process term on the right hand side can idle untill 2, after which it can execute an $\tilde{a}$. Due to the rootedness requirement we can not match this idling properly on the left hand side. We have the following identity instead:

$$\int_{v\in[1,2)} \tilde{\tau}[v] \cdot \int_{v\in[0,2-v]} \tilde{a}[w] \leftrightarrows_{rd} \int_{v\in[1,2)} \tilde{\tau}[v] \cdot \int_{v\in[1,2-v]} \tilde{a}[w] + \int_{v\in[1,2)} \tilde{a}[w]$$

So, in case of $\int_{v\in V} \tilde{\tau}[v] \cdot p$, we may put that part of p "outside" of the $\int_{v\in V} \tilde{\tau}[v]$ that is restricted to V; we denote this part by $p \gg V$.

The law T3$_{ur}$ is rather simple, only "immediate" $\tilde{\tau}$'s are allowed after the action $\tilde{a}$ in the first clause of weak bisimulation, and therefore we can adapt the law T3$_I$ of Section 8.6 to our setting only in case of $\tilde{\tau}[0]$.

There is also a typical pair of rooted delay bisimilar process terms for which there is no equivalent pair in the case without urgent actions:

$$\begin{aligned} & \textstyle\int_{w\in[1,3]} \tilde{\tau}[w] \cdot \tilde{b}[0] + p \\ \leftrightarrows_{rd} \quad & \textstyle\int_{w\in[1,3]} \tilde{\tau}[w] \cdot \tilde{b}[0] + \int_{w\in[1,3]} \tilde{b}[v] + p \end{aligned}$$

This identity follows from the fact that for any $t \in [1,3]$ we have

$$\begin{aligned} & \textstyle\int_{w\in[1,3]} \tilde{\tau}[w] \cdot \tilde{b}[0] + p \xrightarrow{t} \\ & \qquad \textstyle\int_{w\in[0,3-t]} \tilde{\tau}[w] \cdot \tilde{b}[0]\ [+p'] \\ & \textstyle\int_{w\in[1,3]} \tilde{\tau}[w] \cdot \tilde{b}[0] + \int_{w\in[1,3]} \tilde{b}[v] + p \xrightarrow{t} \\ & \qquad \textstyle\int_{w\in[0,3-t]} \tilde{\tau}[w] \cdot \tilde{b}[0] + \int_{w\in[0,3-t]} \tilde{b}[v]\ [+p'] \end{aligned}$$

where we have the p' summand only in case $p \xrightarrow{t} p'$. And we can match the transition

$$\int_{w\in[0,3-t]} \tilde{\tau}[w] \cdot \tilde{b}[0] + \int_{w\in[0,3-t]} \tilde{b}[v]\ [+p'] \xrightarrow{\tilde{b}} \int_{tt} \tilde{\delta}[v]$$

with the sequence

$$\int_{w\in[0,3-t]} \tilde{\tau}[w] \cdot \tilde{b}[0]\ [+p'] \xrightarrow{\tilde{\tau}} \tilde{b}[0] \xrightarrow{\tilde{b}} \int_{tt} \tilde{\delta}[v]$$

on the left hand side. (Note that $\int_\alpha \tilde{a}[v]$ abbreviates $\int_\alpha \tilde{a}[v] \cdot \int_{tt} \tilde{\delta}[v]$.)

We generalize this identity to the law T2$^b_{ur}$. The laws for delay and weak bisimulation for the two phase semantics are given in Table 11.8.

11.10 Alternative Definitions for Weak Bis.

We define $p \overset{\tilde{a}}{\Longrightarrow} p'$ by $\exists z, z'\ p \Longrightarrow z \xrightarrow{\tilde{a}} z' \Longrightarrow p'$ for $\tilde{a} \neq \tilde{\tau}$. Moreover, we define $p \overset{\tilde{\tau}}{\Longrightarrow} p'$ by $p \Longrightarrow p'$. So, $p \overset{\tilde{a}}{\Longrightarrow} p'$ may be an empty sequence in case $\tilde{a} = \tilde{\tau}$.

Using this definition we can give a redefinition of Definition 11.8.2:

$\mathrm{T1}_{ur}$ $\quad w \notin fv(p) \quad \alpha = (\ \llparenthesis b_0, b_1 \rrparenthesis \neq \emptyset \wedge U_{b_1}(p)\ )$

$$\textstyle\int_{\alpha \wedge \beta} \tilde{a}[v] \cdot \int_{w \in \llparenthesis b_0, b_1 \rrparenthesis} \tilde{\tau}[w] \cdot (-w) \cdot p \;=\; \int_{\alpha \wedge \beta} \tilde{a}[v] \cdot (b_0 \gg p)$$

$\mathrm{T2}^{a}_{ur}$ $\quad w \notin fv(p)$

$$\textstyle\int_{w \in \llparenthesis b_0, b_1 \rrparenthesis} \tilde{\tau}[w] \cdot (-w) \cdot p \;=\; \int_{w \in \llparenthesis b_0, b_1 \rrparenthesis} \tilde{\tau}[w] \cdot (-w) \cdot p + p \gg V$$

$\mathrm{T2}^{b}_{ur}$

$$\textstyle\int_{\alpha} \tilde{\tau}[w] \cdot (\int_{v \in [0, b' \rrparenthesis} \tilde{a}[v] \cdot p + z) \;=\; \int_{\alpha} \tilde{\tau}[w] \cdot (\int_{v \in [0, b' \rrparenthesis} \tilde{a}[v] \cdot p + z) + \int_{\alpha} \tilde{a}[w] \cdot p[0/v]$$

$\mathrm{T3}_{ur}$

$$\textstyle\int_{\alpha} \tilde{a}[v] \cdot (\tilde{\tau}[0] \cdot p + q) \;=\; \int_{\alpha} \tilde{a}[v] \cdot (\tilde{\tau}[0] \cdot p + q) + \int_{\alpha} \tilde{a}[v] \cdot p$$

Table 11.8: τ-laws for rooted delay and weak bisimulation

Definition 11.10.1 *$\mathcal{R} \subseteq T^{cl}(\text{ACP}ur\tau) \times T^{cl}(\text{ACP}ur\tau)$ is a* two phase weak *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \xrightarrow{\mu} p'$ then $\exists q'$ such that $q \overset{\mu}{\Longrightarrow} q'$ and $p'\mathcal{R}q'$.*
2. *If $q \xrightarrow{\mu} q'$ then $\exists p'$ such that $p \overset{\mu}{\Longrightarrow} p'$ and $p'\mathcal{R}q'$.*

We can generalize the definition of a weak bisimulation as follows:

Definition 11.10.2 *$\mathcal{R} \subset T(\text{ACP}ur\tau) \times T(\text{ACP}ur\tau)$ is a* generalized two phase weak *bisimulation if whenever $p\mathcal{R}q$ then*

1. *If $p \overset{\mu}{\Longrightarrow} p'$ then $\exists q'$ such that $q \overset{\mu}{\Longrightarrow} q'$ and $p'\mathcal{R}q'$.*
2. *If $q \overset{\mu}{\Longrightarrow} q'$ then $\exists p'$ such that $p \overset{\mu}{\Longrightarrow} p'$ and $p'\mathcal{R}q'$.*

We can show that Definition 11.8.2 and 11.10.2 are in fact equivalent.

Proposition 11.10.3 *$\mathcal{R}$ is a two phase weak bisimulation* iff *$\mathcal{R}$ is a generalized two phase weak bisimulation*

Proof. Omitted. $\square$

11.11 The Embedding of ACPur into ACPρI

Baeten & Bergstra express relative time process terms in absolut time real time ACP by means of the so-called *tick* operator, $\surd v.p$. The expression $\surd v.p$ is like a function; when we start it at time t then t is substituted for v in p. Baeten & Bergstra have a reduction rule that is similar to the β-reduction rule in the λ-calculus:

$$t \gg (\surd v.p) \;=\; t \gg p[t/v]$$

In [BB93b] Baeten & Bergstra have introduced urgent action in real time ACP by taking non standard reals in the time domain, as is shortly explained below.

We denote the set of non standard reals by $\mathbb{R}^*$. Intuitively, for every real number t there is an "environment" $S_t \subset \mathbb{R}^*$ such that each $r \in S_t$ is infinitary close to t. By I we denote the set of positive non standard reals that are infinitary close to 0. As time domain we take

$$\mathbb{R} \cup \{t + \epsilon | t \in \mathbb{R},\ \epsilon \in I\}$$

The process $\tilde{a}[t] \cdot p$ can now be expressed by the following expression:

$$\surd v.(\int_{w \in \{r | \exists \epsilon\ r = t+v+\epsilon\}} a(w) \cdot (\!|p|\!))$$

where $(\!|p|\!)$ is the absolute time process expression for p. In this setting the process $\tilde{a}[0] \cdot \tilde{b}[0] \cdot p$ executes first the action a and then the action b infinitary close to 0.

Note, that we do not have the above process in prefixed integration, as we cannot define the set $\{r | \exists \epsilon\ r = t + v + \epsilon\}$ by a boolean expression.

11.12 The Embedding of ACP into ACPur

Assume that we have generalized ACPur with general multiplication, then there are several embeddings possible for (untimed) ACP.

As we have discussed in Section 6.6 we can embed ACP into ACPur by translating the atomic action a into $\int_{tt} \tilde{a}[v]$ for strong bisimulation and $\int_{tt} \tilde{a}[v] \cdot \int_{tt} \tilde{\tau}[w]$ in the case of branching, delay or weak bisimulation.

However, there is also another embedding possible, namely by simply translating a into $\tilde{a}[0]$, or even $\tilde{a}[t]$ for a fixed t. This embedding relies on the fact that in a two phase semantics the behavior in one point in time corresponds with untimed behavior.

12

Related Work

12.1 Introduction

In this chapter we discuss some related work. The major part of this chapter is dedicated to the translation of several time calculi into the calculus ACPur, that we have introduced in the previous chapter.

In the last section of this chapter we refer very briefly to other interested areas of research that deal with time, though we do not claim to be complete.

In this chapter we translate the following real time calculi into ACPur:

- TCCS of Moller & Tofts ([MT90],[MT92]),
- Wang's Timed CCS ([Wan90],[Wan91b] and [Wan91a]),
- Chen's Timed CCS ([Che91],[Che92],[Che93]),
- ATP of Sifakis & Nicollin ([NS90],[NSY91]),
- TPL of Hennessy & Regan ([HR90]),
- TIC of Quemada et. al. ([QdFA93]).

We also discuss the axiomatizations of weak bisimulations, that can be found in [Wan91a], [MT92], [Che93] and [QdFA93].

For every timed calculus $\mathcal{C}$ we define a translation function

$$(\!|\ |\!) \ : \ \mathcal{C} \longrightarrow \text{ACP}ur$$

We apply the convention that process terms from a certain calculus $\mathcal{C}$ are denoted by P, Q and Z. Process terms from ACPur, however, remain denoted by p, q and z. In certain calculi we encounter constructs which we cannot express in ACPur. In these cases we introduce a similar construct in the context of ACPur; we provide it with two phase action rules and we give the characterizing axioms.

In this chapter we use the following abbreviations.

$$\begin{array}{lllllll}
\underset{.}{\tilde{\delta}} & \overset{abb}{=} & \tilde{\delta}[0] & \underset{.}{\tilde{a}} \cdot p & \overset{abb}{=} & \int_{v=0} \tilde{a}[v] \cdot p & (v \notin fv(p)) \\
\underline{\tilde{\delta}} & \overset{abb}{=} & \int_{tt} \tilde{\delta}[v] & \underline{\tilde{a}} \cdot p & \overset{abb}{=} & \int_{tt} \tilde{a}[v] \cdot p & (v \notin fv(p))
\end{array}$$

Note that in ACPur we have $\underline{\tilde{\delta}} = \tilde{\delta}$, in this chapter we prefer to use $\underset{.}{\tilde{\delta}}$ instead of $\tilde{\delta}$ as it is more conform the use of $\underset{.}{\tilde{a}}$ for $a \in A$.

For $p \in T^{cl}(\mathrm{ACP}ur)$ the condition $U_v(p)$, for arbitrary v, reduces to a condition of the form $v \in \langle -\infty, t \rangle$, and we denote by $U(p)$ the interval $\langle -\infty, t \rangle$.

We will encounter two kinds of prefixes, *immediate* and *delayable* prefixes. A prefix $a.p$ is immediate if the action a has to be executed immediate. A delayable prefix allows the action a to be executed at an arbitrary point in time. An immediate prefix corresponds to our $\underset{.}{\tilde{a}} \cdot p$ and a delayable prefix corresponds to our $\underline{\tilde{a}}.p$. Some of the calculi also have a time prefix that corresponds to our $(t) \cdot p$.

In most of the other calculi we encounter also an operator for encapsulation (∂_H in ACPur) and renaming (ρ_f in ACPur). In the next section we discuss how the parallel merge appears in the timed calculi.

