Eindhoven University of Technology

Department of Mathematics and Computer Science

Master's Thesis

Finite Equational Bases for CCS with Restriction

by

P.J.A. van Tilburg

Supervisor: Dr. S.P. Luttik

Eindhoven, May 30, 2007

Preface

In 2004 I followed the course Process Algebra in my third year of the bachelor Computer Science & Engineering at the Eindhoven University of Technology. I was struck by the elegance and expressiveness of the algebras presented in the lectures and literature. When I was looking for a master project two years later, Bas Luttik suggested among other possibilities to follow up on his work with Luca Aceto et al. [2] to fill a gap in the axiomatisations of the CCS process algebra. Almost immediately I accepted the challenge to find a complete, finite axiomatisation for CCS that would add restriction. After a two month research period, I started to work on a paper on which this thesis is based.

Most of all, I want to thank my project supervisor, Bas Luttik for his extensive, quick feedback and guidance during this project. His experience with the problem without restriction helped me to understand the related literature and led me to full understanding of the current problems and possible solutions.

I also would like to thank Jos Baeten for his inspiring lectures, comprehensive and clear literature, and stimulating enthusiasm for this project. Furthermore, I'd like to thank Luca Aceto, who co-authored the paper on which most of the work in this thesis is based, for his comments on an early version of the thesis. Finally, I want to thank the members of the department of Mathematics and Computers Science and of course my friends and parents for their support in diverse ways.

Paul van Tilburg

Eindhoven, May 30, 2007

Summary

This thesis investigates the equational theory of CCS modulo bisimulation. In earlier work equational bases are given for fragments of CCS with parallelism but without restriction. Restriction is essential for the specification of parallel processes. Although it has been a part of the common process algebras, it has never been fully axiomatised. The goal of this thesis is to determine if the result of earlier work can be generalised to a theory with restriction.

First we discuss a simple fragment of CCS that does not have any form of parallelism. We establish a finite, complete axiomatisation and prove its completeness. Then a more contrived variant with parallelism is considered using the left merge and communication merge from ACP. This variant only considers interleaving of actions. We establish an extension of the previous axiomatisation for which we prove completeness and partial soundness. The third variant adds communication. For this variant we only investigate the problems involved with proving the completeness of a proposed equational base. We are not yet able to present a complete solution. We find that restriction does not distribute over parallel composition if communication is possible. We conclude that a straightforward extension of the completeness proof for the fragment of CCS without restriction is not possible.

Contents

Chapter 1

Introduction

The Calculus of Communication Systems (CCS) was developed by Robin Milner in the late seventies of the 20th century [11]. The calculus introduced a system for describing processes in a formal language using transition systems to interpret the expressions in this language. The restriction operator, already part of CCS when it was introduced, takes a process and a set of actions as arguments. It blocks the execution by the process of the actions in the set. Restriction is often used in specifying systems where it blocks the execution of interleaving actions of parallel processes so that only the result of (synchronous) communication remains. Restriction is also present in ACP [5], where it is called encapsulation.

This thesis deals with equational bases for the CCS process algebra modulo bisimulation [15]. Equational bases are sets of valid equations from which every other valid equation can be derived. These complete axiomatisations are useful starting points for proving properties of elements of the algebra, but also for systematisation of previously formulated concepts. Axiomatisations are the driving force for development of the process algebras. Early work in this area was done by Hennessy and Milner in [10, 12], where they proposed an infinite axiomatisation for CCS and proved it ground-complete, i.e. only for terms without variables. In 1984 Bergstra & Klop presented ACP [5]—the Algebra of Communication Processes, a reformulation of CCS. They added two operators, left merge and communication merge, and obtained a finite ground-complete axiomatisation with parallelism. Later, Moller proved in his PhD thesis [13] that CCS with only the merge operator has no finite ground-complete axiomatisation. He also redid the proof of Hennessy and Milner and obtained a stronger result by proving the axiomatisation of a basic fragment of CCS with left merge ω-complete. (An axiomatisation is ω-complete if any equation is derivable under all closed substitutions, then the equation itself is also derivable.) For more information about the history of process algebra see [3].

However, all the above-mentioned finite bases and completeness proofs do not cover complete, i.e. ground-complete and ω-complete, axiomatisations including communication. In [2] equational bases for more extensive fragments of CCS are proposed and proved sound and complete. Although these fragments consider parallelism, each one does not include restriction. The goal of this thesis is to determine if the equational bases from [2] can be extended and the

proofs generalised to a theory that includes restriction. Because parallelism, especially communication, in combination with restriction adds complexity to the completeness proofs, the thesis will deal with the extensions incrementally.

In [1], two methods for proving the completeness of a base are mentioned: the classic *normal form* strategy, and the *inverted substitutions* technique described by Groote [9]. Following [2] and [13] we use the normal form strategy. This entails showing that all process terms can be proved equal to some normal form using the axioms. This is followed by the construction of a distinguishing valuation that ensures that two normal forms are identical under this valuation only if they can be proved equal.

The behaviour of the restriction operator can be a disruptive element in the proofs. It can interact with the distinguishing valuations and remove actions or change the properties of crucial elements that were assumed to be present. To show more clearly how this disruption is handled in the proofs of the proposed equational bases in this thesis, we will work towards a fragment of CCS with full parallelism and restriction in three incremental steps.

These incremental steps are organised in this thesis as follows. First, we present basic theoretical elements used throughout the thesis in Chapter 2, such as terms, semantics, process algebras and equational theories. In Chapter 3 we introduce the basic fragment of CCS that is used as a general basis in this thesis and formulate the first equational base for the case without parallelism. At the end of the chapter we prove the completeness of this base. Chapter 4 adds parallelism to the algebra fragment using the left merge and communication merge from ACP, though still without communication. A new (extended) base is formulated and proved sound and complete. The same base is slightly modified in the next chapter, Chapter 5, where the addition of communication is discussed. We are not yet able to present a completeness proof for the proposed equational base. Instead, we discuss the difficulties that arise while trying to prove the completeness of the base. Finally, this thesis ends with some concluding remarks in Chapter 6.

Chapter 2

Preliminaries

This chapter introduces the basic theoretical elements on which the rest of the thesis is based. It defines everything from the ground up, so that the thesis is self-contained and requires little prior knowledge except for basic mathematical notions.

In the first section of this chapter we consider the complete CCS process algebra with all the functionality of restriction, prefixing, alternative and parallel composition. Most of the definitions are taken from [2] and [12]. In Section 2.2 and Section 2.3 we introduce a basic fragment of CCS called BCCSP+Res. Both sections are partly based on [1].

This chapter uses the algebras as an illustration for the introduced concepts. The next chapter will use the fragment BCCSP+Res on which the chapters thereafter will extend until the full feature set of CCS is regained.

2.1 Open and Closed Terms

We fix a (finite) set of *action labels* $\mathcal{L}$ and a set of *co-action labels* $\overline{\mathcal{L}}$, disjoint from $\mathcal{L}$ and in bijection with it. The bijection $\bar{\cdot}$ is given as follows: for each $a \in \mathcal{L}$ there is an $\overline{a} \in \overline{\mathcal{L}}$ and $\overline{\overline{a}} = a$. We define the set of *actions* $\mathcal{A}$ as $\mathcal{L} \cup \overline{\mathcal{L}}$. For CCS the synchronous communication action is the internal or silent action labelled with τ. We define the complete set of actions $\mathcal{A}_\tau$ as $\mathcal{A} \cup \{\tau\}$. We also fix a countably infinite set of *variables* $\mathcal{V}$. The meta-variables a, b, and c generally range over $\mathcal{A}$; x and y range over $\mathcal{V}$. The set of terms $\mathcal{T}$ is generated by the following grammar:

$$\mathrm{T} \ ::= \ \mathbf{0} \ \mid \ x \ \mid \ a.\mathrm{T} \ \mid \ \mathrm{T} + \mathrm{T} \ \mid \ \mathrm{T} \parallel \mathrm{T} \ \mid \ \mathrm{T} \setminus H$$

where $a \in \mathcal{A}_\tau$, $x \in \mathcal{V}$, and $H \subseteq \mathcal{L}$. We use the convention that the prefixing $a.$ binds the strongest, then restriction $\setminus H$, then parallel composition[1] $\parallel$, and finally alternative composition $+$.

Members of $\mathcal{T}$ are called *process terms* and the meta-variables p, q and r generally range over $\mathcal{T}$. Process terms that do not contain any variables (i.e. elements of $\mathcal{V}$) are called closed. The set of closed process terms is denoted by $\mathcal{T}^0$.

[1] In CCS parallel composition is denoted by $|$. However, since we want to use this notation for the communication merge introduced later, we will denote parallel composition with $\parallel$.

Example 2.1. Examples of closed terms are $a.\mathbf{0}$, $a.(b.\mathbf{0} + c.\mathbf{0})$, $a.b.\mathbf{0} + b.c.\mathbf{0}$, and $(a.\mathbf{0} + b.\mathbf{0}) \setminus \{a\}$. Examples of open terms are $a.(x \setminus \{a\}) + a.\mathbf{0}$ and $a.x + b.y$. Note that $a.x \setminus H \parallel p + q$ should be interpreted according to the priority convention as $(((a.x) \setminus H) \parallel p) + q$.

Proving properties of closed terms is easier than properties of open terms. Open terms contain variables which consequently have an unknown behaviour. If a property has been proved for an open term, it means that it has been proved for all possible closed instantiations of this open term. Herein obviously lies the difficulty of dealing with open terms.

Operational semantics

To give the terms some meaning in the context of process algebras, we define the semantics of the terms using a transition system.

On the set of closed terms $\mathcal{T}^0$, we define the binary relation $\xrightarrow{a}$ $(a \in \mathcal{A}_\tau)$. This relation is defined using a transition system of which the steps are given using operational semantics. Table 2.1 contains the operational rules in de Simone's format [8] with $p, q \in \mathcal{T}^0$, $a \in \mathcal{A}_\tau$, and $H \subseteq \mathcal{L}$.

$$1\ \frac{}{a.p \xrightarrow{a} p} \qquad 2\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3\ \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'} \qquad 4\ \frac{p \xrightarrow{a} p' \quad a, \overline{a} \notin H}{p \setminus H \xrightarrow{a} p' \setminus H}$$

$$5\ \frac{p \xrightarrow{a} p'}{p \parallel q \xrightarrow{a} p' \parallel q} \qquad 6\ \frac{q \xrightarrow{a} q'}{p \parallel q \xrightarrow{a} p \parallel q'} \qquad 7\ \frac{p \xrightarrow{a} p' \quad q \xrightarrow{\overline{a}} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

Table 2.1: The operational semantics of CCS

If a process term p' exists such that $p \xrightarrow{a} p'$, then we call p' the *residual* of p. When for a term p no such p' exists that $p \xrightarrow{a} p'$, we write $p \overset{a}{\not\longrightarrow}$.