12.2 The Parallel Merge

The parallel merge of most of the calculi correspond to our parallel merge; both components have to proceed in time equally. The left merge ($\mathbin{⌊\!⌊}$) and communication merge ($|$) are not present in several other calculi. In fact, we can remove them from ACPur as well. The price to pay is a more complex axiomatization of the parallel merge, for which we need the following *expansion* law, in case $\mathbin{⌊\!⌊}, |$ are not available:

Proposition 12.2.1 (Expansion Law for ACPur**)**
Let

$$\begin{array}{lll}
p & \simeq & \sum_i \int_{\alpha_i} \tilde{a}_i[v] \cdot p_i \\
q & \simeq & \sum_j \int_{\beta_j} \tilde{b}_j[v] \cdot q_j
\end{array}$$

then

$$\begin{array}{lllll}
p \| q & = & \sum_i & \int_{\alpha_i \wedge U_v^0(q)} & \tilde{a}_i[v] \cdot (p_i \| (-v) \cdot q) \\
 & + & \sum_j & \int_{\beta_j \wedge U_v^0(p)} & \tilde{b}_j[v] \cdot ((-v) \cdot p \| q_j) \\
 & + & \sum_{i,j:\gamma(a_i,b_j) \neq \delta} & \int_{\alpha_i \wedge \beta_j} & \widetilde{\gamma(a_i, b_j)}[v] \cdot (p_i \| q_j) \\
 & + & & \int_{U_v(p) \wedge U_v(q)} & \tilde{\delta}[v]
\end{array}$$

The expression $U_v^0(q)$ abbreviates the condition $U_v(q) \vee v = 0$, that is, in case of $\int_{\alpha_i \wedge U_v^0(q)} \tilde{a}_i[v] \cdot [\ldots]$ then we can execute an action $\tilde{a}$ at points in time to which q can idle, or immediate at time 0. Similar laws can be found in the work of Wang and Chen.

In the previous chapter we have introduced delay and weak bisimulation in ACPur. In the Chapters 1 and 8 we have discussed briefly that the combination of delay and weak bisimulation, and the communication and left merge is a little problematic. Again, we do not work this out in more detail. A possible way out, however, is to disgard these auxiliary operators and their axioms and to add the above expansion law to the axiom system of ACPur.

The expansion law above shows us the need of the time variables; if we consider $p\|q$ where p executes an action at some time t, then the way q proceeds may depend on t. For example, take a simple process like

$$(\tilde{\underline{a}}\cdot\tilde{\underline{\delta}}) \parallel ((1)\cdot\tilde{\underline{b}}\cdot\tilde{\underline{\delta}}),$$

and note that there are no variable dependencies at all in this process. If we remove the $\|$ then we obtain the following term

$$\int_{v\leq 1}\tilde{a}[v]\cdot(1-v)\cdot\tilde{\underline{b}}\cdot\tilde{\underline{\delta}} \;+\; (1)\cdot\tilde{\underline{b}}\cdot\tilde{\underline{a}}\cdot\tilde{\underline{\delta}} \;+\; (1)\cdot\tilde{\underline{a}}\cdot\tilde{\underline{b}}\cdot\tilde{\underline{\delta}},$$

in which a variable dependency has been introduced.

This idea has been formalized by Larsen & Godskesen [GL92]. They proved that there exists no expansion theorem for some dense timed calculi without time variables. This was proved by translating such a calculus into timed graphs [AD90]. The number of parallel components in a process expression corresponds with the number of clocks in the corresponding timed graph. An expansion theorem would imply that a timed graph with $n+1$ clocks is equally expressive as a timed graph with n clocks. They showed, however, that a timed graph with $n+1$ clocks is strictly more expressive than a timed graph with n clocks.

12.3 TCCS of Moller & Tofts

Moller and Tofts have presented in [MT90] a first version of their TCCS. In that paper they considered an arbitrary time domain and strong bisimulation equivalence. The expansion theorem and the completeness were proven only for a discrete time domain. In [MT90] TCCS has an immediate prefix $a.P$ and a time prefix $(t).P$. Moreover, it has two alternative compositions, the *strong choice*, denoted by $+$, and the *weak choice*, denoted by $\oplus$.

The strong choice of P and Q can idle only whenever both P and Q can idle, so no summands can be dropped from the computation by idling. The weak choice of P and Q corresponds to our real time ACP $+$; if P can idle to a point in time to which Q can not idle, then the weak choice of P and Q can idle to this point as well and all summands of Q are dropped from the computation.

In [MT90] Moller and Tofts do not have a delayable prefix. To allow the process $a.nil$ to idle they introduce the *delay operator* $\delta.P$. $\delta.P$ delays all immediate prefixes of P. Moreover, it drops all summands of P that are forced to idle first, for example a summand like $(1).P'$. The delay operator is not expressible in ACPur with prefixed integration, though in the context of general integration it is: construct as follows:

$$(\!|\delta.P|\!) \;=\; \int_{v\in[0,\infty)}(v)\cdot((\!|P|\!)\gg[0,0])$$

$\Delta(p) \xrightarrow{t} \Delta(p)$	$\dfrac{p \xrightarrow{\tilde{a}} p'}{\Delta(p) \xrightarrow{\tilde{a}} p'}$

$\Delta 1$	$\Delta(p+q)$	$=$	$\Delta(p)+\Delta(q)$
$\Delta 2$	$\Delta(\int_\alpha \tilde{a}[v]\cdot p)$	$=$	$(\alpha[0/v] :\rightarrow \underline{\tilde{a}}[v]\cdot p[0/v]) + (\neg\alpha[0/v] :\rightarrow \underline{\tilde{\delta}})$

Table 12.1: Action rules and axioms for $\Delta(p)$

We introduce the delay operator in ACPur (with prefixed integration) for the sake of the translation. We denote this new operator by $\Delta(p)$. Its action rules and axioms are given in Table 12.1.

In [MT92] Moller & Tofts give other notations for the strong and weak choice, instead of $+$ and $\oplus$ they use $+\!\!+$ and $+$ respectively. Their motivation is that the weak choice occurs in the normal forms of the Expansion Theorem of [MT90] and that other calculi have only one choice construct, which for the most cases correspond with their weak choice.

Moreover, in [MT92] they replace the delay operator $\delta.P$ by a delayable prefix $\underline{a}.P$ and the delay nil $\underline{0}$.

The translation of TCCS into ACPur is given in Table 12.2.

	$\llparenthesis 0 \rrparenthesis$	$=$	$\tilde{\delta}$
Only in [MT92]	$\llparenthesis \underline{0} \rrparenthesis$	$=$	$\underline{\tilde{\delta}}$
	$\llparenthesis a.P \rrparenthesis$	$=$	$\tilde{a}\cdot\llparenthesis P \rrparenthesis$
Only in [MT92]	$\llparenthesis \underline{a}.P \rrparenthesis$	$=$	$\underline{\tilde{a}}\cdot\llparenthesis P \rrparenthesis$
	$\llparenthesis (t).P \rrparenthesis$	$=$	$(t)\cdot\llparenthesis P \rrparenthesis$
Only in [MT90]	$\llparenthesis \delta.P \rrparenthesis$	$=$	$\Delta(\llparenthesis P \rrparenthesis)$
In [MT90] as $+$	$\llparenthesis P +\!\!+ Q \rrparenthesis$	$=$	$(\llparenthesis P \rrparenthesis + \llparenthesis Q \rrparenthesis) \gg U(\llparenthesis P \rrparenthesis) \cap U(\llparenthesis Q \rrparenthesis)$
In [MT90] as $\oplus$	$\llparenthesis P + Q \rrparenthesis$	$=$	$\llparenthesis P \rrparenthesis + \llparenthesis Q \rrparenthesis$
	$\llparenthesis P \mid Q \rrparenthesis$	$=$	$\llparenthesis P \rrparenthesis \| \llparenthesis Q \rrparenthesis$
	$\llparenthesis P\backslash a \rrparenthesis$	$=$	$\partial_{\{a,\bar{a}\}}(\llparenthesis P \rrparenthesis)$
	$\llparenthesis P[f] \rrparenthesis$	$=$	$\rho_f(\llparenthesis P \rrparenthesis)$

Table 12.2: Translation of TCCS (Moller and Tofts) into ACPur, where $\gamma(a,\bar{a})=\tau$

12.4 Moller & Tofts's weak bisimulation

Moller & Tofts [MT92] have defined weak bisimulation equivalence as in Definition 11.8.2. They call it $\mathcal{T}$-bisimulation equivalence, they define (*observational*)$\mathcal{T}$-congruence by applying a rootedness condition which coincides with ours. They show that the obtained congruence is indeed the largest possible congruence. They have four τ-laws, these are given in Table 12.3.

$\tau1$	$\underline{\tilde{a}} \cdot p$	$=$	$\underline{\tilde{a}} \cdot \tilde{\tau} \cdot p$
$\tau2$	$\underline{\tilde{\tau}} \cdot p$	$=$	$\underline{\tilde{\tau}} \cdot p + p$
$\tau3$	$\underline{\tilde{\tau}} \cdot (p + (\underline{\tilde{a}} \cdot q \mathbin{++} z))$	$=$	$\underline{\tilde{\tau}} \cdot (p + (\underline{\tilde{a}} \cdot q \mathbin{++} z)) + \underline{\tilde{a}} \cdot q$
$\tau4$	$\underline{\tilde{a}} \cdot (p + (\underline{\tilde{\tau}} \cdot q \mathbin{++} z))$	$=$	$\underline{\tilde{a}} \cdot (p + (\underline{\tilde{\tau}} \cdot q \mathbin{++} z)) + \underline{\tilde{a}} \cdot q$

Table 12.3: Tau laws of Moller & Tofts

The law $\tau1$ follows direct from our $\mathrm{T1}_{ur}$. We have

$$\tau2 \vdash \underline{\tau} \cdot \tilde{b}[1] = \underline{\tau} \cdot \tilde{b}[1] + \tilde{b}[1],$$

and thus as well

$$\tau2 \vdash \underline{\tau} \cdot \tilde{b}[1] + \tilde{c}[1] = \underline{\tau} \cdot \tilde{b}[1] + \tilde{b}[1] + \tilde{c}[1]$$

This example shows us that the law $\tau2$ is not sound for $\underline{\leftrightarrow}_d$, and thus, that it is not sound for $\underline{\leftrightarrow}_{rd}$ either.

Proposition 12.4.1

$$p \equiv \underline{\tau} \cdot \tilde{b}[1] + \tilde{c}[1] \;\; \not\underline{\leftrightarrow}_d \; \underline{\tau} \cdot \tilde{b}[1] + \tilde{b}[1] + \tilde{c}[1] \equiv q$$

Proof. We show that the transition

$$q \xrightarrow{1} \underline{\tau} \cdot \tilde{b}[1] + \tilde{b}[0] + \tilde{c}[0] \equiv q'$$

cannot be matched by p. Note that q' can execute both a $\tilde{b}$ and a $\tilde{c}$.

There are two candidates for a $\overset{1}{\Longrightarrow}$ sequence, which will both fail:

1. $p \overset{1}{\Longrightarrow} \tilde{b}[0]$, no $\tilde{c}$ action is enabled.
2. $p \overset{1}{\Longrightarrow} \underline{\tilde{\tau}} \cdot \tilde{b}[1] + \tilde{c}[0]$, no $\tilde{b}$ action is enabled.

□

The point is that for sound instan ces of $\tau2$ only the "immediate" summands of p may be put outside of the $\underline{\tau}$, as is shown also by our law $\mathrm{T2}^b_{ur}$. Note, that the two processes of proposition 12.4.1 can be expressed as well in the Timed CCS calculus of Moller and Tofts, since $\tilde{a}[t] \cdot p$ can be written in their calculus as $(t).(ap \mathbin{++} 0)$. Hence, it is not clear to us how the law $\tau2$ can be sound in TCCS.

To derive the laws $\tau3$ and $\tau4$ we reformulate them as given in Table 12.4.

$$\begin{array}{lrcl}
\tau 3' & \underline{\tilde{\tau}} \cdot (\underset{.}{\tilde{a}} \cdot p + q) & = & \underline{\tilde{\tau}} \cdot (\underset{.}{\tilde{a}} \cdot p + q) + \underline{\tilde{a}} \cdot p \\
\tau 4' & \underline{\tilde{a}} \cdot (\underset{.}{\tilde{\tau}} \cdot p + q) & = & \underline{\tilde{a}} \cdot (\underset{.}{\tilde{\tau}} \cdot p + q) + \underline{\tilde{a}} \cdot p
\end{array}$$

Table 12.4: Alternative formulations for the laws $\tau 3$ and $\tau 4$ of Moller & Tofts

Proposition 12.4.2 *We can derive the law $\tau 3$ from the laws for strong bisimulation of [MT92] and $\tau 3'$.*

Proof. We list some of the axioms of Moller and Tofts which are sound for strong bisimulation equivalence. We give also one identity ID1 which will be used in the final derivation. ap must be read as $\tilde{a}[0] \cdot p$. Note that these laws can be derived as well in ACPur.

$$\begin{array}{lrcl}
\text{AX1} & p \mathbin{+\!\!+} (q + z) & = & (p \mathbin{+\!\!+} q) + (p \mathbin{+\!\!+} z) \\
\text{AX2} & \underset{.}{\tilde{a}} \cdot p + (q \mathbin{+\!\!+} \underset{.}{\tilde{\delta}}) & = & \underset{.}{\tilde{a}} \cdot p \mathbin{+\!\!+} q \\
\text{AX3} & p + \underset{.}{\tilde{\delta}} & = & p \\
\text{AX4} & \underline{\tilde{a}} \cdot p & = & \underline{\tilde{a}} \cdot p + \underset{.}{\tilde{a}} \cdot p \\
\text{ID1} & (p + \underset{.}{\tilde{a}} \cdot q) \mathbin{+\!\!+} z & = & (p \mathbin{+\!\!+} z) + \underset{.}{\tilde{a}} \cdot q
\end{array}$$

First we derive the identity ID1:

$$\begin{array}{cl}
 & (p + \underset{.}{\tilde{a}} \cdot q) \mathbin{+\!\!+} z) \\
\overset{\text{AX1}}{=} & (p \mathbin{+\!\!+} z) + (\underset{.}{\tilde{a}} \cdot q \mathbin{+\!\!+} z) \\
\overset{\text{AX2}}{=} & (p \mathbin{+\!\!+} z) + (z \mathbin{+\!\!+} \underset{.}{\tilde{\delta}}) + \underset{.}{\tilde{a}} \cdot q \\
\overset{\text{AX1}}{=} & ((p + \underset{.}{\tilde{\delta}}) \mathbin{+\!\!+} z) + \underset{.}{\tilde{a}} \cdot q \\
\overset{\text{AX3}}{=} & (p \mathbin{+\!\!+} z) + \underset{.}{\tilde{a}} \cdot q
\end{array}$$

And then we can derive $\tau 3$:

$$\begin{array}{cl}
 & \underline{\tilde{\tau}} \cdot (p + (\underline{\tilde{a}} \cdot q \mathbin{+\!\!+} z)) \\
\overset{\text{AX4}}{=} & \underline{\tilde{\tau}} \cdot (p + ((\underline{\tilde{a}} \cdot q + \underset{.}{\tilde{a}} \cdot q) \mathbin{+\!\!+} z)) \\
\overset{\text{ID1}}{=} & \underline{\tilde{\tau}} \cdot (p + \underset{.}{\tilde{a}} \cdot q + (\underline{\tilde{a}} \cdot q \mathbin{+\!\!+} z)) \\
\overset{\tau 3'}{=} & \underline{\tilde{\tau}} \cdot (p + \underset{.}{\tilde{a}} \cdot q + (\underline{\tilde{a}} \cdot q \mathbin{+\!\!+} z)) + \underline{\tilde{a}} \cdot q \\
= & \underline{\tilde{\tau}} \cdot (p + (\underline{\tilde{a}} \cdot q \mathbin{+\!\!+} z)) + \underline{\tilde{a}} \cdot q
\end{array}$$

□

Similarly we can derive $\tau 4$ from $\tau 4'$.