Parallelism

The operational semantic rules 5–7 deal with parallelism. Rule 5 and 6 model interleaving of actions. If a process p can perform an action, it may do so and the resulting process will be in parallel with q again. The same holds for process q with respect to p. Rule 7 models the so-called CCS-style communication. If p can perform some action a and q can perform the corresponding co-action $\overline{a}$, then a communication can happen resulting in the silent action τ being performed and the resulting processes being in parallel again.

Example 2.2. Consider the following sender-receiver example where we model a process of two units working together to receive and then send a certain data item.

For an element d of some finite data set D we have the receiver process $r = \sum_{d \in D} recv(d).comm(d).\mathbf{0}$ and the sender process $s = \sum_{d \in D} \overline{comm(d)}.send(d).\mathbf{0}$. If we put the sender and receiver in parallel, $r \parallel s$, the process can perform the following action sequence:

$$r \parallel s \xrightarrow{recv(d)} comm(d).\mathbf{0} \parallel s \xrightarrow{\tau} \mathbf{0} \parallel send(d).\mathbf{0} \xrightarrow{send(d)} \mathbf{0} \parallel \mathbf{0}.$$

But also other actions are possible, such as $comm(d).\mathbf{0} \parallel s \xrightarrow{\overline{comm(d)}} comm(d).\mathbf{0} \parallel send(d).\mathbf{0}$ or $r \parallel s \xrightarrow{\overline{comm(d)}} r \parallel send(d).\mathbf{0}$, etc. The model can perform send, receive and communication actions in a seemingly arbitrary order and it is obvious that this is not what we wanted as a model for a clean sender-receiver. So, we need more.

Restriction

The operational semantics rule 4 reveals the functionality of the restriction operator $\backslash H$ for some $H \subseteq \mathcal{L}$: if a process p can perform some given action, then p under restriction of the action labels H can only perform this action if the action or its co-action is not a member of H.

So, this operator is useful in modelling when one wants to prevent a certain action to happen. This is especially the case for enforcing synchronous communication of two processes when it is also possible for them to each perform an action separately. These separate actions are put under restriction and only the communication can happen.

Example 2.3. We modify our sender-receiver model from Example 2.2 by putting the communication actions under restriction as follows: $(r \parallel q) \setminus H$ with $H = \{comm(d) \mid d \in D\}$. Then only the following action sequence is allowed:

$$(r \parallel s)\backslash H \xrightarrow{recv(d)} (comm(d).\mathbf{0} \parallel s)\backslash H \xrightarrow{\tau} (\mathbf{0} \parallel send(d).\mathbf{0})\backslash H \xrightarrow{send(d)} (\mathbf{0} \parallel \mathbf{0})\backslash H.$$

Using restriction communication was enforced and our sender-receiver process model behaved as expected. See also Chapter 5 for more about communication and restriction.

Process term properties

To be able to use induction on process terms, we introduce the notion of length. For other induction purposes we also introduce the notion of number of symbols of a process term. Besides for induction purposes, the length shall be used in the following chapters to distinguish different kinds of term structures.

Definition 2.4. For the process term $p \in \mathcal{T}$ we define the *length* $|p|$ using the operational semantics (see Table 2.1) as the maximum number of actions it can perform:

$$|p| = \max\{n \mid \exists_{p_1,\ldots,p_n \in \mathcal{T}} \text{ s.t. } p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} p_n\}.$$

Note. The operational semantics may be used for open terms, in which case process variables will not be able to perform any actions and thus have a length of 0.

For a process term $p \in \mathcal{T}$ we define the *number of symbols* $(\!|p|\!)$ using the structure of p as follows:

$$\begin{array}{ll} (\!|\mathbf{0}|\!) = 1, & (\!|q + r|\!) = 1 + (\!|q|\!) + (\!|r|\!), \\ (\!|x|\!) = 1, & (\!|q \setminus H|\!) = 1 + (\!|q|\!), \\ (\!|a.q|\!) = 1 + (\!|q|\!), & (\!|p \parallel q|\!) = 1 + (\!|q|\!) + (\!|r|\!). \end{array}$$

2.2 Process Algebras

An *algebra* is a structure that consists of a set of elements closed under one or more operations, often combined with a set of laws, called axioms, that allow the elements to be analysed or manipulated. The operations of the algebraic structure are given by a *signature*. This is a set of operations combined with a function that assigns the arity to each operation symbol.

In this section we define a process algebra, BCCSP+Res[2], which is a fragment of CCS discussed in the previous section that discards any form of parallelism. Process algebras are geared towards describing processes and systems, using the axioms to manipulate these descriptions with equational reasoning (see Section 2.3).

For **P** we shall use a subset of the terms $\mathcal{T}$ called $\mathcal{P}$ that does not include the parallel composition operator. The terms of $\mathcal{P}$ are generated by the following grammar:

$$\mathrm{P} \ ::= \ \mathbf{0} \ \mid \ x \ \mid \ a.\mathrm{P} \ \mid \ \mathrm{P} + \mathrm{P} \ \mid \ \mathrm{P} \setminus H$$

where $a \in \mathcal{A}$, $x \in \mathcal{V}$, and $H \subseteq \mathcal{A}$. Note that for **P** we do not consider the internal action τ, because it is irrelevant until communication is added. Also, the restriction set H is a subset of $\mathcal{A}$ whereas it is $\mathcal{L}$ for CCS. All previously given definitions for $\mathcal{T}$ also hold for $\mathcal{P}$.

Now, we shall introduce an equivalence relation on the closed terms $\mathcal{P}^0$ to obtain the universe of the process algebra. Then we shall define the signature.

Bisimulation

The operational semantics give the behaviour of the closed terms. To be able to show that two terms have equal behaviour we introduce the notion of bisimulation on the closed terms (similarly to the definition given in [15]).

Definition 2.5. A *bisimulation* is a symmetric binary relation $\mathcal{R}$ on $\mathcal{P}^0$ such that $p \,\mathcal{R}\, q$ implies

$$\text{if } p \xrightarrow{a} p', \text{ then a } q' \in \mathcal{P}^0 \text{ exists such that } q \xrightarrow{a} q' \text{ and } p' \,\mathcal{R}\, q'.$$

Closed process terms $p, q \in \mathcal{P}^0$ are said to be *bisimilar* (notation: $p \leftrightarrows q$) if a bisimulation relation $\mathcal{R}$ exists such that $p \,\mathcal{R}\, q$.

The relation $\leftrightarrows$ is an equivalence relation and thus partitions $\mathcal{P}^0$ into equivalence classes. For $p \in \mathcal{P}^0$ we define its *equivalence class*, denoted by $[p]$, as:

$$[p] = \{q \in \mathcal{P}^0 : p \leftrightarrows q\}.$$

Example 2.6. Bisimulation allows us make the distinction between the equivalence classes [a.(b.**0**+ c.**0**)] and [a.b.**0**+ a.c.**0**], because $a.(b.\mathbf{0}+c.\mathbf{0}) \xrightarrow{a} b.\mathbf{0}+c.\mathbf{0}$ but $a.b.\mathbf{0} + a.c.\mathbf{0} \xrightarrow{a} b.\mathbf{0}$ and $a.b.\mathbf{0} + a.c.\mathbf{0} \xrightarrow{a} c.\mathbf{0}$. Hence, $a.(b.\mathbf{0} + c.\mathbf{0}) \not\leftrightarrows a.b.\mathbf{0} + a.c.\mathbf{0}$.

[2]For simplicity the basic algebra of this thesis is denoted by **P**. Future extensions of the algebra shall add a letter in subscript to indicate which extension is meant.

Definition 2.7. The algebra $\mathbf{P}$ is based on the equivalence classes obtained by dividing $\leftrightarrows$ on $\mathcal{P}^0$ (notation: $\mathcal{P}^0/\leftrightarrows$) and has the constant $\mathbf{0}$, the unary operators $a.$ (for all $a \in \mathcal{A}$), the unary operators $\backslash H$ (for all $H \subseteq \mathcal{L}$), and the binary operator $+$. These operators are defined as follows:

$$\begin{array}{ll} \mathbf{0} = [\mathbf{0}], & [p] \setminus H = [p \setminus H], \\ a.[p] = [a.p], & [p] + [q] = [p + q]. \end{array}$$

Members of $\mathbf{P}$ are called *processes* and will be ranged over by p, q and r like process terms. This convention does not give rise to any confusion because it will be clear from the context which set is meant.

The binary relations $\xrightarrow{a}$ ($a \in \mathcal{A}$) defined earlier for $\mathcal{P}^0$ also induce binary relations on $\mathbf{P}$. We denote these as $\xrightarrow{a}$ too. For all $p, p' \in \mathcal{P}^0$ we define that $[p] \xrightarrow{a} [p']$ iff for all $q \in [p]$ there exists a $q' \in [p']$ that $q \xrightarrow{a} q'$.
This definition allows us to lift the operational semantics of $\mathcal{P}$ (rules 1–4 from Table 2.1) to the process algebra level.

Proposition 2.8. *For all $p, q, r \in \mathbf{P}$ and $a, b \in \mathcal{A}$*

1. $p \not\xrightarrow{a}$ *iff no $p' \in \mathbf{P}$ exists such that $p \xrightarrow{a} p'$;*
2. $a.p \xrightarrow{b} r$ *iff $a = b$ and $p = r$;*
3. $p + q \xrightarrow{a} r$ *iff $p \xrightarrow{a} r$ or $q \xrightarrow{a} r$;*
4. $p \setminus H \xrightarrow{a} q \setminus H$ *iff $p \xrightarrow{a} q$ and $a, \overline{a} \notin H$.*

Because bisimulation preserves the notion of length, all process members of an equivalence classes have the same length and therefore Definition 2.4 also holds for any process $p \in \mathbf{P}$. Besides the length, we also need the notion of branching degree to distinguish certain kinds of term structures.

Definition 2.9. Based on the operational semantics, we define the for $p \in \mathbf{P}$ the *branching degree* $\langle p \rangle$ as:

$$\langle p \rangle = |\{(a, p') \mid p \xrightarrow{a} p'\}|.$$

Note. Because the branching degree definition operates modulo bisimulation the branching degree of both the processes $a.\mathbf{0} + a.(\mathbf{0} + \mathbf{0})$ and $a.\mathbf{0}$ is 1.

2.3 Equational Theory

Now that we have an equivalence based on classes of processes, we can move on to the notion of equation and derivability. However, we need more definitions first.