The law $\tau 3'$ can be derived from our law $T2^{b}_{ur}$ and the law $\tau 4'$ is a direct instance of our law $T3_{ur}$.

12.5 Wang's Timed CCS

In [Wan90] Wang Yi proposes a dense timed CCS calculus with a delayable prefix $a.P$ (delayable for actions $a \neq \tau$), a time prefix $\epsilon(t).P$, maximal progress and a strong choice. In that paper he does not have time variables yet, so he can not give a proper expansion law. Therefore, he introduces in [Wan91b], [Wan91a] the construct $\epsilon(b).P$, where b is a bound in which time variables may occur, and the construct $a@v.P$, where v is a time variable of which the occurences in P become bound by the prefix $a@v$. This construct idles till it executes the action a at time t, after which it evolves in $P[t/v]$. Due to the addition of time variables, Wang could deduce an expansion law. The translation of Wang's Timed CCS into ACPur is given in Table 12.5. Note that we cannot put $(\!|a@v.P|\!) = \underline{\tilde{a}} \cdot (\!|P|\!)$, as the time variable v may occur in $(\!|P|\!)$.

$$\boxed{\begin{array}{lcl}
(\!|NIL|\!) & = & \underline{\tilde{\delta}} \\
(\!|\epsilon(b).P|\!) & = & (b) \cdot (\!|P|\!) \\
(\!|a@v.P|\!) & = & \int_{tt} \tilde{a}[v] \cdot ((\!|P|\!)) \\
(\!|P+Q|\!) & = & ((\!|P|\!) + (\!|Q|\!)) \gg U((\!|P|\!)) \cap U((\!|Q|\!)) \\
(\!|P|Q|\!) & = & \mathcal{U}_{\mathcal{T}}((\!|P|\!) \| (\!|Q|\!)) \\
(\!|P \backslash H|\!) & = & \partial_H((\!|P|\!)) \\
(\!|P[f]|\!) & = & \rho_f((\!|P|\!))
\end{array}}$$

Table 12.5: Translation of Wang's Timed CCS into ACPur, where $\gamma(a, \overline{a}) = \tau$

Jeffrey gives in [Jef91b] a generalization of Wang's timed CCS [Wan91a]. He has a non-delayable prefix, a time prefix $\epsilon t.P$ and maximal progress. The special feature is the generalized sum $\sum \mathcal{P}$ where $\mathcal{P}$ is a set of processes. He can express Wang's $a@v.P$ by $\sum\{\epsilon t.a.P \mid t \in \mathcal{T}\}$ where $\mathcal{T}$ is the underlying time domain. Moreover, he gives a sound and complete axiom system.

12.6 Wang's weak bisimulation

Wang [Wan91b] [Wan91a] has defined weak bisimulation equivalence as in Definition 11.10.2. Also Wang imposes a rootedness condition on a weak bisimulation, which coincides with ours, to obtain a (largest possible) congruence.

In Section 12.5 we have discussed that Wang's calculus is based on maximal progress. This means, in ACPur terms, that every process term must be considered in a context $\mathcal{U}_{\{\tau\}}(\ldots)$, and we can reduce every $\int_{v \in V} \tilde{\tau}[v] \cdot p$ to either $\tilde{\tau}[b] \cdot p$ or $\tilde{\delta}[b] \cdot p$. More precisely:

$$\mathcal{U}_{\{\tau\}}(\int_{v \in [b_0, b_1\rangle} \tilde{\tau}[v] \cdot p) \;=\; ([b_0, b_1\rangle \neq \emptyset) :\longrightarrow \tilde{\tau}[b_0] \cdot \mathcal{U}_{\{\tau\}}(p[b_0/v])$$

$$\begin{array}{ll}
\tau 1 & \underset{\cdot}{\tilde{\tau}} \cdot p \mathbin{+\!\!+} p \;=\; \underset{\cdot}{\tilde{\tau}} \cdot p \\[1em]
\tau 2 & \underline{\tilde{a}} \cdot (\underset{\cdot}{\tilde{\tau}} \cdot p \mathbin{+\!\!+} q) \;=\; \underline{\tilde{a}} \cdot (\underset{\cdot}{\tilde{\tau}} \cdot p \mathbin{+\!\!+} q) \mathbin{+\!\!+} \underline{\tilde{a}} \cdot p \\[1em]
\tau 3 & \underline{\tilde{a}} \cdot (\sum_i^{+\!\!+} (t_i) \cdot (\underline{\tilde{a}}_i \cdot p_i)) \\
 & =\; \underline{\tilde{a}} \cdot (\sum_i^{+\!\!+} (t_i) \cdot (\underline{\tilde{a}}_i \cdot p_i) \mathbin{+\!\!+} \tilde{\tau}[r] \cdot \sum_i^{+\!\!+} (t_i \dot{-} r) \cdot (\underline{\tilde{a}}_i \cdot p_i))
\end{array}$$

Table 12.6: Tau laws of Wang

and

$$\mathcal{U}_{\{r\}}(\int_{v \in \langle b_0, b_1 \rangle\!\rangle} \tilde{\tau}[v] \cdot p) \;=\; (\langle b_0, b_1 \rangle\!\rangle \neq \emptyset) :\!\rightarrow \tilde{\delta}[b_0]$$

Thus, for the image of Wang's Timed CCS in ACPur we can reduce the laws of Table 11.8 considerably.

For example, if we use the strong choice $p \mathbin{+\!\!+} q$ as abbreviation for $(p + q) \gg U(p) \cap U(q)$ then we can reduce the law $\mathrm{T2}^a_{ur}$ to

$$\mathrm{T2}^{+\!\!+}_{ur} \quad \tilde{\tau}[t] \cdot p = \tilde{\tau}[t] \cdot p \mathbin{+\!\!+} (t) \cdot p$$

and no $\gg [t, t]$ is needed as in $\mathrm{T2}^a_{ur}$.

In Table 12.6 we formulate the tau laws of Wang in ACPur, where $p \mathbin{+\!\!+} q$ must be considered as an abbreviation as given above. The symbol $\sum^{+\!\!+}$ denotes the generalized strong plus, thus $\sum^{+\!\!+}_{i \in \{1,\ldots,n\}} p_i$ abbreviates $p_1 \mathbin{+\!\!+} \ldots \mathbin{+\!\!+} p_n$.

The law $\tau 1$ is an immediate consequence of the above identity $\mathrm{T2}^{+\!\!+}_{ur}$. The law $\tau 2$ is an immediate consequence of our law $\mathrm{T3}_{ur}$, and, finally, $\tau 3$ can be derived from our branching law B_{ur}, that on its turn is derivable from $\mathrm{T1}_{ur}$ and $\mathrm{T2}^a_{ur}$.

12.7 Chen's Timed CCS

In [Che91],[Che92],[Che93] Liang Chen [Che91],[Che92],[Che93] presents a Timed CCS calculus with a weak choice and the prefix construct $a(v)|_e^{e'}.P$, where e, e' are so called *time expression*. These time expressions are similar to our bounds. The difference is that his time expressions may be of the form $max(e, e')$, so he allows conditional time expressions as well. For r, r' in the time domain $a(v)|_r^{r'}.P$ executes an a at some t, where $t \in [r, r']$, after which it evolves into $P[t/v]$. The process expression $a(v)|_r^{r'}.P$ is very close to our $\int_{v \in [r,r']} a[v] \cdot P$, the difference is that $a(v)|_r^{r'}.P$ is always allowed to idle until r', even if $r > r'$.

Chen's Timed CCS is not based on maximal progress. The translation of Chen's calculus into ACPur is given in Table 12.7. The translation of $(e)P$ is not completely sound, as we do not have conditional bounds in ACPur. Formally, we have to define the translation of $(e)P$ by induction to the structure of e, and we need defining rules like

$$(\!|(max(e,e')P|\!) = \begin{cases} & e \leq e' :\rightarrow (e')(\!|P|\!) \\ + & e' < e :\rightarrow (e)(\!|P|\!) \end{cases}$$

and similar rules for the other constructors of his time expressions.

Chen obtains a decidability result along a different route than we do. He introduces for every pair of processes P, Q a first order formula $WC(P,Q)$ which is the least condition such that P and Q are bisimilar. Decidability now follows from the decidability of the first order theory of the underlying time domain.

This method, however, is not at all constructive and does not lead to a completeness result. Therefore, he introduced a conditional axiom system. If two process expressions P, Q, possibly containing free time variables, are equal under the condition α, then he has $\alpha \vdash P = Q$. He constructs a boolean expression $\alpha(P,Q)$, which is also a condition in our terminology, that is equivalent with the first order formula $WC(P,Q)$. He then shows that $\alpha(P,Q) \vdash P = Q$ from which the completeness follows.

So, Chen has derivations which are relative to some condition and he has proof rules for dealing with the conditions. Intuitively $\alpha \vdash p = q$ corresponds to our $\vdash \{\alpha :\rightarrow p\} = \{\alpha :\rightarrow q\}$, the difference, however, is that his derivation uses proof trees while ours fits within pure equational reasoning.

$$\begin{array}{lcl}
(\!|NIL|\!) & = & \underline{\tilde{\delta}} \\
(\!|a(v)|_e^{e'}.P|\!) & = & \int_{v\in[e,e']} \tilde{a}[v] \cdot (\!|P|\!) + \tilde{\delta}[e'] \\
(\!|P+Q|\!) & = & (\!|P|\!) + (\!|Q|\!) \\
(\!|(e)P|\!) & = & (e) \cdot (\!|P|\!) \\
(\!|e \gg P|\!) & = & (-e) \cdot (\!|P|\!) \\
(\!|P|Q|\!) & = & (\!|P|\!) \| (\!|Q|\!) \\
(\!|P\backslash a|\!) & = & \partial_{\{a,\bar{a}\}}((\!|P|\!)) \\
(\!|P[f]|\!) & = & \rho_f((\!|P|\!))
\end{array}$$

Table 12.7: Translation of Chen's Timed CCS into ACPur, where $\gamma(a,\bar{a}) = \tau$

12.8 Chen's Weak Bisimulation

In his thesis [Che93] Chen has defined weak bisimulation, that he calls behavioral abstraction. His definition does not correspond directly with some of the definitions of weak bisimulations given before, as he does not have a two phase semantics. His tau laws are given for ACPur in Table 12.8.

We have already discussed that there are some subtle differences between Chen's time expressions and our bounds. Due to this difference we have to present his τ laws a little different than he does. For example his first τ-law his given below.

$$\tau 1 \qquad \tau(v)_0^e \cdot (\tilde{q} \cdot p + q) \;=\; \tau(v)_0^e \cdot (\tilde{q} \cdot p + q) + \tilde{q} \cdot p$$

His bounds are always positive, i.e. for all substitutions. Hence, no condition that $e \geq 0$ is needed.

Chen denotes the ultimate delay, or *maximal delay* as he calls it, of a process term P by $|P|$, following Moller & Tofts. Some of its axioms are given below.

$$\begin{aligned} |a(v)_e^{e'}.P| &= e' \\ |P+Q| &= \max(|P|, |Q|) \end{aligned}$$

Note, that the time expression $|P|$ is a conditional one, as $\max(e, e')$ abbreviates $\{e' \leq e :\rightarrow e\} + \{e < e' :\rightarrow e'\}$.

Chen formulates his third τ law as follows.

$$\begin{aligned} & w \notin fv(Q) \text{ and } |P| \leq b_0' \leq b_1' \\ & a(v)_{b_0}^{b_1} \cdot (P + \tau(w)_{b_0'}^{b_1'} \cdot (w \gg (b_0')Q) + (b_0')Q) \\ & = \\ & a(v)_{b_0}^{b_1} \cdot (P + \tau(w)_{b_0'}^{b_1'} \cdot (w \gg (b_0')Q)) \end{aligned}$$

According to Chen the condition $|P| \leq b_0 \leq b_1$ must be read as "if $|P| \leq b_0' \leq b_1'$ is valid for all substitutions". Indeed Chen does not have an axiom system for his time expressions and conditions. This interpretation is rather strict as in case $v \in [b_0, b_1] \Rightarrow |P| \leq b_0' \leq b_1'$ is valid for all subsitutions (i.e. reduces to true in the context of an axiom system like CA) the law is sound as well, though it is not clear how such an identity can be derived. Take for example $b_0 = 1$, $b_1 = 2$, $b_0' = v$, $b_1' = 2$ and $P = a'(w)_1^1$, then obviously $v \in [1, 2] \Rightarrow 1 \leq v \leq 2$.