Definition 2.10. A valuation is a mapping $\nu : \mathcal{V} \to \mathbf{P}$. Such a mapping may be applied on process terms of $\mathcal{P}$ using an *evaluation mapping* $[\![\cdot]\!]_\nu : \mathcal{P} \to \mathbf{P}$. We can define the mapping inductively by:

$$\begin{array}{ll} [\![\mathbf{0}]\!]_\nu = \mathbf{0}, & [\![q + r]\!]_\nu = [\![q]\!]_\nu + [\![r]\!]_\nu, \\ [\![x]\!]_\nu = \nu(x), & [\![q \setminus H]\!]_\nu = [\![q]\!]_\nu \setminus H, \\ [\![a.q]\!]_\nu = a.[\![q]\!]_\nu\,. & \end{array}$$

Note. The evaluation mappings map process terms to members of the algebra $\mathbf{P}$, which are equivalence classes of closed terms. When an evaluation mapping is applied to a closed process term, it can be omitted, e.g. $[\![a.\mathbf{0} + b.\mathbf{0}]\!]_\nu$ can just as well be written as $a.\mathbf{0} + b.\mathbf{0}$.

Note. An equivalence class can be described by one of its closed term members. Therefore, from here on we will make no distinction between a closed term and its equivalence class.

Process equations

To be able to reason syntactically about equality in $\mathbf{P}$ it is convenient to have a set of process equations. We write a *process equation* as a pair of process terms: $p \approx q$. The equation $p \approx q$ is called valid if $[\![p]\!]_\nu = [\![q]\!]_\nu$ for all valuations $\nu : \mathcal{V} \to \mathbf{P}$. The notion of a process equation $p \approx q$ being valid will later be denoted by $p \leftrightarrows q$. This effectively extends the usage of the operator $\leftrightarrows$ to also encompass open process terms besides the closed process terms as defined in Definition 2.10.

Equational base

Axioms are process equations chosen to function as the most basic equations to reason from. Table 2.2 contains such a set of axioms.

Note that the axioms A1–A4 are considered to be the most basic axioms, shared between all process algebras. The axioms in Table 2.2 are not the complete equational base that we shall use for $\mathbf{P}$. Refer to the next chapter for the complete base.

(A1)	$x + y$	$\approx y + x$
(A2)	$(x + y) + z$	$\approx x + (y + z)$
(A3)	$x + x$	$\approx x$
(A4)	$x + \mathbf{0}$	$\approx x$

Table 2.2: Basic subset of the axioms of $\mathbf{P}$

An *equational base* of an algebra is a set of valid equations from which all other valid equations can be derived. Put differently, an equational base is a complete axiomatisation. Valid process equations can be *derived* from each other using the common inference rules of equational logic [6] and axioms. Table 2.3 contains these common inference rules for $p, q \in \mathbf{P}$, $a \in \mathcal{A}$, $H \subseteq \mathcal{A}$, and $\sigma : \mathcal{V} \to \mathcal{P}$. Here, we silently assume that $x[\sigma] = \sigma(x)$ and that $.[\sigma]$ distributes over all the operators in the term to which it is applied. We call E1–E3 the equivalence rules, CG1-CG3 the congruence rules, and S1 the substitution rule.

$$\text{(E1)}\frac{}{p \approx p} \qquad \text{(E2)}\frac{p \approx q}{q \approx p} \qquad \text{(E3)}\frac{p \approx q \quad q \approx r}{p \approx r}$$

$$\text{(CG1)}\frac{p \approx q}{a.p \approx a.q} \qquad \text{(CG2)}\frac{p \approx p' \quad q \approx q'}{p + q \approx p' + q'} \qquad \text{(CG3)}\frac{p \approx q}{p \setminus H \approx q \setminus H}$$

$$\text{(S1)}\frac{p \approx q}{p[\sigma] \approx q[\sigma]}$$

Table 2.3: Common inference rules of equational logic

When a base contains a finite number of axioms, the base is called *finite*. All bases presented in this paper are finite as long as $\mathcal{L}$ is finite. The complete set of all valid process equations derivable from the base is called the *equational theory*.

We can show that a set of axioms is an equational base by proving the soundness and completeness of the axioms.

Soundness

To verify that the axioms do not allow us to make a derivation that does not comply with the behaviour of the transition system, we have to prove that the equational base is sound. Soundness means that for every $p, q \in \mathbf{P}$ we have that if an equation $p \approx q$ is derivable, then that $p \leftrightarroweq q$. To show soundness it suffices to show soundness for each of the axioms of the algebra. For the axioms that are well-known, we shall not prove their soundness in this thesis.

Completeness

An equational base is a complete axiomatisation. So, the base also has to make sure that if two processes have equal behaviour, i.e. they are bisimilar, then this has to be syntactically derivable. More formally, completeness means that for every $p, q \in \mathbf{P}$, $p \approx q$ is implied by $p \leftrightarroweq q$. As mentioned before in the introduction, there are several strategies that we can follow for proving completeness. We shall use the following strategy:

1. Prove that all terms of $\mathcal{P}$ are rewritable to certain normal forms using the axioms.
2. Prove that if normal forms s, t are not bisimilar, then $[\![s]\!]_* \neq [\![t]\!]_*$ holds for some special valuation $*$.

Chapter 3

Basic Equational Base with Restriction

This chapter discusses the base of the basic process algebra $\mathbf{P}$, introduced in the previous chapter (Section 2.2). In this chapter we aim to explore the effect that restriction has on normal forms, soundness, and completeness proofs without considering any form of parallelism. Because we do not consider parallelism, in particular no communication, we also will not consider the existence of the internal action τ.

The basic operational rules for the set of closed terms $\mathcal{P}$ are given in Table 3.1 with $p, q \in \mathcal{P}^0$, $a \in \mathcal{A}$, and $H \subseteq \mathcal{A}$.

$$1\frac{}{a.p \xrightarrow{a} p} \qquad 2\frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3\frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'} \qquad 4\frac{p \xrightarrow{a} p' \quad a \notin H}{p \setminus H \xrightarrow{a} p' \setminus H}$$

Table 3.1: The operational semantics with restriction

Table 3.2 contains a proposed equational base for $\mathbf{P}$. It has the basic axioms A1–A4, but also additional axioms that deal with restriction.

To remove multiple occurrences of the same variable, it would be useful to have the following axiom:

$$\text{(DX4)} \quad x \setminus H + x \setminus J \approx x \setminus (H \cap J).$$

However, it is not sound, as the following example shows.

Example 3.1. Instantiate x with $a.b.\mathbf{0}$, and choose $H = \{a\}$, $J = \{b\}$. Then, $x \setminus H + x \setminus J = a.b.\mathbf{0} \setminus \{a\} + a.b.\mathbf{0} \setminus \{b\} \approx a.\mathbf{0}$. However, $x \setminus (H \cap J) = a.b.\mathbf{0} \setminus (\{a\} \cap \{b\}) \approx a.b.\mathbf{0}$. Obviously, $a.\mathbf{0} \not\approx a.b.\mathbf{0}$.

3.1 Normal Forms

If it can be shown that all terms $p \in \mathcal{P}$ have a normal form, we can start the completeness proof assuming normal forms only. The normal forms are defined

(A1)	$x + y$	$\approx y + x$	
(A2)	$(x + y) + z$	$\approx x + (y + z)$	
(A3)	$x + x$	$\approx x$	
(A4)	$x + \mathbf{0}$	$\approx x$	
(D1)	$\mathbf{0} \setminus H$	$\approx \mathbf{0}$	
(D2)	$a.x \setminus H$	$\approx \mathbf{0}$	if $a \in H$
(D3)	$a.x \setminus H$	$\approx a.(x \setminus H)$	otherwise
(D4)	$(x + y) \setminus H$	$\approx x \setminus H + y \setminus H$	
(DX1)	$x \setminus \emptyset$	$\approx x$	
(DX2)	$x \setminus \mathcal{A}$	$\approx \mathbf{0}$	
(DX3)	$(x \setminus H) \setminus J$	$\approx x \setminus (H \cup J)$	

Table 3.2: The axioms of **P**

as the set $\mathcal{P}^{\mathrm{nf}}$. Its elements are generated by the following grammar:

$$\mathrm{N} ::= \mathbf{0} \mid a.\mathrm{N} \mid \mathrm{N} + \mathrm{N} \mid x \setminus H$$

where $a \in \mathcal{A}, x \in \mathcal{V}$, and $H \subset \mathcal{A}$. We designate the normal forms $a.N$ and $x \setminus H$ as *simple normal forms*. If these simple normal forms occur in a summation, we also call them *summands*.

For the normal form summands $x \setminus H$, we shall not consider the case where $H = \mathcal{A}$ since $x \setminus \mathcal{A} \approx \mathbf{0}$ (by DX2). We *do* allow $H = \emptyset$ for conveniently writing a variable x as a term with an empty restriction $x \setminus \emptyset$ (allowed by DX1).

Note. The normal forms can be considered as the result of repeatedly applying the axioms D1–D4 to the process terms. This results in only variables being restricted. Therefore, normal forms are considered to be unique modulo A1, A2 and A4, i.e. except for duplicate summands.

To be able to prove that every process term has a certain normal form we use the following lemma:

Lemma 3.2. *For every normal form $s \in \mathcal{P}^{\mathrm{nf}}$ and $H \subseteq \mathcal{A}$ a normal form $s' \in \mathcal{P}^{\mathrm{nf}}$ exists such that $s' \approx s \setminus H$.*

Proof. This can be shown by induction on the structure of s.

1. If $s = \mathbf{0}$, then $s \setminus H = \mathbf{0} \setminus H \approx \mathbf{0}$ by D1, and $\mathbf{0} \in \mathcal{P}^{\mathrm{nf}}$.

2. If $s = a.t$, then we have two sub-cases. If $a \in H$, then $s \setminus H = a.t \setminus H \approx \mathbf{0}$ by D2 and $\mathbf{0} \in \mathcal{P}^{\mathrm{nf}}$. If $a \notin H$, then $s \setminus H \approx a.t \setminus H \approx a.(t \setminus H)$ by D3. Knowing that $t \in \mathcal{P}^{\mathrm{nf}}$ because of the structural definition of normal forms, we also know a $t' \in \mathcal{P}^{\mathrm{nf}}$ exists such that $t' \approx t \setminus H$ by induction hypothesis, so $s \setminus H \approx a.t'$ and $a.t' \in \mathcal{P}^{\mathrm{nf}}$.

3. If $s = t + u$, with $t, u \in \mathcal{P}^{\mathrm{nf}}$, then we have by the induction hypothesis that $t', u' \in \mathcal{P}^{\mathrm{nf}}$ such that $t' \approx t \setminus H$ and $u' \approx u \setminus H$. This means that $s \setminus H \approx (t \setminus H) + (u \setminus H) \approx t' + u'$ by D4 and $(t' + u') \in \mathcal{P}^{\mathrm{nf}}$.

4. If s is a variable under restriction, say $s = x \setminus J$, then $s \setminus H = (x \setminus J) \setminus H \approx x \setminus (H \cup J)$ by DX3 and $x \setminus (H \cup J) \in \mathcal{P}^{\mathrm{nf}}$. □

Lemma 3.3. *Every process term $p \in \mathcal{P}$ has a normal form $s \in \mathcal{P}^{\mathrm{nf}}$ such that $p \approx s$.*

Proof. This can be shown by induction on the structure of p.

1. If $p = \mathbf{0}$, then p is a normal form since $\mathbf{0} \in \mathcal{P}^{\mathrm{nf}}$.
2. If p is a variable, say $p = x$, then $p \approx x \setminus \emptyset$ by D1 and $(x \setminus \emptyset) \in \mathcal{P}^{\mathrm{nf}}$.
3. If $p = a.q$, we know by induction hypothesis that q has a normal form $t \in \mathcal{P}^{\mathrm{nf}}$ such that $q \approx t$. This means that $p \approx a.t$ and $a.t \in \mathcal{P}^{\mathrm{nf}}$.
4. If $p = q + r$, we know by induction hypothesis that q and r have normal forms t and u such that $q \approx t$ and $r \approx u$. So, $p \approx t + u$ and $(t + u) \in \mathcal{P}^{\mathrm{nf}}$.
5. If $p = q \setminus H$, we know that a $t \in \mathcal{P}^{\mathrm{nf}}$ exists such that $q \approx t$. Lemma 3.2 gives us that if $t \in \mathcal{P}^{\mathrm{nf}}$ then a $s \in \mathcal{P}^{\mathrm{nf}}$ exists such that $s \approx t \setminus H$. Then $p \approx t \setminus H \approx s$ and $s \in \mathcal{P}^{\mathrm{nf}}$. □

Lemma 3.4. *If $s \in \mathcal{P}^{\mathrm{nf}}$, then it can be written using the following general form (modulo A1, A2 and A4):*

$$s = \sum_{i \in I} a_i.s_i + \sum_{j \in K} x_j \setminus H_j \quad \text{and } s_i \in \mathcal{P}^{\mathrm{nf}}$$

with $a_i \in \mathcal{A}$, $x_j \in \mathcal{V}$ and $H_j \subset \mathcal{A}$.

Proof. The associativity and commutativity of $+$ (by A1 and A2) allows us to use $\sum$. Using the agreement that for an empty index set we write $\sum_{i \in \emptyset} s = \mathbf{0}$ gives us a notation for $\mathbf{0}$ that fits the format. Finally, $\mathbf{0}$-summands can be removed by A4. □

Because of Lemma 3.3, it suffices to prove completeness by showing that $s \leftrightarrows t \Rightarrow s \approx t$ for all normal forms s and t.

3.2 Soundness and Completeness

Since the axioms A1–A4 and D1–D4 are well-known (see [7] or [10]), we shall not prove soundness of the axioms here.

Proposition 3.5 (Soundness)**.** *For all process terms $p, q \in \mathbf{P}$, if $p \approx q$, then $p \leftrightarrows q$.*

Note. The soundness of the extra axioms DX1–DX3 (also known from [12]) can be understood intuitively by looking at operational semantic rule 4 from Table 3.1. Concerning DX1, all possible actions that p may perform, can still be performed. For DX2 no actions can be performed, and concerning DX3 only the actions can be performed that are both not in H and not in J.

To be able to prove completeness, we first need a valuation that is able to distinguish action prefixes from variables. We also have to make sure that after restriction is applied, this distinction is still detectable.

Definition 3.6. We define the valuation $*_m$ by assigning a uniquely identifiable process for each variable $x \in \mathcal{V}$:

$$*_m(x) = \sum_{a \in \mathcal{A}} a.\psi_{\lceil x \rceil \cdot m} \text{ with } \psi_i = \sum_{a \in \mathcal{A}} a^i.\mathbf{0}$$

with $m \geq 1$ and some injective function $\lceil \cdot \rceil : \mathcal{V} \to (\mathbb{N} - \{0\})$.

Initially, $*_m(x)$ was defined as $\psi_{\lceil x \rceil \cdot m+1}$. However, this resulted in the normal forms x and $x \setminus \{a\} + x \setminus \{b\}$ being indistinguishable, because $(a^j.\mathbf{0} + b^j.\mathbf{0}) \setminus \{a\} + (a^j.\mathbf{0} + b^j.\mathbf{0}) \setminus \{b\} \approx a^j.\mathbf{0} + b^j.\mathbf{0}$ by A3. Defining $*_m$ as a summation of all $a \in \mathcal{A}$ prefixing $\psi_{\lceil x \rceil \cdot m}$ solved this issue. Because $H \neq \mathcal{A}$, it is always possible to perform an action and the residual will have the following distinguishing properties:

If m is chosen high enough, i.e. equal or greater than the maximum prefix length $|\cdot|$ that can be found in the process term, the difference between an action prefix term and a variable can be detected. This valuation makes it also possible to detect which variable was substituted because, using the injective function $\lceil \cdot \rceil$, the valuation assigns different multiples of prefix length m for each separate variable. It is also possible to detect which restriction was applied by looking at the subset of actions that the residual of a variable can take.

When the valuation $*_m$ is applied to a simple normal form such as $x \setminus H$, the residual of the process after performing any action is always *unique*, namely $\psi_{\lceil x \rceil \cdot m} \setminus H$.

Example 3.7. Consider a set of action labels $\mathcal{L} = \{a, b\}$, a set of variables $\mathcal{V} = \{x\}$ and a process term: $t = a.a.\mathbf{0} + x \setminus \{b\}$. We choose $m = |t| = 2$ and assume $\lceil x \rceil = 1$. Then $*_m$ is defined as:

$$*_m(x) = a.\psi_2 + b.\psi_2 = a.(a.a.\mathbf{0} + b.b.\mathbf{0}) + b.(a.a.\mathbf{0} + b.b.\mathbf{0}).$$

If we examine $[\![t]\!]_{*_m} = a.a.\mathbf{0} + a.a.a.\mathbf{0}$, then we still can see which summand was the closed term and which summand was the variable considering the length of both summands.

Given the same set of actions, set of variables and value of m, we can also show that the valuation $*_m$ can show the difference between the term $x \setminus \{a\} + x \setminus \{b\}$ and $x \setminus \emptyset$:

$$[\![x \setminus \{a\} + x \setminus \{b\}]\!]_{*_m} = b.b.b.\mathbf{0} + a.a.a.\mathbf{0},$$

$$[\![x \setminus \emptyset]\!]_{*_m} = a.(a.a.\mathbf{0} + b.b.\mathbf{0}) + b.(a.a.\mathbf{0} + b.b.\mathbf{0}).$$

Note that the difference of behaviour of the two processes is visible after performing the first action. The latter process offers a choice after the first action whereas the first process doesn't.

Lemma 3.8. *Let $s, s' \in \mathcal{P}^{\mathrm{nf}}$ be simple normal forms, $x \in \mathcal{V}$, $H \subset \mathcal{A}$ and let $m \geq |s|$. If $s = x \setminus H$, then the unique residual of $[\![s]\!]_{*_m}$ has a length larger than 0 and divisible by m, whereas if $s = a.s'$, then the unique residual of $[\![s]\!]_{*_m}$ has a length 0 or a length not divisible by m.*

Proof. Assume that p is the residual of s: $[\![s]\!]_{*_m} \xrightarrow{a} p$ for some $a \in \mathcal{A}$. Then by case analysis:

1. If $s = x \setminus H$, then according to the definition of $*_m$ (see Definition 3.6) $p = \psi_{\lceil x \rceil \cdot m}$, so $|p| = \lceil x \rceil \cdot m$ and hence $|p|$ is divisible by m.
2. If $s = a.s'$, then $p = [\![s']\!]_{*_m}$. We can distinguish the following two sub-cases: either s' does not contain a process variable, then $0 \leq |p| < m$, or s' does contain a process variable. In the latter sub-case we consider for the length of p specifically the variable $x \in \mathcal{V}$ in p for which $\lceil x \rceil$ yields the largest value. Then, by the definition of $*_m$ (see Definition 3.6) the length $|p| \geq \lceil x \rceil \cdot m + 1$, so $|p| > \lceil x \rceil \cdot m$ and also $|p| \leq \lceil x \rceil \cdot m + (|s| - 1) \leq \lceil x \rceil \cdot m + (m-1) < (\lceil x \rceil + 1) \cdot m$. So, in both cases $|p|$ is not divisible by m. □

Theorem 3.9 (Completeness)**.** *For every two normal forms $s, t \in \mathcal{P}^{\mathrm{nf}}$ with $m > |s|$ and $m > |t|$, it holds that if $[\![s]\!]_{*_m} = [\![t]\!]_{*_m}$, then $s \approx t$ modulo A1–A4.*

Proof. Using Lemma 3.4, we know that the normal forms s and t are summations of terms of the form $a.s'$ or $x \setminus H$. We now prove that $s \approx t$ assuming that $[\![s]\!]_{*_m} = [\![t]\!]_{*_m}$ by induction on the sum of the lengths of s and t. We do this by proving that for every summand s_i of s a summand t_j of t exists such that $s_i \approx t_j$ modulo A1–A4. Consider the following case analysis based on the syntax of an arbitrary summand s_i of s:

1. If $s_i = a.s_i'$, then $[\![s_i]\!]_{*_m} \xrightarrow{a} [\![s_i']\!]_{*_m}$. Because $[\![s]\!]_{*_m} = [\![t]\!]_{*_m}$ there must also be a t_j in t such that $[\![t_j]\!]_{*_m} \xrightarrow{a} [\![s_i']\!]_{*_m}$.

   It cannot be that t_j has the simple normal form $x \setminus H$ (with $x \in \mathcal{V}$ and $a \notin H$), since by Lemma 3.8 we know that in this case the length residual will differ from the length from $[\![s_i']\!]_{*_m}$. This means that t_j must have the form $b.t_j'$.