In the context of ACP*ur* we have our ultimate delay instead of $|P|$. Since we do not want to deal with conditional bounds we have decided to define the ultimate delay on terms with prefixed integration as a condition. The condition $|P| \leq b_0'$ expresses that P may not contribute any behavior after b_0'. In ACP*ur* we express this by putting the process term in the scope of $\ldots \gtrsim b_0'$, where $p \gtrsim b$ abbreviates $p \gg (-\infty, b]$. Hence, no reference to the ultimate delay of p is needed. Furthermore, we put the condition $b_0' \leq b_1'$ within the process terms and we obtain the following formulation. So, we formulate Chen's third τ-law as follows, where $\tilde{a}[v]_{b_0}^{b_1} \cdot p$ denotes $\int_{v \in [b_0, b_1]} \tilde{a}[v] \cdot p$.

$$\begin{aligned} & w \notin fv(Q) \\ & \tilde{a}[v]_{b_0}^{b_1} \cdot b_0' \leq b_1' :\rightarrow (\ p \gtrsim b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot (b_0') \cdot q + (b_0') \cdot q\ ) \\ & = \\ & \tilde{a}[v]_{b_0}^{b_1} \cdot b_0' \leq b_1' :\rightarrow (\ p \gtrsim b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot (b_0') \cdot q\ ) \end{aligned}$$

$$\tau 1\ \tilde{\tau}[v]_0^b \cdot (\tilde{a} \cdot p + q) \;=\; \tilde{\tau}[v]_0^b \cdot (\tilde{a} \cdot p + q) + b \geq 0 :\rightarrow \tilde{a} \cdot p[0/v]$$

$$\tau 2\ \tilde{a}[v]_{b_0}^{b_1} \cdot (\tilde{\tau} \cdot p + q) \;=\; \tilde{a}[v]_{b_0}^{b_1} \cdot p + \tilde{a}[v]_{b_0'}^{b_1'} \cdot (\tilde{\tau} \cdot p + q)$$

$$\begin{array}{l}
\tau 3\ w \notin fv(q) \\
\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1' \;:\rightarrow\; (p \gtrsim b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot (b_0') \cdot q + (b_0') \cdot q)\} \\
= \\
\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1' \;:\rightarrow\; (p \gtrsim b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot (b_0') \cdot q)\}
\end{array}$$

$$\begin{array}{l}
\tau 4\ w \notin fv((b_0')p + (b_1')\delta) \\
\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1' \;:\rightarrow\; (\tilde{\tau}[w]_{b_0'}^{b_1'} \cdot ((-w) \cdot ((b_0') \cdot p + (b_1') \cdot \delta)))\} \\
= \\
\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1' \;:\rightarrow\; ((b_0') \cdot p + (b_1') \cdot \tilde{\delta})\}
\end{array}$$

$$\begin{array}{l}
\tau 5\ w \notin fv((b_0') \cdot p + (b_1) \cdot \tilde{\delta}) \\
\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1' \;:\rightarrow\; (p \gtrsim b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot ((b_0') \cdot q + (b_1') \cdot \tilde{\delta})))\} \\
= \\
\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1' \;:\rightarrow\; (p \gtrsim b_0' + (b_0') \cdot q + (b_1') \cdot \tilde{\delta})\}
\end{array}$$

Table 12.8: τ-laws of Chen

Note, that this formulation results in an identity for $b_0 = 1$, $b_1 = 2$, $b_0' = v$, $b_1' = 2$ and $p = \tilde{a}'[w]_1^1$. The law $\tau 1$ can be derived from our law $\mathrm{T2}_{ur}^b$ and the law $\tau 2$ can be derived from our law $\mathrm{T3}_{ur}$. Moreover, $\tau 3$ follows from $\mathrm{T1}_{ur}$ and $\mathrm{T2}_{ur}^a$ as is shown below.

Proposition 12.8.1 $\mathrm{ACP}ur + \mathrm{T1}_{ur} + \mathrm{T2}_{ur} \vdash \tau 3$

Proof. It is left to the reader to proof that $\mathrm{ACP}ur + \mathrm{T1}_{ur} + \mathrm{T2}_{ur} \vdash \mathrm{T2}'_{ur}$ where

$$\mathrm{T2}'_{ur} \quad w \notin fv(p) \quad \alpha = \langle\!|b_0, b_1|\!\rangle \neq \emptyset$$

$$\begin{aligned}&\textstyle\int_{\alpha\wedge\beta} \tilde{a}[v] \cdot (\int_{w \in \langle\!|b_0,b_1|\!\rangle} \tilde{\tau}[w] \cdot (-w) \cdot p + q \ggeq b_1) \\ &= \textstyle\int_{\alpha\wedge\beta} \tilde{a}[v] \cdot (\int_{w \in \langle\!|b_0,b_1|\!\rangle} \tilde{\tau}[w] \cdot (-w) \cdot p + b_0 \gg p + q \ggeq b_1)\end{aligned}$$

By this identity, Our reformulation of Chen's third τ-law is a direct consequence of this derived identity $\mathrm{T2}'_{ur}$.

$$\begin{aligned}&\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1'\} :\rightarrow \{(p \ggeq b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot (b_0') \cdot q)\} \\ \overset{\mathrm{T2}_{ur}^b}{=} \; &\tilde{a}[v]_{b_0}^{b_1} \cdot \{b_0' \leq b_1'\} :\rightarrow \{(p \ggeq b_0' + \tilde{\tau}[w]_{b_0'}^{b_1'} \cdot (-w) \cdot (b_0') \cdot q + (b_0') \cdot q)\}\end{aligned}$$

□

Similarly we can derive $\tau 4$ from $\mathrm{T1}_{ur}$. Finally, $\tau 5$ follows from our branching law B_{ur}.

12.9 ATP of Nicollin and Sifakis

In [NRSV90] and [NS91] Nicollin et al. introduced the discrete time process algebra ATP (Algebra for Timed Processes). ATP is built from operators from CCS and ACP. It has an immediate prefix aP and a strong choice, denoted by $\oplus$. The additional feature is the binary unit delay operator $\lfloor P \rfloor(Q)$ which either executes an immediate action from P or it idles one time unit and evolves into Q. The unit delay operator is generalized to $\lfloor P \rfloor^t(Q)$ which allows actions from P until t time units has passed after which it evolves into Q. In [NSY91] Nicollin et al. generalized ATP with a general time domain. In that paper they denote the immediate prefix by $\dot{a}P$ and they also introduce a delayable prefix $\tilde{a}P$. Furthermore, the (discrete) delay operator $\lfloor P \rfloor^t(Q)$ is generalized to the *time out* operator $P \overset{t}{\rhd} Q$. For a discrete time domain these operators coincide.

The processes in ATP are deadlock free, i.e., each process can either do an action or it can idle. This is certainly not the case in real time ACP, and in ACPur in particular. As a consequence the translation of the encapsulation operator is not that straightforward. In ATP we have $\partial_{\{a\}}(aP) = \delta$ where the ATP δ corresponds to our $\underset{\sim}{\tilde{\delta}}$, and aP is an immediate prefix like $\underset{\sim}{\tilde{a}}.p$. Hence, we need a variant of our encapsulation that can rename $\underset{\sim}{\tilde{a}}$ into $\underset{\sim}{\tilde{\delta}}$. We denote this encapsulation by ∂_H^∞ and

	$(\!\vert\delta\vert\!)$	$=$	$\underline{\tilde{\delta}}$
Only in [NSY91]	$(\!\vert\dot{a}P\vert\!)$	$=$	$\underset{\cdot}{\tilde{a}} \cdot (\!\vert P\vert\!)$
In [NS90] as aP	$(\!\vert\tilde{a}P\vert\!)$	$=$	$\underline{\tilde{a}} \cdot (\!\vert P\vert\!)$
	$(\!\vert P \oplus Q\vert\!)$	$=$	$((\!\vert P\vert\!) + (\!\vert Q\vert\!)) \gg U((\!\vert P\vert\!)) \cap U((\!\vert Q\vert\!))$
In [NS90] as $\lfloor P\rfloor^t(Q)$	$(\!\vert P \overset{t}{\rhd} Q\vert\!)$	$=$	$(\!\vert P\vert\!) \gg [0,t\rangle + (t) \cdot Q$
	$(\!\vert P\Vert Q\vert\!)$	$=$	$(\!\vert P\vert\!)\Vert(\!\vert Q\vert\!)$
Only in [NS90]	$(\!\vert P \mathbin{\|\!\_} Q\vert\!)$	$=$	$(\!\vert P\vert\!) \mathbin{\|\!\_} (\!\vert Q\vert\!)$
Only in [NS90]	$(\!\vert P\mid Q\vert\!)$	$=$	$(\!\vert P\vert\!)\mid(\!\vert Q\vert\!)$
	$(\!\vert\partial_H(P)\vert\!)$	$=$	$\partial_H^\infty((\!\vert P\vert\!))$
Only in [NSY91]	$(\!\vert\lceil P\rceil^t(Q)\vert\!)$	$=$	$\lceil(\!\vert P\vert\!)\rceil^t((\!\vert Q\vert\!))$

Table 12.9: Translation of ATP into ACPur

we call it the *time stop free encapsulation*. Its action rules and its axioms are given in Table 12.11. For a time closed process term p we can describe $\partial_H^\infty(p)$ as follows; if $\partial_H(p) = p' + \tilde{\delta}[t]$ where p' does not contain ∂_H and p' cannot idle till t (i.e., $\neg(U_t(p'))$), then $\partial_H^\infty(p) = p' + \underline{\tilde{\delta}}$. Otherwise, it $\partial_H^\infty(p)$ is just p'. Some examples are given below.

$$\begin{array}{lcl}
\partial_{\{a\}}^\infty(\int_{v\in[0,2]} \tilde{a}[v]) & = & \underline{\tilde{\delta}} \\
\partial_{\{a\}}^\infty(\int_{v\in[0,2\rangle} \tilde{a}[v]) & = & \int_{v\in[0,2\rangle} \tilde{\delta}[v] \\
\partial_{\{a\}}^\infty(\int_{v\in[0,2]} \tilde{a}[v] + \tilde{b}[1]) & = & \underline{\tilde{\delta}} \\
\partial_{\{a\}}^\infty(\int_{v\in[0,2]} \tilde{a}[v] + \tilde{b}[2]) & = & \tilde{b}[2]
\end{array}$$

The unary operator $\partial_H^\infty(p)$ is axiomatized by means of an auxiliary binary operator, denoted by $p\ \partial_H^\infty\ q$. This latter operator distributes over all the summands of its left argument p. Since $\underset{\cdot}{\tilde{\delta}}\ \partial_H^\infty\ \underset{\cdot}{\tilde{\delta}} = \underline{\tilde{\delta}}$ we have as well

$$\underset{\cdot}{\tilde{\delta}}\ \partial_H^\infty\ q \quad = \quad (q = \underset{\cdot}{\tilde{\delta}}) :\rightarrow \underline{\tilde{\delta}}$$

where $q = \underset{\cdot}{\tilde{\delta}}$ is an abbreviation that expresses that all conditions of the initial integrals of q are false.

$$(\ (\sum_i \int_{\alpha_i} \tilde{a}_i[v_i] \cdot p_i + \sum_j \int_{\beta_j} \tilde{\delta}[w_j]) \ = \ \underset{\cdot}{\tilde{\delta}}\ ) \quad \overset{abb}{=} \quad \neg(\vee_i \alpha_i \vee \vee_j \beta_j)$$

For similar reasons we have the conditions $\neg(\alpha) \wedge q = \underset{\cdot}{\tilde{\delta}}$, and $V = \emptyset \wedge q = \underset{\cdot}{\tilde{\delta}}$ in the axioms IE3-6. We have also the following abbreviations:

$$\begin{array}{lcll}
\langle\!| b_0, b_1 |\!\rangle_{\geq 0} = \emptyset & \overset{abb}{=} & \langle\!| b_0, b_1 |\!\rangle = \emptyset \vee 0 \leq b_1 & \text{if } |\!\rangle = \rangle \\
 & \overset{abb}{=} & \langle\!| b_0, b_1 |\!\rangle = \emptyset \vee 0 < b_1 & \text{otherwise} \\
\langle\!| b_0, b_1 |\!\rangle_{\geq 0} \neq \emptyset & \overset{abb}{=} & \neg(\langle\!| b_0, b_1 |\!\rangle_{\geq 0} = \emptyset) &
\end{array}$$

$$\frac{p \xrightarrow{t} p' \quad t \leq r}{\lceil p \rceil^r(q) \xrightarrow{t} \lceil p' \rceil^{r-t}(q)} \qquad \frac{q \xrightarrow{\mu} q'}{\lceil p \rceil^0(q) \xrightarrow{\mu} q'}$$

$$\frac{p \xrightarrow{t} p' \quad q \xrightarrow{r} q'}{\lceil p \rceil^t(q) \xrightarrow{t+r} q'}$$

$$\frac{p \xrightarrow{\tilde{a}} p' \quad a \neq \xi \quad t > 0}{\lceil p \rceil^t(q) \xrightarrow{\tilde{a}} \lceil p' \rceil^t(q)} \qquad \frac{p \xrightarrow{\xi} p' \quad t > 0}{\lceil p \rceil^t(q) \xrightarrow{\tilde{\tau}} p'}$$