   Given that t_j has this form, it can perform only one action, namely $[\![t_j]\!]_{*_m} \xrightarrow{b} [\![t_j']\!]_{*_m}$. Hence, from $[\![t_j]\!]_{*_m} \xrightarrow{a} [\![s_i']\!]_{*_m}$ it follows that $a = b$ and $[\![s_i']\!]_{*_m} = [\![t_j']\!]_{*_m}$. By induction hypothesis we know that $s_i' \approx t_j'$ modulo A1–A4. Hence, we may conclude that $s_i = a.s_i' \approx b.t_j' = t_j$.

2. If $s_i = x \setminus H$, then, since $H \subset \mathcal{A}$, $[\![s_i]\!]_{*_m} \xrightarrow{a} p$ for some $a \in \mathcal{A} - H$. By a similar reasoning as in the previous action we know that there also is a t_j in t such that $[\![t_j]\!]_{*_m} \xrightarrow{a} p$. By Lemma 3.8 we know that t_j must have the form $y \setminus J$ for some $y \in \mathcal{V}$ and $J \subset \mathcal{A}$. We proceed to show that $x = y$ and $H = J$.

   Because of the definition of $*_m$ (see Definition 3.6), we know that $p = \psi_{\lceil x \rceil \cdot m} \setminus H \approx \sum_{a \in \mathcal{A} - H} a^{\lceil x \rceil \cdot m}.\mathbf{0}$ by D1–D4 and the residual of $y \setminus J$ is $\psi_{\lceil y \rceil \cdot m} \setminus H$. From this observation it follows by soundness that

   $$\sum_{a \in \mathcal{A} - H} a^{\lceil x \rceil \cdot m}.\mathbf{0} = \psi_{\lceil x \rceil \cdot m} \setminus H = p = \psi_{\lceil y \rceil \cdot m} \setminus J = \sum_{a \in \mathcal{A} - J} a^{\lceil y \rceil \cdot m}.\mathbf{0} \ .$$

   Hence, $\lceil x \rceil \cdot m = |p| = \lceil y \rceil \cdot m$, so $x = y$ since $\lceil \cdot \rceil$ is an injection and $H = J$.

   Now it has been established that $x = y$ and $H = J$; we may conclude that $s_i = x \setminus H = y \setminus J = t_j$.

The case analysis shows that for every summand s_i of s a summand t_j of t exists such that $s_i \approx t_j$ modulo A1–A4. It follows by a symmetric argument that every summand of t is also provably equal to a summand of s. Hence $s \approx s + t \approx t$ modulo A1–A4. □

Corollary 3.10. *For all process terms $p, q \in \mathbf{P}$ it holds that $p \approx q$ if and only if $p \leftrightarrows q$.*

Proof. The implication from the left to the right follows from Proposition 3.5.

For the proof for the implication from the right to the left, we assume that $p \leftrightarrows q$. For p and q two normal forms s and t exist respectively such that $p \approx s$ and $q \approx t$ by Lemma 3.3. If $p \leftrightarrows q$ then by Proposition 3.5 we also know that $s \leftrightarrows t$ and thus $[\![s]\!]_{*_m} = [\![t]\!]_{*_m}$. Hence, by Theorem 3.9 we know that $s \approx t$ and we can conclude that $p \approx s \approx t \approx q$. □

Chapter 4

Equational Base with Interleaving and Restriction

In this chapter we extend the fragment of the previous chapter with parallelism. However, communication is postponed until the next chapter. First, we will consider interleaving of actions without communication.

Note. CCS has only one operator for parallelism. In [14], Moller shows that we need at least one auxiliary operator to get a finite, complete set of axioms. Although this is only shown for the case without restriction, it is probable that the same holds for the case with restriction. Therefore we define parallelism by introducing the $\parallel$, $\between$ and $\mid$ operators of ACP [5] to make a finite, complete set possible.

We proceed by introducing extensions adding parallelism to the grammar of the original process term set $\mathcal{P}$, set of normal forms $\mathcal{P}^{\mathrm{nf}}$, and algebra $\mathbf{P}$ called respectively $\mathcal{P}_F$, $\mathcal{P}_F^{\mathrm{nf}}$ and $\mathbf{P}_F$.

The process terms of $\mathcal{P}_F$ are generated by the following grammar:

$$\mathrm{P} \ ::= \ \mathbf{0} \ \mid \ x \ \mid \ a.\mathrm{P} \ \mid \ \mathrm{P} + \mathrm{P} \ \mid \ \mathrm{P} \setminus H \ \mid \ \mathrm{P} \between \mathrm{P} \ \mid \ \mathrm{P} \mid \mathrm{P} \ \mid \ \mathrm{P} \parallel \mathrm{P}$$

where $a \in \mathcal{A}$, $x \in \mathcal{V}$, and $H \subseteq \mathcal{A}$.

Table 4.1 contains an extension of the operational semantics given in Table 3.1 that adds parallelism in a purely interleaving fashion.

$$5\frac{p \xrightarrow{a} p'}{p \between q \xrightarrow{a} p' \parallel q} \qquad 6\frac{p \xrightarrow{a} p'}{p \parallel q \xrightarrow{a} p' \parallel q} \qquad 7\frac{q \xrightarrow{a} q'}{p \parallel q \xrightarrow{a} p \parallel q'}$$

Table 4.1: The operational semantics extension with interleaving

Definition 4.1. The algebra $\mathbf{P}_F$ extends the algebra $\mathbf{P}$ (see Definition 2.7) and is based on the equivalence classes obtained by dividing $\leftrightarrows$ on $\mathcal{P}_F^0$. It adds the following operators:

$$[p] \between [q] = [p \between q], \quad [p] \mid [q] = [p \mid q],$$
$$[p] \parallel [q] = [p \parallel q].$$

Now that the operational semantics is extended, we can lift the extension to the process algebra level by extending Proposition 2.8 as follows:

Proposition 4.2. *For all $p, q, r \in \mathbf{P}_F$ and $a, b \in \mathcal{A}$*

5. *$p \between q \xrightarrow{a} r$ iff there exists $p' \in \mathbf{P}_F$ such that $p \xrightarrow{a} p'$ and $r = p' \parallel q$;*
6. *$p \parallel q \xrightarrow{a} r$ iff $p \between q \xrightarrow{a} r$ or $q \between p \xrightarrow{a} r$.*

Table 4.2 contains an extension of the axiom schemata given in Table 3.2 for $\mathbf{P}$ that adds parallelism, called $\mathbf{P}_F$. Refer to Appendix A.2 for the full equational base. The resulting equational base is finite if $\mathcal{L}$ is finite.

(L1)	$\mathbf{0} \between x$	$\approx \mathbf{0}$
(L2)	$a.x \between y$	$\approx a.(x \parallel y)$
(L3)	$(x + y) \between z$	$\approx x \between z + y \between z$
(L4)	$(x \between y) \between z$	$\approx x \between (y \parallel z)$
(L5)	$x \between \mathbf{0}$	$\approx x$
(D5)	$(x \between y) \setminus H$	$\approx x \setminus H \between y \setminus H$
(M)	$x \parallel y$	$\approx x \between y + y \between x + x \mid y$
(F)	$x \mid y$	$\approx \mathbf{0}$

Table 4.2: The axiom extension for $\mathbf{P}_F$

The new grammar leads to an extension of the definition of the number of symbols of a process term. The definition of length and branching degree will remain the same, since these were defined on the semantics of the process term.

Definition 4.3. We define the number of symbols of a process term $p \in \mathcal{P}_F$ by extending Definition 2.4 with the following rules:

$$(\!|q \between r|\!) = 1 + (\!|q|\!) + (\!|r|\!), \qquad (\!|q \parallel r|\!) = 1 + (\!|q|\!) + (\!|r|\!),$$
$$(\!|q \mid r|\!) = 1 + (\!|q|\!) + (\!|r|\!).$$

4.1 Normal Forms

The normal forms of $\mathcal{P}_F^{\mathrm{nf}}$ are generated by the following grammar:

$$\mathrm{N} \ ::= \ \mathbf{0} \ \mid \ a.\mathrm{N} \ \mid \ \mathrm{N} + \mathrm{N} \ \mid \ (x \setminus H) \between \mathrm{N}$$

where $a \in \mathcal{A}$, $x, y \in \mathcal{V}$ and $H \subset \mathcal{A}$.

The normal forms are similar to the normal forms of $\mathcal{P}^{\mathrm{nf}}$. The rule $N ::= x \setminus H$ got replaced by $N ::= (x \setminus H) \between \mathrm{N}$. Therefore we also call $(x \setminus H) \between \mathrm{N}$ a simple normal form.

Lemma 4.4. *For every normal form $s \in \mathcal{P}_F^{\mathrm{nf}}$ and $H \subseteq \mathcal{A}$ a normal form $s' \in \mathcal{P}_F^{\mathrm{nf}}$ exists such that $s' \approx s \setminus H$.*

Proof. This can be shown by induction on the structure of s. We adapt Lemma 3.2 by replacing case 4 with:

4. If $s = (x \backslash J) \between t$ with $t \in \mathcal{P}_F^{\mathrm{nf}}$, then $s \backslash H = ((x \backslash J) \between t) \backslash H \approx ((x \backslash J) \backslash H) \between t \backslash H$ by D5. By induction hypothesis a $t' \in \mathcal{P}_F$ exists such that $t' \approx t \backslash H$. Hence, by DX3 we have that $((x \setminus J) \setminus H) \between t \setminus H \approx (x \setminus J \cup H) \between t'$ and $(x \setminus J \cup H) \between t' \in \mathcal{P}_F^{\mathrm{nf}}$. □

We introduce the following lemma, similar to the previous lemma, to simplify the normal form proof.

Lemma 4.5. *For all normal forms $s, t \in \mathcal{P}_F^{\mathrm{nf}}$ a normal form $u \in \mathcal{P}_F^{\mathrm{nf}}$ exists such that $u \approx s \between t$.*

Proof. This can be shown by induction on the number of symbols in s and t, $(\!|s|\!) + (\!|t|\!)$. We distinguish cases according to the syntactic form of s.

1. If $s = \mathbf{0}$, then $s \between t = \mathbf{0} \between t \approx \mathbf{0}$ by L1 and $\mathbf{0} \in \mathcal{P}_F^{\mathrm{nf}}$.

2. If $s = a.s'$, then $s \between t = a.s' \between t \approx a.(s' \parallel t) \approx a.(s' \between t + t \between s')$ by L2, M and F. Since $(\!|s' \between t|\!) < (\!|s \between t|\!)$ and $(\!|t \between s'|\!) < (\!|s \between t|\!)$, we know by the induction hypothesis that two normal forms $u', u'' \in \mathcal{P}_F^{\mathrm{nf}}$ exist such that $u' \approx s' \between t$ and $u'' \approx t \between s'$. Hence, $s \between t \approx a.(u' + u'')$ and since $(u' + u'') \in \mathcal{P}_F^{\mathrm{nf}}$, also $a.(u' + u'') \in \mathcal{P}_F^{\mathrm{nf}}$.

3. If $s = s' + s''$, then $s \between t = (s' + s'') \between t \approx s' \between t + s'' \between t$ by L3. Since $(\!|s' \between t|\!) < (\!|s \between t|\!)$ and $(\!|s'' \between t|\!) < (\!|s \between t|\!)$, we have by the induction hypothesis that two normal forms $u', u'' \in \mathcal{P}_F^{\mathrm{nf}}$ exist such that $u' = s' \between t$ and $u'' = s'' \between t$. Hence, $s \between t \approx u' + u''$ and $(u' + u'') \in \mathcal{P}_F^{\mathrm{nf}}$.

4. If $s = (x \backslash H) \between s'$, then $s \between t = ((x \backslash H) \between s') \between t \approx (x \backslash H) \between (s' \between t + t \between s')$ by L4, M and F. Similarly to the previous case, we have two normal forms $u', u'' \in \mathcal{P}_F^{\mathrm{nf}}$, so that $s \between t = (x \setminus H) \between (u' + u'')$. Since, $(u' + u'') \in \mathcal{P}_F^{\mathrm{nf}}$, also $(x \setminus H) \between (u' + u'') \in \mathcal{P}_F^{\mathrm{nf}}$. □

To prove that every process term has a normal form, we extend Lemma 3.3 to suit the new forms that some $p \in \mathcal{P}_F$ can take.

Lemma 4.6. *Every process term $p \in \mathcal{P}_F$ has a normal form $s \in \mathcal{P}_F^{\mathrm{nf}}$ such that $p \approx s$.*

Proof. This can be shown by induction on the structure of p. We adapt Lemma 3.3 by replacing case 5 with:

5. If $p = q \backslash H$, then we know that a $t \in \mathcal{P}_F^{\mathrm{nf}}$ exists such that $q \approx t$. Lemma 4.4 gives us that if $t \in \mathcal{P}_F^{\mathrm{nf}}$, then a $s \in \mathcal{P}_F^{\mathrm{nf}}$ exists such that $s \approx t \setminus H$. Then, $p \approx t \setminus H \approx s$ and $s \in \mathcal{P}_F^{\mathrm{nf}}$.

Then, we add the following cases to supplement the case analysis already performed in Lemma 3.3:

6. If $p = q \between r$, then we know by the induction hypothesis that there are two normal forms $t, u \in \mathcal{P}_F^{\mathrm{nf}}$ such that respectively $q \approx t$ and $r \approx u$. Therefore, by Lemma 4.5, we have that a normal form $s \in \mathcal{P}_F^{\mathrm{nf}}$ exists such that $s \approx t \between u$. Hence, $p \approx t \between u \approx s$ and $s \in \mathcal{P}_F^{\mathrm{nf}}$.

7. If $p = q \mid r$, then $p \approx \mathbf{0}$ by F and $\mathbf{0} \in \mathcal{P}_F^{\mathrm{nf}}$.

8. If $p = q \parallel r$, we have by M, F and A4 that $p \approx q \between r + r \between q$. Similarly as in case 6 we know that there must be two normal forms $s, t \in \mathcal{P}_F^{\mathrm{nf}}$ such that respectively $s \approx q \between r$ and $t \approx r \between q$. Hence, $p \approx s + t$ and $(s + t) \in \mathcal{P}_F^{\mathrm{nf}}$. □

Lemma 4.7. *If $s \in \mathcal{P}_F^{\mathrm{nf}}$, then it can be written in the following general form (modulo A1, A2 and A4):*

$$s = \sum_{i \in I} a_i.s_i + \sum_{j \in K} (x_j \setminus H_j) \between s_j \quad \textit{and } s_i, s_j \in \mathcal{P}_F^{\mathrm{nf}}$$

with $a_i \in \mathcal{A}, x_j \in \mathcal{V}$ and $H_j \subset \mathcal{A}$.

Proof. This lemma is an adaptation of Lemma 3.4 where $x_j \setminus H_j$ is now replaced by $(x_j \setminus H_j) \between s_j$. □

4.2 Soundness and Completeness

Proposition 4.8 (Soundness)**.** *For all process terms $p, q \in \mathbf{P}_F$, if $p \approx q$, then $p \leftrightarrows q$.*

Proof. Since the axioms given in Table 3.2 and L1–L5, M of Table 4.2 and their soundness is well-known (see [5, 7, 10]), we are only going to show the soundness of D5 as follows:

We use the following symmetric relation $\mathcal{R}$:

$$\begin{aligned}\mathcal{R} = \{&(\ (p \between q) \setminus H,\ (p \setminus H) \between (q \setminus H)\ ),\\ &(\ (p \setminus H) \between (q \setminus H),\ (p \between q) \setminus H\ ),\\ &(\ (p \parallel q) \setminus H,\ (p \setminus H) \parallel (q \setminus H)\ ),\\ &(\ (p \setminus H) \parallel (q \setminus H),\ (p \parallel q) \setminus H\ ) \mid p, q \in \mathcal{P}_F, H \subseteq \mathcal{A}\}\end{aligned}$$

and show that this is a bisimulation following Definition 2.5 for all $p, q \in \mathcal{P}_F$ and $H \subset \mathcal{A}$:

1. $(\ (p \between q) \setminus H,\ (p \setminus H) \between (q \setminus H)\ )$: Assume that $(p \between q) \setminus H \xrightarrow{a} r$ for some $r \in \mathcal{P}_F$ and $a \in \mathcal{A}$. Then, by the semantics given in Table 3.1 this is possible only by rule 4 when $p \between q \xrightarrow{a} r'$, $a \notin H$, and $r = r' \setminus H$ for some $r' \in \mathcal{P}_F^{\mathrm{nf}}$. So, we must have that $p \between q \xrightarrow{a} r'$. Then, by the semantics given in Table 4.1 this is possible only by rule 5 when $p \xrightarrow{a} p'$ and $r' = p' \between q$.

   But, if $p \xrightarrow{a} p'$ and $a \notin H$, then $p \setminus H \xrightarrow{a} p' \setminus H$ and also $(p \setminus H) \between (q \setminus H) \xrightarrow{a} (p \setminus H) \parallel (q \setminus H)$.

   Finally, $(\ (p' \parallel q) \setminus H,\ (p \setminus H) \parallel (q \setminus H)\ ) \in \mathcal{R}$.

2. $(\ (p \setminus H) \between (q \setminus H),\ (p \between q) \setminus H\ )$: This case is symmetrical with the previous case.

3. $(\ (p \parallel q) \setminus H,\ (p \setminus H) \parallel (q \setminus H)\ )$: We know that $(p \parallel q)\setminus H \xrightarrow{a} (p' \parallel q)\setminus H$ if $a \notin H$ and $p \parallel q \xrightarrow{a} p' \parallel q$ or that $(p \parallel q)\setminus H \xrightarrow{a} (p \parallel q')\setminus H$ if $a \notin H$ and $p \parallel q \xrightarrow{a} p \parallel q$. This is only possible if $p \xrightarrow{a} p'$ or if $q \xrightarrow{q}{}'$ respectively.

   We only consider the case where $p \xrightarrow{a} p'$ since the case for $q \xrightarrow{a} q'$ is symmetrical: if $p \xrightarrow{a} p'$ and $a \notin H$, then $p\setminus H \xrightarrow{a} p'\setminus H$ and consequently $(p \setminus H) \parallel (q \setminus H) \xrightarrow{a} (p' \setminus H) \parallel (q \setminus H)$.

   Finally, we have that $(\ (p' \parallel q) \setminus H,\ (p \setminus H) \parallel (q \setminus H)\ ) \in \mathcal{R}$.

4. $(\ (p \setminus H) \parallel (q \setminus H),\ (p \parallel q) \setminus H\ )$: This case is symmetrical with the previous case.

Hence, if $(p \between q) \setminus H \approx (p \setminus H) \between (q \setminus H)$, then $\mathcal{R}$ is a bisimulation relation such that $(p \between q) \setminus H \leftrightarrows_{\mathcal{R}} (p \setminus H) \between (q \setminus H)$. □

Similarly to the completeness proof for the fragment of CCS without parallelism we need a distinguishing valuation. However, now that parallelism has been introduced it can no longer be based on the length, since for example the processes $a.\mathbf{0} \parallel a.\mathbf{0}$ and $a.a.\mathbf{0}$ are bisimilar and thus have the same length but are syntactically different. Therefore, we introduce a distinguishing valuation based on branching degree combined with the notion of length.

Definition 4.9. We define the valuation $\diamond_w$ by assigning a unique identifiable process for each variable $x \in \mathcal{V}$:

$$\diamond_w(x) = \sum_{a \in \mathcal{A}} a.\xi_{\lceil x \rceil \cdot w} \text{ with } \xi_i = \sum_{j=1}^{i} \psi_j$$

with $w \geq 1$ and some injective function $\lceil \cdot \rceil : \mathcal{V} \to (\mathbb{N} - \{0\})$.

If w is chosen high enough, i.e. higher than the branching that can occur in the processes of the parallel decomposition of a term and higher than the cardinality of the set of actions $\mathcal{A}$, the difference between a left merge term with a variable and a prefix term can be detected.

To accomplish that, we need to choose w higher than the lower bound of the branching degree of the processes that have to be distinguished. The evaluation mapping using $\diamond_w$ on a variable results in a process that not immediately has a distinguishably high branching degree, but it gets a high branching degree after performing an action.

Definition 4.10. For all $s \in \mathcal{P}_F^{\text{nf}}$, the lower bound estimation of the branching degree $\overline{\langle s \rangle}$ is defined inductively as follows:

$$\begin{aligned} &\overline{\langle \mathbf{0} \rangle} = 0, && \overline{\langle s + t \rangle} = \overline{\langle s \rangle} + \overline{\langle t \rangle}, \\ &\overline{\langle a.t \rangle} = \max(1, \overline{\langle t \rangle}), && \overline{\langle (x \setminus H) \between t \rangle} = \max(|\mathcal{A} - H|, \overline{\langle t \rangle}). \end{aligned}$$

with $a \in \mathcal{A}$, $x \in \mathcal{V}$, $H \subset \mathcal{A}$ and $t \in \mathcal{P}_F^{\text{nf}}$.

Note. The lower bound $|\mathcal{A} - H|$ follows from the definition of $\diamond_w$ (see Definition 4.9), since $[\![x \setminus H]\!]_{\diamond_w} \xrightarrow{a} \xi_{\lceil x \rceil \cdot w}$ for all $a \in \mathcal{A} - H$.