ED1	$\lceil p + q \rceil^b(z)$	$=$	$\lceil p \rceil^b(z) + \lceil q \rceil^b(z)$
ED2 $v \notin var(b)$			
	$\lceil \int_\alpha \tilde{\xi}[v] \cdot p \rceil^b(q)$	$=$	$\int_{\alpha \wedge v<b} \tilde{\xi}[v] \cdot p + U_b(\alpha) :\longrightarrow (b) \cdot q$
ED3 $a \neq \xi,\ v \notin var(b)$			
	$\lceil \int_\alpha \tilde{a}[v] \cdot p \rceil^b(q)$	$=$	$\int_{\alpha \wedge v<b} \tilde{a}[v] \cdot (\lceil p \rceil^{b-v} q) + U_b(\alpha) :\longrightarrow (b) \cdot q$

Table 12.10: Action rules and axioms for the execution delay $\lceil p \rceil^b(q)$

In [NSY91] Nicollin et al. also introduce the *execution delay*, denoted by $\lceil P \rceil^r(Q)$. It behaves as P until time r; at time r process P is aborted and Q is started. However, if P performs the special action ξ, called the *cancel* action, then the delay is cancelled, and the subsequent behavior is that of P after ξ. The cancel execution is internal, i.e. it is renamed to τ. This operator is not expressible in ACP*ur* and we have to add it as well. Its action rules and axioms are given in Table 12.10.

$$\frac{p \xrightarrow{t} p'}{\partial_H^\infty(p) \xrightarrow{t} \partial_H^\infty(p')} \qquad \frac{p \xrightarrow{\tilde{a}} p' \quad a \notin H}{\partial_H^\infty(p) \xrightarrow{\tilde{a}} \partial_H^\infty(p')}$$

$$\frac{\forall r\ p \overset{r}{\not\longrightarrow} \quad \forall a \in A - H\ p \overset{a}{\not\longrightarrow}}{\partial_H^\infty \xrightarrow{t} \underline{\tilde{\delta}}}$$

$$\frac{p \xrightarrow{t} p' \quad q \xrightarrow{t} q'}{p\ \partial_H^\infty\ q \xrightarrow{t} p'\ \partial_H^\infty\ q'} \qquad \frac{p \xrightarrow{t} p' \quad q \overset{t}{\not\longrightarrow}}{p\ \partial_H^\infty\ q \xrightarrow{t} \partial_H^\infty(p')}$$

$$\frac{p \xrightarrow{\tilde{a}} p' \quad a \notin H}{p\ \partial_H^\infty\ q \xrightarrow{\tilde{a}} \partial_H^\infty(p')} \qquad \frac{\begin{array}{c}\forall t\ p \overset{t}{\not\longrightarrow} \quad \forall t\ q \overset{t}{\not\longrightarrow} \\ \forall a \in A - H\ p \overset{\tilde{a}}{\not\longrightarrow}\end{array}}{p\ \partial_H^\infty\ q \xrightarrow{t} \underline{\tilde{\delta}}}$$

$$\begin{array}{lllll}
\text{IE1} & \partial_H^\infty(p) & & = & p\ \partial_H^\infty\ p \\
\text{IE2} & (p+q)\ \partial_H^\infty\ z & & = & p\ \partial_H^\infty\ z + q\ \partial_H^\infty\ z \\
\text{IE3} & (\alpha :\rightarrow p)\ \partial_H^\infty\ q & & = & \\
 & \quad\ \ \{\alpha & :\rightarrow & (p\ \partial_H^\infty\ q)\} & \\
 & + \ \ \{(\neg(\alpha) \wedge q = \tilde{\delta} & :\rightarrow & \underline{\tilde{\delta}}\} & \\
\text{IE4}\ a \notin H_\delta & (\int_{v \in V} \tilde{a}[v] \cdot p)\ \partial_H^\infty\ q & & = & \\
 & \quad\ \ \{V_{\geq 0} \neq \emptyset & :\rightarrow & \int_{v \in V} \tilde{a}[v] \cdot \partial_H^\infty(p)\} & \\
 & + \ \ \{V_{\geq 0} = \emptyset \wedge q = \tilde{\delta} & :\rightarrow & \underline{\tilde{\delta}}\} & \\
\text{IE5}\ a \in H_\delta & (\int_{v \in (\!| b_0, b_1]} \tilde{a}[v] \cdot p)\ \partial_H^\infty\ q & & = & \\
 & \quad\ \ \{(\!| b_0, b_1]_{\geq 0} \neq \emptyset \wedge U_{b_1}(q) & :\rightarrow & \int_{v \in (\!| b_0, b_1]} \tilde{\delta}[v]\} & \\
 & + \ \ \{(\!| b_0, b_1]_{\geq 0} \neq \emptyset \wedge \neg(U_{b_1}(q)) & :\rightarrow & \underline{\tilde{\delta}}\} & \\
 & + \ \ \{(\!| b_0, b_1]_{\geq 0} = \emptyset \wedge q = \tilde{\delta} & :\rightarrow & \underline{\tilde{\delta}}\} & \\
\text{IE6}\ a \in H_\delta & (\int_{v \in (\!| b_0, b_1\rangle} \tilde{a}[v] \cdot p)\ \partial_H^\infty\ q & & = & \\
 & \quad\ \ \{(\!| b_0, b_1\rangle_{\geq 0} \neq \emptyset & :\rightarrow & \int_{v \in (\!| b_0, b_1\rangle} \tilde{\delta}[v]\} & \\
 & + \ \ \{(\!| b_0, b_1\rangle_{\geq 0} = \emptyset \wedge q = \tilde{\delta} & :\rightarrow & \underline{\tilde{\delta}}\} &
\end{array}$$

Table 12.11: Action rules and axioms for the time stop free encapsulation $\partial_H^\infty(X)$

12.10 TPL of Hennessy & Regan

Hennessy and Regan have presented in [HR90] a timed CCS variant, called TPL (Temporal Process Language). TPL has a delayable prefix (for external actions), maximal progress and a strong choice. Due to the delay prefix $\sigma.P$, which expresses the delay P by one time unit, the underlying time domain is discrete. The main difference is that Hennessy & Regan use preorders based on testing equivalences as their semantic domain, instead of bisimulation equivalence. Therefore, we will give a translation that corresponds only to their operational semantics modulo bisimulation equivalence, see Table 12.12.

$$\begin{array}{lcl}
(\!|nil|\!) & = & \underline{\tilde{\delta}} \\
(\!|\Omega|\!) & = & \tilde{\delta} \\
(\!|\sigma.P|\!) & = & (1)\cdot(\!|P|\!) \\
(\!|a.P|\!) & = & \underline{\tilde{a}}\cdot(\!|P|\!) \\
(\!|\tau.P|\!) & = & \tilde{\tau}\cdot(\!|P|\!) \\
(\!|P+Q|\!) & = & ((\!|P|\!)+(\!|Q|\!)) \gg (U((\!|P|\!))\cap U((\!|Q|\!))) \\
(\!|\lfloor P\rfloor(Q)|\!) & = & (\!|P|\!) \gg [0,0] + (1)\cdot(\!|Q|\!) \\
(\!|P|Q|\!) & = & \mathcal{U}_\tau((\!|P|\!)\|(\!|Q|\!)) \\
(\!|P[f]|\!) & = & \rho_f((\!|P|\!)) \\
(\!|P\backslash a|\!) & = & \partial_{\{a\}}((\!|P|\!))
\end{array}$$

Table 12.12: Translation of Hennessy & Regan's TPL into ACPur, where $\gamma(a,\overline{a})=\tau$

12.11 TIC of Quemada, de Frutos and Azcorra

Quemada, de Frutos and Azcorra have presented in [QdFA93] a timed calculus, based on the syntax of LOTOS [ISO87]. The calculus has a discrete time domain, furthermore it has timed deadlocks, a time stamped prefix and weak choice. It has also a prefix construct which corresponds with our prefix integration where the bounds are taken from the time domain, so no time variables occur. Moreover, it has an auxiliary construct, similar to our $(t)\cdot\ldots$. The translation of TIC into ACPur is given in Table 12.13. In that Table we denote by $H\to\tau$ the mapping that maps all actions in H to τ and that leaves all actions not in H unchanged. It is also possible to translate $(\!|a[t,t']P|\!)$ into $\int_{v\in[t,t']}\tilde{a}[v]\cdot(\!|P|\!)+\tilde{\delta}[t']$. We have chosen for the translation into $t\gg(\underline{\tilde{a}}\cdot(\!|P|\!))\ \underline{\gg}\ t'+\tilde{\delta}[t']$ in order to stress that $(\!|a[t,t']P|\!)$ does not introduce any time variables.

The authors give also an expansion law, similar to the one we showed in Section 12.2. Since the time domain is discrete their generalized sum can indeed be

$(\![stop(t)]\!)$	$=$	$\tilde{\delta}[t]$
$(\![Idle]\!)$	$=$	$\tilde{\underline{\delta}}$
$(\![at;P]\!)$	$=$	$\tilde{a}[t]\cdot(\![P]\!)$
$(\![a[t,t']P]\!)$	$=$	$(\tilde{\underline{a}}\cdot(\![P]\!))\gg[t,t']+\tilde{\delta}[t']$
$(\![P\Box Q]\!)$	$=$	$(\![P]\!)+(\![Q]\!)$
$(\![P\|H\|Q]\!)$	$=$	$\partial_{A-H}((\![P]\!)\,\|\,(\![Q]\!))$
$(\![hide\,H\,in\,P]\!)$	$=$	$\rho_{H\to\tau}((\![P]\!))$
$(\![P[f]]\!)$	$=$	$\rho_f((\![P]\!))$
$(\![Age(t,P)]\!)$	$=$	$(-t)\cdot(\![P]\!))$

Table 12.13: Translation of TIC into ACPur, $(\gamma(a,a)=a)$

considered as an abbreviation of a finite sum, and no time variables are needed.

12.12 Weak bisimulation in TIC

Quemada et al. define weak bisimulation a little different from Definition 11.10.2. Their τ laws are given in Table 12.14.

$\tau 1$	$at;\tau t';p$	$=$	$at;Age(-t',p)$
$\tau 2$	$\tau t;p$	$=$	$Age(-t,p)\Box\tau t;p$
$\tau 3$	$at;(p\Box\tau t';q)$	$=$	$at;(p\Box\tau t';q)\Box at;Age(-t',q)$

Table 12.14: The tau laws in TIC

The first law can be formulated in ACPur as follows

$$\tilde{a}[t]\cdot\tilde{\tau}[t']\cdot p=\tilde{a}[t]\cdot(t)\cdot p,$$

which is a direct instance of our law T1$_{ur}$. The law $\tau 2$ can be formulated in ACPuras follows

$$\tau[t]\cdot p=\tau[t]\cdot p+(t)\cdot p$$

which is a direct instance of T2$^{a}_{ur}$. Finally, $\tau 3$ is formulated as

$$\tilde{a}[t]\cdot(X+\tau[t']\cdot Y)=\tilde{a}[t]\cdot(X+\tau[t']\cdot Y)+\tilde{a}[t]\cdot(t')\cdot Y.$$

This identity is certainly more general than our $T3_{ur}$, which corresponds only with the case where $t' = 0$. As a matter of fact, the law $\tau 3$ identifies the following.

$$\tau 3 \vdash \tilde{a}[1] \cdot (\tilde{b}[1] + \tilde{\tau}[2] \cdot \tilde{c}[1]) = \tilde{a}[1] \cdot (\tilde{b}[1] + \tilde{\tau}[2] \cdot \tilde{c}[1]) + \tilde{a}[1] \cdot \tilde{c}[3]$$

Though these process terms can be distinguished within $\underline{\leftrightarrow}_{rw}$, by the context $\partial_b(...\|\tilde{b}[2])$.

$$\begin{aligned} &\partial_b(\ (\tilde{a}[1] \cdot (\tilde{b}[1] + \tilde{\tau}[2] \cdot \tilde{c}[1])) \parallel \tilde{b}[2]\ ) &&= \tilde{a}[1] \cdot \tilde{b}[1] \\ &\partial_b(\ (\tilde{a}[1] \cdot (\tilde{b}[1] + \tilde{\tau}[2] \cdot \tilde{c}[1]) + \tilde{a}[1] \cdot \tilde{c}[3]) \parallel \tilde{b}[2]\ ) &&= \\ &&& \tilde{a}[1] \cdot \tilde{b}[1] + \tilde{a}[1] \cdot \tilde{\delta}[1] \end{aligned}$$

This example is similar to the counterexample against weak bisimulation in BPA$\rho\delta$, see Section 8.7.

12.13 Other related work

In the previous sections we have discussed the papers that can be explained in detail by a translation into ACPur. However, there are several other papers, that can not be discussed in detail in the context of ACPur, these papers are discussed briefly below.

Timed CSP

Mike Reed & Bill Roscoe have presented (dense) Timed CSP in [RR88], they have given a denotational semantics based on timed traces and timed failures. In [Sch92] Steve Schneider has given an operational semantics for a slightly simplified version of Timed CSP. Alan Jeffrey has developed Discrete Timed CSP ([Jef91a]), due to this simplification he could give a complete axiomatization.

Other references to timed process algebras

An extension of the specification language LOTOS with time is discussed in [BL91]. For a combination of time and probability we refer to [Han91]. For a presentation of a real timed theory incorporating true concurrency and event refinement we refer to [Mur91].