As for $*_m$, it also holds for the valuation $\diamond_w$ that when it is applied to a simple normal form such as $(x \setminus H) \between s$, the residual of the process after performing any action is unique. In this case the unique residual is $(\xi_{\lceil x \rceil \cdot w} \setminus H) \parallel [\![s]\!]_{\diamond_w}$.

Example 4.11. Consider a set of action labels $\mathcal{L} = \{a, b\}$, a set of variables $\mathcal{V} = \{x\}$ with $\lceil x \rceil = 1$ and a process term $t = a.a.a.\mathbf{0} + (x \setminus \{b\}) \between a.\mathbf{0}$. Since $\overline{\langle t \rangle} = 2$, we choose $w = 3$. Then:

$$\begin{aligned}\diamond_w(x) = a.\xi_3 + b.\xi_3 &= a.(\psi_1 + \psi_2 + \psi_3) + b.(\psi_1 + \psi_2 + \psi_3) \\ &= a.(a.\mathbf{0} + b.\mathbf{0} + a.a.\mathbf{0} + b.b.\mathbf{0} + a.a.a.\mathbf{0} + b.b.b.\mathbf{0}) \\ &\quad + b.(a.\mathbf{0} + b.\mathbf{0} + a.a.\mathbf{0} + b.b.\mathbf{0} + a.a.a.\mathbf{0} + b.b.b.\mathbf{0})\end{aligned}$$

and we have that $[\![t]\!]_{\diamond_w} = a.a.a.\mathbf{0} + a.\,((a.\mathbf{0} + a.a.\mathbf{0} + a.a.a.\mathbf{0}) \parallel a.\mathbf{0})$.

When $[\![t]\!]_{\diamond_w}$ performs the action a, we can exactly determine which summand of $[\![t]\!]_{\diamond_w}$ performed that action. If it was the closed term $a.a.a.0$, then the branching degree of the residual $a.a.\mathbf{0}$ does not exceed w. If it was the open term $(x \setminus \{b\}) \between a.\mathbf{0}$, then the residual $(\xi_3 \setminus \{b\}) \parallel a.\mathbf{0}$ has a branching degree 3, which exceeds w.

The following lemma shows that our branching degree estimation is a still a good estimation when a valuation is applied.

Lemma 4.12. *For every normal form $s \in \mathcal{P}_F^{\mathrm{nf}}$, if $\overline{\langle s \rangle} < w$, then $\langle [\![s]\!]_{\diamond_w} \rangle \leq w$.*

Proof. Structural induction on s. □

With parallelism in the algebra it is harder to distinguish a prefix normal form from a normal form starting with a variable. We will work towards distinguishing them based on the branching degree of the elements in a unique parallel decomposition which is defined below. We will show that the elements of the composition in the residual of a prefix normal still have a low branching degree, whereas the residual a normal form starting with a variable will contain a component with a high branching degree.

Definition 4.13. An element $p \in \mathbf{P}_F$ is *parallel prime* if $p \neq \mathbf{0}$, and $p = q \parallel r$ implies $q = \mathbf{0}$ or $r = \mathbf{0}$.

A *parallel decomposition* of p is a finite multiset $[p_1, \ldots, p_n]$ of parallel primes such that $p = p_1 \parallel \ldots \parallel p_n$.

Note. The decomposition of the process $\mathbf{0}$ is the empty multiset, and the decomposition of a parallel prime process is the singleton multiset $[p]$[1].

Lemma 4.14. *Every element of $\mathbf{P}_F$ has a unique parallel decomposition.*

Proof. Please refer to [2, Theorem 11] for the proof. This proof is done for a fragment of ACP that is similar to our fragment of CCS, $\mathbf{P}_F$. All ingredients necessary to redo the proof at this point are present in this thesis. □

[1] The notation for the equivalence class of p is the same as the singleton multiset containing the process p.

Corollary 4.15. *Let $p, q, r \in \mathbf{P}_F$. If $p \parallel q = p \parallel r$, then $q = r$.*

The following lemmas show some of the properties of an important parallel prime process $\xi_i \setminus H$. This process is a component of the residual of a normal form that starts with a variable when the valuation $\diamond_w$ is used.

Lemma 4.16. *For all $i \geq 1$ and $H \subset \mathcal{A}$, the process $\xi_i \setminus H$*

1. *is parallel prime, and*
2. *its branching degree is $\{\!|\xi_i \setminus H|\!\} = i \cdot |\mathcal{A} - H|$.*

Proof. 1. We use an adapted version of [2, Lemma 14(i)]: Clearly $\xi_i \setminus H \neq \mathbf{0}$, given also that $H \neq \mathcal{A}$. Suppose $\xi_i \setminus H = p \parallel q$; to prove that $\xi_i \setminus H$ is parallel prime, we need to establish that either $p = \mathbf{0}$ or $q = \mathbf{0}$. Note that $p \parallel q \xrightarrow{a} \mathbf{0}$ for some $a \in \mathcal{A} - H$. Since $p \mid q \approx \mathbf{0}$ (by F), it follows from Proposition 4.2 that we can distinguish two cases:

   (a) If there exists p' such that $p \xrightarrow{a} p'$ with $a \in \mathcal{A} - H$ and $p' \parallel q = \mathbf{0}$, then it follows by Definition 2.4 that $|p' \parallel q| = 0$ and therefore $|q| = 0$, hence $q = \mathbf{0}$.

   (b) If there exists q' such that $q \xrightarrow{a} q'$ with $a \in \mathcal{A} - H$ and $p \parallel q' = \mathbf{0}$, then it follows by Definition 2.4 that $|p \parallel q'| = 0$ and therefore $|p| = 0$, hence $p = \mathbf{0}$.

2. Consider the branching degree using the definition of $\diamond_w$ (see Definition 4.9), ψ_j (see Definition 3.6), and the axioms D1–D4. We can show modulo bisimulation that:

$$\{\!|\xi_i \setminus H|\!\} = \left\langle\!\left| \sum_{j=1}^{i} \psi_j \setminus H \right|\!\right\rangle = \left\langle\!\left| \sum_{j=1}^{i} \sum_{a \in \mathcal{A}-H} a^j.\mathbf{0} \right|\!\right\rangle = i \cdot |\mathcal{A} - H|.$$

   The above equation may be explained as follows: by Definition 3.6, D2, and D3, the process $\psi_j \setminus H$ can perform all actions except the actions present in H. By Definition 4.9, $\xi_i \setminus H$ is actually i times a $\psi_j \setminus H$ process, each with different length and therefore not bisimilar with any of the other $i - 1$ processes.

   So, $\xi_i \setminus H$ can perform $|\mathcal{A} - H|$ distinct actions and have i bisimilarly different residuals. Hence, the branching degree of $\xi_i \setminus H$ is $i \cdot |\mathcal{A} - H|$. □

Lemma 4.17. *For $i, j \geq 1$, $H, J \subset \mathcal{A}$, it holds that if the processes $p = \xi_i \setminus H$ and $q = \xi_j \setminus J$ are equal, then $H = J$.*

Proof. Because of the definition of ξ_i (see Definition 4.9), we know that $p \xrightarrow{a} p'$ and therefore $q \xrightarrow{a} q'$ for all $a \in \mathcal{A} - H$.

Assume that $a \notin H$. By Definition 4.9, Proposition 2.8, and D3 we know that some $r \in \mathbf{P}$ exists such that $\xi_i \setminus H \xrightarrow{a} r$ and therefore $\xi_j \setminus J \xrightarrow{a} r$. However, by D3 this also means that $a \notin J$.

The case when assuming that $a \notin J$ is symmetrical. Hence, since $a \notin H$ iff $a \notin J$, $H = J$. □

Lemma 4.18. *For all $p, q \in \mathbf{P}_F$, it holds that $\langle p \parallel q \rangle \geq \langle p \rangle$ and $\langle p \parallel q \rangle \geq \langle q \rangle$.*

Proof. We use the proof given in [2, Lemma 13]: First we prove that $\langle p \parallel q \rangle \geq \langle q \rangle$. By Proposition 4.2, if $q \xrightarrow{a} q'$, then $p \parallel q \xrightarrow{a} p \parallel q'$. Suppose that q_1 and q_2 are distinct processes such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$. Then $p \parallel q \xrightarrow{a} p \parallel q_1$ and $p \parallel q \xrightarrow{a} p \parallel q_2$. Since $p \parallel q_1 = p \parallel q_2$ implies that $q_1 = q_2$, by Corollary 4.15, it follows that $p \parallel q_1$ and $p \parallel q_2$ are distinct. Hence $\langle p \parallel q \rangle \geq \langle q \rangle$. By commutativity of $\parallel$, it also follows that $\langle p \parallel q \rangle \geq \langle p \rangle$. □

Lemma 4.19. *Let $s, s' \in \mathcal{P}_F^{\mathrm{nf}}$ be simple normal forms, $x \in \mathcal{V}$, $H \subset \mathcal{A}$ and let $w > \overline{\langle s \rangle}$. If $s = (x \setminus H) \between s'$, then the unique residual of $[\![s]\!]_{\diamond_w}$ has a branching degree larger than w, whereas if $s = a.s'$, then the unique residual of $[\![s]\!]_{\diamond_w}$ has a branching degree smaller than or equal to w.*

Proof. Assume that p is the residual of s: $[\![s]\!]_{\diamond_w} \xrightarrow{a} p$ for some $a \in \mathcal{A}$. We have the following residuals:

1. If $s = a.s'$, then $p = [\![s']\!]_{\diamond_w}$. Because $\overline{\langle s \rangle} < w$, we know that the branching degree $\overline{\langle s' \rangle} \leq \overline{\langle s \rangle} < w$ by Definition 4.10. Hence, by Lemma 4.12, the branching degree of $[\![s']\!]_{\diamond_w}$ does not exceed w.