Assertional Methods

In this thesis we consider an algebraic approach to real time systems. Besides the algebraic approach, also other approaches exist in the literature.

Prominent are approaches based on temporal logic extended with quantative time. An example of this approach with applications to real time distributed systems is [Koy89]. Also, we can find the approach of extending Hoare triples with quantitive time, see e.g. [Hoo91]. It seems that a logical approach is more suited to express high level properties of systems, that abstract from the time points of internal choices. On the other hand, the algebraic approach seems more suited for system specification. Perhaps, the most promising future line of research would be to try to combine algebraic specification featuring equational reasoning with logical properties featuring proof systems.

Another interesting development is the extension of *graphs* and *automata* with time ([AD90], [MMT91] [LV91]). An advantage of timed graphs is that they are finite systems, so model checking is possible. In [NSY91] a translation from ATP into timed graphs is given.

Appendix A

Bounds and Conditions

A.1 Introduction

In this appendix we elaborate the bounds and conditions in detail. First we give a very simple construction that gives for each bound a bound in normal form. Then, we give an axiom system CA for bounds, see Table A.1.

Using the axioms of CA we can prove the Refinement Lemma, on which the decidibility of BPA$\rho\delta$I depends. Finally, we prove that if two conditions are equivalent, i.e., they coincide for all possible substitutions, then they can be proven equal within CA.

A.2 Bounds in normal form

Proposition A.2.1 (Bounds in Normal Form) *For every $b \in$ Bound there is a bound b' of the form*

$$r_1 \cdot v_1 + \ldots + r_n \cdot v_n + t \quad (n \geq 0)$$

such that all v_i's are different and $r_i \in$ Time$\backslash 0$, and $b = b'$.

Proof. Due to the axioms of an ordered field, we may consider the bounds modulo associativity and commutativity of the $+$.

- We rewrite b into $b + 0$, such that we obtain a bound of the form $b' + t$, and we rewrite each v into $1 \cdot v$, to guarantee that it occurs in the form $r \cdot v$.
- We rewrite each $r \cdot (b_0 + b_1)$ into $r \cdot b_0 + r \cdot b_1$ and $r_0 \cdot (r_1 \cdot b)$ into $(r_0 \cdot r_1) \cdot b$, such that we obtain a bound of the form. $r_1 \cdot v_1 + \ldots + r_n \cdot v_n + t_1 + \ldots + t_m$, where $r_i, t_j \in T(\mathcal{S})$.
- We rewrite $r \cdot v + r' \cdot v$ into $(r + r') \cdot v$ whenever possible.

  We replace each $c \in T(\mathcal{S})$ by its $c_t \in$ *Time*. Finally, we remove all summands $0 \cdot v$.

□

This Proposition implies that we can consider the set of bounds ($\neq -\infty, \infty$) to be closed under division since

$$\begin{array}{rl} & r^{-1} \cdot b \\ = & r^{-1} \cdot (r_1 \cdot v_1 + \ldots + r_n \cdot v_n + t) \\ = & ((r^{-1} \cdot r_1) \cdot v_1 + \ldots + (r^{-1} \cdot r_n) \cdot v_n + (r^{-1} \cdot t) \end{array}$$

and we can replace every $r^{-1} \cdot t'$ by some $t'' \in Time$.

In the sequel a bound b will be in general taken from *Bound*, unless otherwise stated.

A.3 A Proof System for Conditions

Proposition A.3.1 *For any $t_0, t_1 \in Time$ and $b_0, b_1 \in Bound$ we have the following identities in CA:*

$$\begin{array}{lcl} t_0 < t_1 & = & \mathit{ff} \quad \text{if } t_0 \not< t_1 \\ t_0 = t_1 & = & \mathit{tt} \quad \text{if } t_0 = t_1 \\ t_0 = t_1 & = & \mathit{ff} \quad \text{if } t_0 \neq t_1 \\ \neg(\mathit{ff}) & = & \mathit{tt} \\ \alpha \wedge \mathit{ff} & = & \mathit{ff} \\ \alpha \vee \mathit{ff} & = & \alpha \\ \alpha \vee \neg(\alpha) & = & \mathit{tt} \\ \neg(\alpha \vee \beta) & = & \neg(\alpha) \wedge \neg(\beta) \\ & & \\ b_0 < b_1 \vee b_1 < b_0 \vee b_0 = b_1 & = & \mathit{tt} \\ V \sim W \wedge (v \in V \vee v \in W) & = & V \sim W \wedge v \in V \cup W \\ r \cdot c_0 < r \cdot c_1 & = & c_1 < c_0 \quad \text{if } r < 0 \end{array}$$

Proof. Omitted.

A.4 The Refinement Lemma

Lemma 4.2.2 ([Refinement Lemma]) *Fix a time variable v. For each condition α there is an equivalent condition of the form $\vee_j(\beta_j \wedge v \in V_j)$, where $var(\beta_j) \cup var(V_j) \subseteq var(\alpha) \backslash \{v\}$ for all j.*

Proof. We reduce α, using equalities from the axiom system CA for conditions (see Table A.1). First, rewrite α to a condition of the form $\vee_i \gamma_i$, with each γ_i of the form $\wedge_j(b_j < b'_j) \wedge \wedge_k(c_k = c'_k)$. Reduce the bounds in γ_i to normal form (see Proposition A.2.1), i.e., to the form $r_1 \cdot v_1 + ... + r_l \cdot v_l + t$. In each (in)equality, collect factors $r \cdot v$ at one side, and collect the remaining parts of the bounds on the other side, such that either v is deleted from the (in)equality, or it takes the form $r \cdot v < b$ or $r \cdot v = b$, with $r \neq 0$ and $v \notin var(b)$. In the latter case, multiply both

$$\begin{array}{rrcl}
 & \alpha \wedge \beta & = & \beta \wedge \alpha \\
 & (\alpha \wedge \beta) \wedge \gamma & = & \alpha \wedge (\beta \wedge \gamma) \\
 & \alpha \wedge (\beta \vee \gamma) & = & (\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \\
 & \alpha \wedge (\alpha \vee \beta) & = & \alpha \\
 & & & \\
 & \alpha \wedge tt & = & \alpha \\
 & \alpha \vee tt & = & tt \\
 & & & \\
 & \neg(\alpha \wedge \beta) & = & \neg\alpha \vee \neg\beta \\
 & \neg tt & = & \mathit{ff} \\
 & \neg(\neg\alpha) & = & \alpha \\
 & & & \\
 & \neg(b_0 = b_1) & = & (b_0 < b_1) \vee (b_1 < b_0) \\
 & \neg(b_0 < b_1) & = & (b_0 = b_1) \vee (b_1 < b_0) \\
 & & & \\
t_0 < t_1 \Longrightarrow & t_0 < t_1 & = & tt \\
 & & & \\
 & b_0 = b_1 \wedge b_0 = b & = & b = b_1 \wedge b_0 = b \\
 & b_0 < b_1 \wedge b_0 = b & = & b < b_1 \wedge b_0 = b \\
 & & & \\
 & b_0 + b < b_1 + b & = & b_0 < b_1 \\
t > 0 \Longrightarrow & t \cdot b_0 < t \cdot b_1 & = & b_0 < b_1 \\
 & & & \\
 & b_0 < b \wedge b < b_1 \wedge b_0 < b_1 & = & b_0 < b \wedge b < b_1
\end{array}$$

$(t, t_0, t_1 \in T(\mathcal{S}),\ b, b_0, b_1 \in \mathit{Bound})$

Table A.1: The axiom system CA for conditions

sides with $r^- \in \mathit{Time}$, and replace $1 \cdot v$ by v. Hence, we can reduce each γ_i to an equivalent condition γ_i' of the form

$$\gamma \ \wedge \ \bigwedge_{j\in J} b_j < v \ \wedge \ \bigwedge_{k\in K} v < c_k \ \wedge \ \bigwedge_{l\in L} v = d_l$$

where v does not occur in γ, b_j, c_k, d_l. We show that such a γ_i' is equivalent to a condition of the form $\bigvee_j(\beta_j \wedge v \in V_j)$, with $v \notin var(\beta_j) \cup var(V_j)$.

First, suppose $L \neq \emptyset$. Fix an $l_0 \in L$ and put $d = d_{l_0}$. Then the following condition is equivalent to γ_i'.

$$(\gamma \wedge \bigwedge_{j\in J} b_j < d \wedge \bigwedge_{k\in K} d < c_k \wedge \bigwedge_{l\in L} d = d_l) \ \wedge \ v \in [d,d]$$

So we may assume $L = \emptyset$. Moreover, we may assume $J \neq \emptyset$ and $K \neq \emptyset$, because we can always add conditions $-\infty < v$ and $v < \infty$, as they abbreviate *tt*. Then the following condition is equivalent to γ_i'.

$$\bigvee_{(j,k)\in J\times K} (\gamma \wedge \bigwedge_{j'\in J} b_{j'} \leq b_j \wedge \bigwedge_{k'\in K} c_k \leq c_{k'} \wedge v \in \langle b_j, c_k \rangle)$$

□

A.5 An Axiomatization for Conditions

Finally, Table A.1 contains an axiom system CA for conditions. We have the following proposition:

Proposition 4.2.1[**Soundness and Completeness of CA**]

$$\models \alpha = \beta \quad \Longleftrightarrow \quad \mathrm{CA} \vdash \alpha = \beta$$

Proof. The soundness, i.e., the case $\Longleftarrow$, is left to the reader.

For the completeness, i.e., the case $\Longrightarrow$, we use induction to the number of variables that occur in α or β. In case this number is zero it is left to the reader to check that α and β reduce to either *tt* or *ff*.

So assume that we have proved the case for n variables, and let α and β contain $n+1$ variables. Fix a variable v that occurs in α or in β. Using the construction from the proof of the Refinement Lemma, we can deduce in CA:

$$\begin{aligned} \alpha &= \bigvee_i \gamma_i \wedge (v \in V_{i1} \vee \ldots \vee v \in V_{im_i}) \\ \beta &= \bigvee_i \gamma_i \wedge (v \in W_{i1} \vee \ldots \vee v \in W_{in_i}) \end{aligned}$$

where v does not occur in the γ_i, V_{ij}, W_{ij}, and moreover $\{\gamma_i\}$ is a partition and the V_{ij} and W_{ij} are non-empty in the context of γ_i. Apply the identity

$$V \sim W \wedge (v \in V \vee v \in W) \ = \ V \sim W \wedge v \in V \cup W$$

so that under condition γ_i both the V_{ij} and the W_{ij} are pairwise disjoint.

Fix an interval $V_{ij} = \langle\!\!| b_0, b'_0 |\!\!\rangle$. Since $[\alpha] = [\beta]$, and since the V_{ij} and the W_{ij} are pairwise disjoint, it follows that there is exactly one k such that V_{ij} is equal to the interval $W_{ik} = \langle\!\!| b_1, b'_1 |\!\!\rangle$ under condition γ_i. In other words, for this k we have

$$[\gamma_i \wedge v \in V_{ij}] = [\gamma_i \wedge v \in W_{ik}]$$

This implies $[\gamma_i \wedge b_0 = b_1 \wedge b'_0 = b'_1] = [tt]$, and so by the induction hypothesis CA $\vdash \gamma_i \wedge b_0 = b_1 \wedge b'_0 = b'_1 = tt$. Hence, $\gamma_i \wedge v \in V_{ij} = \gamma_i \wedge v \in W_{ik}$ in CA. This holds for all intervals V_{ij}, and conversely for all intervals W_{ij}, so

$$\begin{aligned} \text{CA} \vdash \quad & \textstyle\bigvee_i \gamma_i \wedge (v \in V_{i1} \vee \ldots \vee v \in V_{im_i}) \\ = \quad & \textstyle\bigvee_i \gamma_i \wedge (v \in W_{i1} \vee \ldots \vee v \in W_{in_i}) \end{aligned}$$

$\square$

References

[AD90] R. Alur and D. Dill. Automata for modeling real-time behaviour. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, LNCS 443, pages 322–335. Springer-Verlag, 1990.

[AH92] L. Aceto and M. Hennessy. Termination, deadlock and divergence. *Journal of the ACM,* 39(1):147–187, Januari 1992.

[Bae90] J.C.M. Baeten, editor. *Applications of Process Algebra.* Cambridge Tracts in Theoretical Computer Science 17. Cambridge University Press, 1990.

[Bae92] J.C.M. Baeten. Personal communication, 1992.

[BB91] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Journal of Formal Aspects of Computing Science,* 3(2):142–188, 1991.

[BB92] J.C.M. Baeten and J.A. Bergstra. Discrete time process algebra. In W.R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook *(Invited Talk),* volume 630 of *LNCS,* pages 401–420. Springer-Verlag, 1992. A full version has appeared as technical report 92/06 of the Eindhoven University of Technology.

[BB93a] J.C.M. Baeten and J.A. Bergstra. Real space process algebra. *Journal of Formal Aspects of Computing Science,* 1993.

[BB93b] J.C.M. Baeten and J.A. Bergstra. Real time process algebra with infinitesimals. Technical report P9325, University of Amsterdam, 1993.

[BBK87] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science,* 51(1/2):129–176, 1987.

[Ber92] J.A. Bergstra. Personal communication, 1992.

[BG87] J.C.M. Baeten and R.J. van Glabbeek. Merge and termination in process algebra. In K.V. Nori, editor, *Proceedings* 7^{th} *Conference on Foundations of Software Technology and Theoretical Computer Science,* Pune, India, volume 287 of *LNCS,* pages 153–172. Springer-Verlag, 1987.

[BK82] J.A. Bergstra and J.W. Klop. Fixed point semantics in process algebras. Report IW 206, Mathematisch Centrum, Amsterdam, 1982.

[BK84a] J.A. Bergstra and J.W. Klop. Fair FIFO queues satisfy an algebraic criterion for protocol correctness. Report CS-R8405, CWI, Amsterdam, 1984.

[BK84b] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[BK86] J.A. Bergstra and J.W. Klop. Verification of an alternating bit protocol by means of process algebra. In W. Bibel and K.P. Jantke, editors, *Math. Methods of Spec. and Synthesis of Software Systems '85, Math. Research 31*, pages 9–23, Berlin, 1986. Akademie-Verlag. First appeared as: Report CS-R8404, CWI, Amsterdam, 1984.

[BL91] T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In J.W. de Bakker et al., editor, *Proceedings of the REX Workshop "Real-Time :Theory in Practice"*, volume 600 of *LNCS*, pages 124–146. Springer-Verlag, 1991.

[BV93] J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. Report CSN-93/05, Eindhoven University of Technology, Eindhoven, 1993. This paper will appear in the proceedings of CONCUR '93, which will be published in the LNCS series.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Che91] L. Chen. Decidability and completeness in real-time processes. Technical Report ECS-LFCS-91-185, University of Edinburgh, 1991.

[Che92] L. Chen. An interleaving model for real time systems. In A. Nerode and M. Taitslin, editors, *Proceedings of the second International Symposium on Logical Foundation of Computer Science, Tver '92*, volume 620 of *LNCS*, pages 81–92. Springer-Verlag, 1992.

[Che93] L. Chen. *Timed Processes: Models, Axioms and Decidability.* PhD thesis, The University of Edinburgh, 1993. Also appeared as report ECS-LFCS-93-271, University of Edinburgh.

[CK90] C.C. Chang and H.J. Keisler. *Model Theory*, volume 73 of *Studies in logic and the foundations of mathematics.* North-Holland, 1990.

[DS89] J. Davis and S. Schneider. An introduction to timed csp. Techn. Monograph PRG-75, Oxford Univ. Comp. Lab., 1989.

[FK92] W.J. Fokkink and A.S. Klusener. Real time process algebra with prefixed integration. Report CS-R9219, CWI, Amsterdam, 1992. Submitted.

[GL92] J.C. Godskesen and K.G. Larsen. Real-time calculi and expansion theorems. In R. Shyamasundar, editor, *Proceedings* 12th *Conference on Foundations of Software Technology and Theoretical Computer Science,* New Delhi, India, LNCS 652, pages 302–315. Springer-Verlag, 1992.

[Gla87] R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proceedings STACS 87*, volume 247 of *LNCS*, pages 336–347. Springer-Verlag, 1987.

[GV92] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100:202–260, 1992.

[GW89] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989.

[GW91] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. Report CS-R9120, CWI, Amsterdam, 1991. An extended abstract of an earlier version has appeared in G.X. Ritter, editor, *Information Processing 89*, North-Holland, 1989.

[Han91] H.A. Hansson. *Time and probability in formal design of distributed systems.* PhD thesis, Computer Science, University of Uppsala, Sweden, 1991.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.

[Hoo91] J.J.M. Hooman. *Specification and compositional verification of real-time systems*, volume 558 of *LNCS.* Springer-Verlag, 1991.

[HR90] M. Hennessy and T. Regan. A temporal process algebra. Report 2/90, Computer Science Department, University of Sussex, 1990.

[ISO87] ISO. *Information processing systems – open systems interconnection – LOTOS – a formal description technique based on the temporal ordering of observational behaviour* ISO/TC97/SC21/N DIS8807, 1987.

[Jef91a] A. Jeffrey. Discrete timed CSP. Technical Report Memo 78, Chalmers University, Goteborg, 1991.

[Jef91b] A. Jeffrey. A linear time process algebra. In K.G. Larsen and A. Skou (eds.), editors, *Proceedings CAV '91,* Aalborg, *Denmark*, LNCS 575, pages 432–442. Springer-Verlag, 1991.

[Jef91c] A. Jeffrey. Translating timed process algebra into prioritized process algebra. Technical Report Memo 77, Chalmers University, Goteborg, 1991.

[Klu91a] A.S. Klusener. Abstraction in real time process algebra. Report CS-R9144, CWI, Amsterdam, 1991. An extended abstract appeared in J.W. de Bakker, C. Huizing, W.P. de Roever and G. Rozenberg, editors, *Proceedings of the REX workshop "Real-Time: Theory in Practice"*, LNCS 600, Springer-Verlag, 1991.

[Klu91b] A.S. Klusener. Completeness in real time process algebra. Report CS-R9106, CWI, Amsterdam, 1991. An extended abstract appeared in J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91*, Amsterdam, LNCS 527, pages 376–392. Springer-Verlag, 1991.

[Klu92] A.S. Klusener. The silent step in time. Report CS-R9221, CWI, 1992. An extended abstract appeared in W.R. Cleaveland, editor, *Proceedings of CONCUR 92*, LNCS 630, Springer-Verlag, 1992.

[Koy89] R.L.C. Koymans. *Specifying message passing and time-critical systems with temporal logic.* PhD thesis, Technical University Eindhoven, 1989.

[LV91] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing based systems. In J.W. de Bakker et al., editor, *Proceedings of the REX Workshop "Real-Time :Theory in Practice"*, volume 600 of *LNCS*, pages 397–446. Springer-Verlag, 1991.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and concurrency.* Prentice Hall International, 1989.

[MMT91] M. Merrit, F. Modugno, and M.R. Tuttle. Time-constrained automata. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *LNCS*, pages 408–423. Springer-Verlag, 1991.

[Mol89] F. Moller. *Axioms for concurrency.* PhD thesis, Report CST-59-89, Department of Computer Science, University of Edinburgh, 1989..

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *LNCS*, pages 401–415. Springer-Verlag, 1990.

[MT92] F. Moller and C. Tofts. Behavioural abstraction in TCCS. In *Proceedings ICALP 92,* Vienna, LNCS. Springer-Verlag, 1992.

[Mur91] D. Murphy. 3 papers on classical concurrency theory (IPA, nets, and event refinement). Report CSC 91/R5, University of Glasgow, Dep. of Computer Science, 1991.

[NRSV90] X. Nicollin, J.L. Richier, J. Sifakis, and J. Voiron. ATP: An algebra for timed processes. In M. Broy and C.B. Jones, editors, *Proceedings IFIP Working Conference on Programming Concepts and Methods,* Sea of Gallilea, Israel, pages 155–177. North Holland, 1990. This paper has also been published as IMAG report RT-C16.

[NS90] X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application. Technical Report RT-C26, IMAG, Laboratoire de Génie informatique, Grenoble, 1990.

[NS91] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In J.W. de Bakker et al., editor, *Proceedings of the REX Workshop "Real-Time :Theory in Practice"*, volume 600 of *LNCS*, pages 526–548. Springer-Verlag, 1991.

[NSY91] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In J.W. de Bakker et al., editor, *Proceedings of the REX Workshop "Real-Time :Theory in Practice"*, volume 600 of *LNCS*, pages 549–572. Springer-Verlag, 1991.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5th *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[QdFA93] J. Quemada, D. de Frutos, and A. Azcorra. TIC A TImed Calculus. *Journal of Formal Aspects of Computing Science*, 5(?):224–252, 1993.

[RR88] M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[Sch92] S. Schneider. An operational semantics for Timed CSP. Technical report, Oxford Univ. Comp. Lab., 1992. To appear in *Information & Computation.*

[Sto88] A. Stoughton. Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.

[Ver93a] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. Technical report CSN 93/18, Eindhoven University of Technology, Eindhoven, 1993.

[Ver93b] C. Verhoef. A general conservative extension theorem in process algebra. Draft, Eindhoven University of Technology, Eindhoven, 1993.

[Wan90] Y. Wang. Real time behaviour of asynchronous agents. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *LNCS*, pages 502–520. Springer-Verlag, 1990.

[Wan91a] Y. Wang. *A Calculus of Real Time Systems.* PhD thesis, Chalmers University of Technology, Göteborg, 1991.

[Wan91b] Y. Wang. CCS + time = an interleaving model for real time systems. In J. Leach Albert, B. Monien, and M. Rodríguez, editors, *Proceedings ICALP 91,* Madrid, volume 510 of *LNCS.* Springer-Verlag, 1991.

Samenvatting

Dit proefschrift behandelt de uitbreiding van proces algebra met tijd.

Proces algebra is de studie van parallelle processen op een algebraïsche grondslag. Het is geïnitieerd door Milner, die de proces algebra CCS ontwikkelde [Mil80],[Mil89]. Bergstra en Klop ontwikkelden vervolgens de proces algebra ACP (Algebra van Communicerende Processen) [BK84b], waarvoor wij eveneens naar het leerboek [BW90] verwijzen. Het eerste deel van het proefschrift geeft een korte inleiding in de proces algebra, gebaseerd op [BW90].

Enkele jaren geleden hebben Baeten en Bergstra ACP uitgebreid met tijd [BB91], door atomaire acties te voorzien van een tijdstip welke het tijdstip aangeeft waarop de bewuste actie geacht wordt te worden uitgevoerd. Zo stelt $a(5)$ het proces voor dat de actie a op tijdstip 5 uitvoert. Om aan te kunnen geven dat acties ook in een bepaald interval uitgevoerd kunnen worden, voerden Baeten en Bergstra het *integratie* construct in. De expressie $\int_{v \in S} p(v)$ stelt het proces voor, dat zich kan gedragen als $p(t)$ voor een willekeurig tijdstip t in S. De naam integratie ontleent het aan het feit dat het de continue versie betreft van het gegeneraliseerde som construct uit de proces algebra. In dit proefschrift beperken wij ons tot expressies van de vorm $\int_{v \in V} a(v)$ and $\int_{v \in V} (a(v) \cdot p)$, waarbij V een interval aanduidt in het onderliggende tijdsdomein. Deze beperking op de toegelaten expressies noemen wij *prefix-integratie.*

In het tweede deel van het proefschrift bestuderen wij de operationale semantiek en de axiomatizering van ACP met tijd en prefix-integratie. Het probleem van prefix-integratie is het redeneren met expressies waarin tijdsvariabelen nog vrij voor komen. Als oplossing bieden wij aan om de syntax uit te breiden tot expressies van de vorm $\int_\alpha a(v)$ en $\int_\alpha (a(v) \cdot p)$, waarbij α een boolse expressie is over tijdsvariabelen. Vervolgens geven wij een (eindige) axiomatizering waarvan wij bewijzen dat het overeenkomt met de gelijkheid van transitiesystemen modulo sterke bisimulatie. Ook geven wij een beslissingsprocedure die voor twee expressies (zonder recursie) bepaalt of zij gelijk zijn of niet. Dit deel vindt zijn oorsprong in [BB91], [Klu91b] en [FK92]; het is gedeeltelijk gezamenlijk geschreven met Willem Jan Fokkink.

Binnen de proces algebra worden er verschillende equivalenties gehanteerd die betrekking hebben op abstractie, zoals vertakkende-, wacht- en zwakke bisimulatie. In deel 3 van dit proefschrift definiëren wij deze equivalenties in ACP met tijd, en wij introduceren bijbehorende axiomas. Eerdere versies van dit werk kunnen gevonden worden in [Klu91a] en [Klu92].

Proces algebra kan gebruikt worden bij de specificatie en verificatie van parallelle systemen. In deel 4 van dit proefschrift behandelen we eerst ACP met tijd en beperkte recursie, daar wij eerst dan parallelle systemen met tijd daadwerkelijk kunnen uitdrukken. Vervolgens geven wij een specificatie en een verificatie van een protocol, waarvan een eerdere versie ook te vinden is in [Klu91a].

Een van de uitgangspunten van ACP met tijd is, dat opeenvolgende acties niet op hetzelfde tijdstip uitgevoerd kunnen worden. In deel 5 laten wij dit punt los, en bestuderen wij een variant van ACP met tijd met *urgente actie*, dat zijn acties

die achter elkaar op hetzelfde tijdstip uitgevoerd kunnen worden. Ook introduceren wij extra operatoren, waarmee een phenomeen als *maximale progressie* uitgedrukt kan worden. Tot slot geven wij in deel 5 een vertaling van enkele andere proces algebras met tijd naar ACP met tijd en urgente acties. In het bijzonder bestuderen wij de axiomas voor zwakke bisimulatie met tijd, zoals ze door verschillende anderen voorgesteld zijn.

Curriculum Vitae

- 25 juli 1965. Geboren te Nieuwveen (ZH).
- juni 1983. VWO-diploma behaald aan de Rijksscholen Gemeenschap Broklede te Breukelen. Examenpakket: Ne, Eng, Gesch, Ec, Wi1, Wi2, Nat, Schei.
- september 1983 - juni 1984. Vooropleiding Conservatorium, Utrecht (Cello).
- september 1984 - januari 1990. Informatica, Universiteit van Amsterdam.

  In de eerste jaren heb ik het studieprogramma van de afstudeerrichting *Bestuurlijke Informatica* afgerond, op de afstudeerscriptie na.

  Vervolgens ben ik overgestapt naar de *(theoretische) programmatuurkunde* en ben daar onder leiding van prof.dr. J.A. Bergstra afgestudeerd.

  Als onderdeel van mijn afstuderen heb ik enige maanden stage gelopen bij Philips Research, onder begeleiding van dr. L.M.G. Feijs en dr. H.B.M. Jonkers.
- maart 1990 - december 1993. Junior project medewerker aan het CWI (Centrum voor Wiskunde en Informatica) te Amsterdam. Mijn aanstelling is voornamelijk gefinancierd uit de volgende twee projecten.
  - maart 1990 - juni 1992. Esprit 2 Project ATMOSPHERE.

    ATMOSPHERE was een Europees onderzoeksproject op het gebied van software-ontwikkelingsomgevingen. Het CWI was binnen het project een subcontractor van Philips Research.

    Binnen dit project heb ik gewerkt aan het opstellen van een executeerbare semantiek van een deeltaal van de specificatietaal COLD.
  - juli 1992 - december 1993. RACE Project BOOST.

    BOOST is een Europees onderzoeksproject op het gebied van de telecommunicatie en intelligente netwerken.

    In dit project heb ik mij beziggehouden met het formeel specificeren van componenten van het model voor intelligente netwerken zoals het door de CCITT is vastgelegd.

  Naast het onderzoek in het kader van bovengenoemde projecten heb ik mij gedurende deze periode ook bezig gehouden met onderzoek op het gebied van de uitbreiding van de proces algebra ACP met tijd, onder begeleiding van de promotores prof.dr. J.C.M. Baeten (TUE) en prof.dr. J.A. Bergstra (UvA). Dit onderzoek heeft geleid tot het onderhavige proefschrift.

Stellingen behorende bij het proefschrift

“Models and axioms for a fragment of real time process algebra”

door A.S. Klusener

I

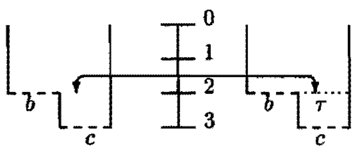
Gewortelde vertakkende bisimulatie equivalentie in ACP-met-tijd en prefix-integratie is volledig geaxiomatiseerd door de wetten voor sterke bisimulatie en de extra wet

$$\int_\alpha a(v) \cdot \left(\int_{w \in (b_0,b_1)} \tau(w) \cdot (p+q) + p\right) \;=\; \int_\alpha a(v) \cdot (p + b_0 \gg q)$$

waarbij α nog condities oplegt aan v, b_0, b_1, en het wachtgedrag van p en q; zo moet een van de procestermen p,q tot b_1 kunnen wachten terwijl de ander dat niet kan (zie de hoofdstukken 6 en 7).

II

In de stijl van Baeten en Bergstra stelt $a(t)$ het proces voor dat *tot* het tijdstip t wacht (en niet *tot en met*!) waarna het op tijdstip t de actie a uitvoert. Deze zienswijze leidt ertoe dat de procestermen $b(2)+c(3)$ en $b(2)+\tau(2)\cdot c(3)$ binnen vertakkende bisimulatie als gelijk beschouwd worden (zie hoofdstuk 6).



Binnen CCS met tijd hanteert men veelal de opvatting dat $a(t)$ *tot en met* het tijdstip t wacht waarna het a uitvoert, evenzeer op tijdstip t (zie hoofdstuk 11). Dit betekent echter dat de bovengenoemde procestermen niet meer gelijk zijn.

Zo blijkt dat een verschil in opvattingen, dat aanvankelijk nogal esoterisch lijkt, wel degelijk tot concrete verschillen kan leiden.

III

De technieken uit dit proefschrift ten aanzien van prefix-integratie zijn ook van toepassing op de analoge prefix-sommatie uit de procesalgebra met data.

Daarbij kan men overigens op de volgende manier prefix-sommatie eenvoudig uitbreiden tot algemene sommatie. We gaan ervan uit dat een atomaire actie a voor elke datum d en elke substitie σ een transitie heeft. De procesterm $\sum_{v:\alpha} p$ “erft” het gedrag van p dat voldoet aan $v : \alpha$. Enkele van de benodigde SOS regels zijn hieronder gegeven.

$$a \xrightarrow{a(d)}_\sigma \surd \qquad \frac{p \xrightarrow{a(d)}_{\sigma[d/v]} \surd \quad \models_{\sigma[d/v]} \alpha}{\sum_{v:\alpha} p \xrightarrow{a(d)}_\sigma \surd} \qquad \frac{p \xrightarrow{a(d)}_{\sigma[d/v]} p' \quad \models_{\sigma[d/v]} \alpha}{\sum_{v:\alpha} p \xrightarrow{a(d)}_\sigma p'} \qquad \frac{p \xrightarrow{a(d)}_\sigma p'}{p \cdot q \xrightarrow{a(d)}_\sigma p' \cdot \sigma(q)}$$

De axiomatisering is tamelijk eenvoudig:

$$
\begin{array}{|lcl|llcl|}
\hline
\sum_{v:\alpha} p + \sum_{v:\beta} p & = & \sum_{v:\alpha\vee\beta} p & & \sum_{v:\alpha}(\sum_{w:\beta} p) & = & \sum_{v:\alpha\wedge(\beta[v/w])} p[v/w] \\
\sum_{v:\alpha}(p+q) & = & \sum_{v:\alpha} p + \sum_{v:\alpha} q & w \notin fv(p) \cup var(\alpha) & & & \\
 & & & & \sum_{v:\alpha}(a \cdot p) & = & \sum_{v:\alpha}(a \cdot \sum_{w:\alpha} p) \\
\sum_{v:ff} p & = & \delta & v \notin fv(p) & \sum_{v:tt} p & = & p \\
\hline
\end{array}
$$

Merk op dat de conditionele procesterm $\alpha :\rightarrow p$ uitgedrukt kan worden door $\sum_{v:\alpha} p$ voor $v \notin fv(p) \cup var(\alpha)$.

Een dergelijke generalisatie van de prefix integratie in ACP-met-tijd is in principe ook mogelijk, hoewel de operationale regels voor het tijdvoortschrijdgedrag van deadlocks complicaties teweegbrengen.

IV

Stel een taal vooı met een alfabet A van atomaire acties, en een alfabet I van inverse acties. We nemen ook een constante $\delta \notin A \cup I$ aan. De taal kent als operatoren de alternatieve compositie met voorkeur, die wij noteren met $\overleftarrow{+}$ conform [BPvW93], en de algemene sequentiële compositie, genoteerd met $\cdot$. Verder is er een verzameling Γ van toestanden γ, γ', waarin expressies p, q uit deze taal worden geïnterpreteerd. Er is een partiële functie $val : A \cup \{\delta\} \times \Gamma \longrightarrow I \times \Gamma$, en een totale functie $val : I \times \Gamma \longrightarrow \Gamma$. Als $val(a,\gamma) = (i,\gamma')$ dan vereisen we dat $val(i,\gamma') = \gamma$, waarmee we uitdrukken dat zo'n actie i inderdaad de inverse actie van a in γ is. Bovendien nemen we aan dat voor elke γ de applicatie $val(\delta,\gamma)$ ongedefinieerd is.

Deze taal is alsvolgt van een semantiek te voorzien (met $a \in A \cup \{\delta\}, i \in I$):

$$
\frac{p \xrightarrow{(\gamma,\gamma')} p' \quad p' \neq \delta}{p \overleftarrow{+} q \xrightarrow{(\gamma,\gamma')} p' \overleftarrow{+} q}
\qquad
\frac{p \xrightarrow{(\gamma,\gamma')} p' \quad p' \neq \delta \quad q \overleftarrow{+} p' \cdot q \xrightarrow{(\gamma',\gamma'')} q'}{p \cdot q \xrightarrow{(\gamma,\gamma'')} q'}
\qquad
\frac{val(a,\gamma) = (i,\gamma')}{a \xrightarrow{(\gamma,\gamma')} i}
$$

$$
\frac{p \xrightarrow{(\gamma,\gamma')} \delta \quad q \xrightarrow{(\gamma',\gamma'')} q'}{p \overleftarrow{+} q \xrightarrow{(\gamma,\gamma'')} q'}
\qquad
\frac{p \xrightarrow{(\gamma,\gamma')} \delta}{p \cdot q \xrightarrow{(\gamma,\gamma')} \delta}
\qquad
\frac{val(a,\gamma) \text{ is not defined}}{a \xrightarrow{(\gamma,\gamma)} \delta}
\qquad
i \xrightarrow{(\gamma,val(i,\gamma))} \delta
$$

V

Protocold [Jon91] is voorgesteld als een deeltaal van de specificatietaal Cold [FJKR87] waaraan een operationele semantiek gegeven kan worden. Echter, de Protocold-semantiek van een expressie kan subtiel verschillen van zijn semantiek binnen Cold.

Door het uitbreiden van Protocold met extra operatoren, zoals de zogenaamde *soft-cut* en *hard-cut* uit het logisch programmeren, zou het bovengenoemd verschil zoveel mogelijk syntactisch uitgedrukt kunnen worden, wat het inzicht in de relatie van Protocold tot Cold vergroot. (zie [Klu91]).

VI

In tijd-grafen, zoals bijvoorbeeld die van Alur en Dill [AD90], heeft elke component een eigen klok. De uitgaande transities van zo'n component zijn enerzijds gelabeld met condities waarin de klokwaarde een rol kan spelen, en anderzijds met een assignment, die bijvoorbeeld de klok op nul kan zetten. Dergelijke grafen kennen een zekere redundantie, daar men ervan uitgaat dat alle klokken even hard lopen.

Het is ook mogelijk om slechts één klok te hanteren, waar de componenten in hun condities en assignments naar kunnen refereren, zonder dat zij deze klok echter op nul kunnen zetten.

Een mogelijk voordeel van dit alternatief is dat het verband tussen tijd-grafen en procesalgebra-met-tijd eenvoudiger wordt, waardoor resultaten gemakkelijker uitwisselbaar worden. Zo zou het interessant zijn te weten of Fokkink's eliminatie stelling voor reguliere processen met tijd [Fok93] over te dragen is naar het model van de tijd-grafen.

VII

Het conceptuele model voor intelligente netwerken van de CCITT [CCITT92] geeft een goed beeld van de opbouw en indeling van deze netwerken. Echter, een aantal details wekt nog vragen op, wanneer zij gezien worden in het licht van enkele basisprincipes uit de software engineering. Zo wordt er een aantal Service Onafhankelijke Componenten (in Engelse afkorting SIB's) geïntroduceerd, die onderling geen overlappende functionaliteiten geacht worden te hebben.

Een van deze componenten, de Elementaire Bel Component (in Engelse afkorting BCP), verzorgt zelf de communicatie met de gebruikers en de afrekening, terwijl deze taken toebedeeld zijn aan andere componenten, respectievelijk de Gebruiker Interactie Component en de Afrekening Component.

Een ander voorbeeld is dat de parameters van de Gebruiker Interactie Component een vorm van negatieve afhankelijkheid kennen. Indien een bepaalde parameter een bepaalde waarde heeft, dan zijn andere parameters van geen enkel belang.

Teneinde tot een specificatie te komen waarin dergelijke details helder zijn uitgewerkt, verdient het aanbeveling (gedeeltes van) het model te specificeren in een daartoe ontworpen formalisme als PSF [Mau91] of LOTOS [ISO87]. Een voorbeeld van een dergelijke studie is te vinden in [KVW93].

VIII

Bij het opstellen van de Huurwet in 1950 was het slechts voor artikel 3 van deze wet van belang zich uit te spreken over de vraag wanneer een ruimte als woon- dan wel als bedrijfsruimte beschouwd werd, indien de ruimte beide bestemmingen had; dit leidde tot lid 6 van dit artikel waarin een ruimte als woonruimte erkend wordt indien meer dan 60 % van de ruimte daadwerkelijk als woonruimte in gebruik is.

Bij de wetswijziging van 1972 is woonruimte onder het Burgelijk Wetboek komen te vallen, en derhalve is bovenstaande vraag nu in een veel groter verband aan de orde. Het wekt dan ook bevreemding dat men bij deze wetswijziging bovenstaande 60 % regel, of een aangepaste versie daarvan, niet een meer algemene zeggingskracht toebedeeld heeft.

IX

Een nieuw idee geeft pas echt voldoening wanneer men niet denkt het bedacht, maar het ontdekt denkt te hebben.

Referenties

[AD90] R. Alur and D. Dill. Automata for modeling real-time behaviour. In M. Paterson, editor, *Proceedings* 17th *ICALP*, Warwick, LNCS 443, pages 322–335. Springer-Verlag, 1990.

[BPvW93] J.A. Bergstra, A. Ponse, and J.J. van Wamel. Process algebra with backtracking. Technical report P9306, University of Amsterdam, 1993.

[CCITT92] Study Group XI (WP XI/4). *New Recommendation Q.1200- Q Series Intelligent Network Recommendation Structure.* CCITT, Geneva, 10-17 march 1992.

[FJKR87] L.M.G. Feijs, H.B.M. Jonkers, C.P.J. Koymans, and G.R Renardel de Lavelette. Formal definition of the design language COLD-K. Technical report, Philips Research Laboratories, April 1987.

[Fok93] W.J. Fokkink. An elimination theorem for regular behaviours with integration. In E. Best, editor, *Proceedings CONCUR 93,* Hildesheim, LNCS 715, pages 432–446. Springer-Verlag, 1993.

[ISO87] ISO. *Information processing systems – open systems interconnection – LOTOS – a formal description technique based on the temporal ordering of observational behaviour* ISO/TC97/SC21/N DIS8807, 1987.

[Jon91] H.B.M. Jonkers. Protocold 1.1 user manual. Technical report, Philips Research Laboratories, August 1991.

[Klu91] A.S. Klusener. An executable semantics for a subset of COLD. Report CS-R9145, CWI, Amsterdam, 1991.

[KVW93] A.S. Klusener, S.F.M. van Vlijmen and A. van Waveren. Service independent building blocks-I; concepts, examples and formal specifications. Report CS-R9326, CWI, Amsterdam, 1993.

[Mau91] S. Mauw. *PSF, A Process Specification Formalism.* PhD thesis, University of Amsterdam, Amsterdam, 1991.