2. If $s = (x \setminus H) \between s'$, then $p = (\xi_{\lceil x \rceil \cdot w} \setminus H) \parallel [\![s']\!]_{\diamond_w}$. We have by Definition 4.9 that $\langle \xi_{\lceil x \rceil \cdot w} \setminus H \rangle = \lceil x \rceil \cdot w \cdot |\mathcal{A} - H| \geq w$ (given that $H \subset \mathcal{A}$ and $\lceil x \rceil \geq 1$). Because $[\![s']\!]_{\diamond_w}$ does not decrease the branching degree of the residual (by Lemma 4.18), we may conclude that the residual has a branching degree that exceeds w. □

Theorem 4.20 (Completeness)**.** *For every two normal forms $s, t \in \mathcal{P}_F^{\mathrm{nf}}$ with $w > \overline{\langle s \rangle}$ and $w > \overline{\langle t \rangle}$, it holds that if $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$, then $s \approx t$ modulo A1–A4.*

Proof. By Lemma 4.7, we can assume that the normal forms s and t are summations of terms of the form $a.s'$ or $(x \setminus H) \between s'$. We now prove that $s \approx t$ assuming that $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$ by induction on the sum of the lengths of s and t. We do this by proving that for every summand s_i of s a summand t_j of t exists such that $s_i \approx t_j$ modulo A1–A4. Consider the following case analysis based on the syntax of an arbitrary summand s_i of s:

1. If $s_i = a.s_i'$, then $[\![s_i]\!]_{\diamond_w} \xrightarrow{a} [\![s_i']\!]_{\diamond_w}$. Because $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$, there also must be a t_j in t such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s_i']\!]_{\diamond_w}$. By Lemma 4.19 we know that t_j must have the form $b.t_j'$, because the branching degree of the residual $[\![t_j']\!]_{\diamond_w}$ does not exceed w.

   Given that t_j has this form, it can only perform one action: $[\![t_j]\!]_{\diamond_w} \xrightarrow{b} [\![t_j']\!]_{\diamond_w}$. Since also $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s_i']\!]_{\diamond_w}$ if follows that $a = b$ and $[\![s_i']\!]_{\diamond_w} = [\![t_j']\!]_{\diamond_w}$. By induction hypothesis we have that $s_i' \approx t_j'$ modulo A1–A4. Hence, we may conclude that $s_i = a.s_i' \approx b.t_j' = t_j$.

2. If $s_i = (x \setminus H) \between s_i'$, then, since $H \subset \mathcal{A}$, $[\![s_i]\!]_{\diamond_w} \xrightarrow{a} p$ for some $a \in \mathcal{A} - H$. We know that also a t_j in t exists such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} p$. Definition 4.9 gives us that $p = (\xi_{\lceil x \rceil \cdot w} \setminus H) \parallel [\![s_i']\!]_{\diamond_w}$. Similarly to the previous case, by Lemma 4.19 we also know that t_j must have the form $(y \setminus J) \between t_j'$ for some

$y \in \mathcal{V}$ and $J \subset \mathcal{A}$. The residual of t_j after performing an action $a \in \mathcal{A} - J$ is $(\xi_{\lceil y \rceil \cdot w} \setminus J) \parallel [\![t'_j]\!]_{\diamond_w}$ (also by Definition 4.9). This residual is equal to p, so we know that $(\xi_{\lceil x \rceil \cdot w} \setminus H) \parallel [\![s'_i]\!]_{\diamond_w} = (\xi_{\lceil y \rceil \cdot w} \setminus J) \parallel [\![t'_j]\!]_{\diamond_w}$.

By Lemma 4.16 we have that the process $\xi_{\lceil x \rceil \cdot w} \setminus H$ is parallel prime and has a branching degree that exceeds w. This process cannot occur in the unique parallel decomposition of $[\![t'_j]\!]_{\diamond_w}$ (see Definition 4.13) because, by Lemma 4.18 and the fact that $w > \overline{\langle t'_j \rangle}$, the branching degrees of all processes in the decomposition do not exceed w. Conversely, this also holds in a symmetric way for the process $\xi_{\lceil y \rceil \cdot w} \setminus J$ with respect to the unique parallel decomposition of $[\![s'_i]\!]_{\diamond_w}$. Hence, $\xi_{\lceil x \rceil \cdot w} \setminus H = \xi_{\lceil y \rceil \cdot w} \setminus J$ and $[\![s'_i]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$. Therefore, we proceed show that $H = J$, $x = y$ and $[\![s'_i]\!]_{\diamond_w} = [\![t_j]\!]_{\diamond_w}$.

From $\xi_{\lceil x \rceil \cdot w} \setminus H = \xi_{\lceil y \rceil \cdot w} \setminus J$ it follows by Lemma 4.17 that $H = J$. Now, consider the branching degrees of the two processes. From Lemma 4.16 it follows that

$$\lceil x \rceil \cdot w \cdot |\mathcal{A} - H| = \langle\!\langle \xi_{\lceil x \rceil \cdot w} \setminus H \rangle\!\rangle = \langle\!\langle \xi_{\lceil y \rceil \cdot w} \setminus J \rangle\!\rangle = \lceil y \rceil \cdot w \cdot |\mathcal{A} - J|.$$

Since $\lceil \cdot \rceil$ is an injection and $H = J$, it follows that $\lceil x \rceil = \lceil y \rceil$, so $x = y$.

By the two previous observations that $H = J$ and $x = y$ we have that the residual of $[\![t_j]\!]_{\diamond_w}$ is $(\xi_{\lceil x \rceil \cdot w} \setminus H) \parallel [\![t'_i]\!]_{\diamond_w}$. We also know that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} (\xi_{\lceil x \rceil \cdot w} \setminus H) \parallel [\![s'_i]\!]_{\diamond_w}$, so it follows by cancellation (see Corollary 4.15) that $[\![s'_i]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$.

Now it has been established that $H = J$, $x = y$, and $[\![s'_i]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$; we may conclude that $s_i = (x \setminus H) \between s'_i = (y \setminus J) \between t'_j = t_j$.

The above analysis shows that for each summand s_i of s a summand t_j of t exists such that $s_i \approx t_i$ modulo A1–A4. It follows by a symmetric argument that every summand of t is also provably equal to a summand of s. Hence $s \approx s + t \approx t$ modulo A1–A4. □

Corollary 4.21. *For all process terms $p, q \in \mathbf{P}_F$ it holds that $p \approx q$ if and only if $p \leftrightarrows q$.*

Chapter 5

Restriction and Communication

We extend the fragment of the previous chapter again. This time we add communication. For CCS the synchronous communication action is the internal or silent action labelled with τ. We define the complete set of actions $\mathcal{A}_\tau$ as $\mathcal{A} \cup \{\tau\}$. In CCS communication is defined as follows: if some process can perform some action $a \in \mathcal{A}$ and another process in parallel with the first process can perform the corresponding co-action $\overline{a}$, then a synchronous communication occurs resulting in the silent action τ. Note that the resulting action of a communication τ can never communicate with any other process.

We now introduce extensions of the sets of process terms and the algebra that adds communication to $\mathcal{P}_F$, $\mathcal{P}_F^{\text{nf}}$ and $\mathbf{P}_F$ called respectively $\mathcal{P}_H$, $\mathcal{P}_H^{\text{nf}}$ and $\mathbf{P}_H$. Although the process terms of $\mathcal{P}_H$ are the same as $\mathcal{P}_F$ and thus there is no real extension, we shall use the $\mathcal{P}_H$ set notation for clarity.

Table 5.1 contains an extension of the operational semantics reflecting the communication principle.

$$8\ \frac{p \xrightarrow{a} p' \qquad q \xrightarrow{\overline{a}} q'}{p \mid q \xrightarrow{\tau} p' \parallel q'} \qquad 9\ \frac{p \xrightarrow{a} p' \qquad q \xrightarrow{\overline{a}} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

Table 5.1: The operational semantics extension with communication

The extended operational semantics leads to an adaptation of Proposition 4.2 so that we can lift the communication behaviour to the algebra level. Note that we need to replace case 6, now that communication is part of a merge $p \parallel q$.

Proposition 5.1. *For all $p, q, r \in \mathbf{P}_H$ and $a \in \mathcal{A}_\tau$*

6. *$p \parallel q \xrightarrow{a} r$ iff $p \between q \xrightarrow{a} r$ or $q \between p \xrightarrow{a} r$ or $p \mid q \xrightarrow{a} r$;*
7. *$p \mid q \xrightarrow{a} r$ iff $a = \tau$ and there exist an action $b \in \mathcal{A}$ and processes $p', q' \in \mathbf{P}_H$ such that $p \xrightarrow{b} p'$, $q \xrightarrow{\overline{b}} q'$, and $r = p' \parallel q'$.*

Table 5.2 contains a proposed extension of the axiom schemata given in Table 4.2 for $\mathbf{P}_F$ that adds communication, called $\mathbf{P}_H$. With respect to $\mathbf{P}_F$ we have adapted axiom D2, removed axiom F and replaced it by the handshaking axiom H. We also have removed D5 because it is no longer sound (see Section 5.1). Refer to Appendix A.3 for the full equational base.

(D2)	$a.x \setminus H$	$\approx \mathbf{0}$	if $a, \overline{a} \in H$
(C1)	$\mathbf{0} \mid x$	$\approx \mathbf{0}$	
(C2)	$a.x \mid b.y$	$\approx \tau.(x \parallel y)$	if $b = \overline{a}$
(C3)	$a.x \mid b.y$	$\approx \mathbf{0}$	otherwise
(C4)	$(x + y) \mid z$	$\approx x \mid z + y \mid z$	
(C5)	$x \mid y$	$\approx y \mid x$	
(C6)	$(x \mid y) \mid z$	$\approx x \mid (y \mid z)$	
(C7)	$(x \between y) \mid z$	$\approx (x \mid z) \between y$	
(H)	$x \mid (y \mid z)$	$\approx \mathbf{0}$	

Table 5.2: The proposed axiom extension for $\mathbf{P}_H$

From here on, the chapter will follow a different structure as the previous chapters. First, we will consider the soundness of the base in the next section and show that obvious axioms for distribution of restriction over left merge and communication merge are not sound. These results lead us to believe that we deal with much more complex normal forms than we had in the previous cases. We will discuss the complexity issues of the normal forms in Section 5.2. Finally, we will propose an improved distinguishing valuation in Section 5.3 and show its distinguishing possibilities, followed by a discussion of other problems that arise before doing the completeness proof.

5.1 Soundness

The axioms C1–C7 and H are well-known axioms ([5, 7, 10]) and therefore we will not prove their soundness here. One might think of adding the axiom D5 (see page 18) and the following axiom, D6, to reduce the number of normal forms by distributing the restriction over communication merge:

$$\text{(D6)} \quad (p \mid q) \setminus H \approx (p \setminus H) \mid (q \setminus H).$$

However, the following example illustrates that D5 and D6 are not sound. In our example we apply the axioms in Table 5.2, which is allowed since they are sound.

Example 5.2. We prove by counter example through instantiation of the left-hand and right-hand sides of the equations and then showing that the resulting processes are not bisimilar.

Ad D5. Instantiate with $H = \{b\}$, $p = a.b.\mathbf{0}$ and $q = \overline{b}.c.\mathbf{0}$. Then we have for the left-hand side that

$$(p \between q) \setminus H = (a.b.\mathbf{0} \between \overline{b}.c.\mathbf{0}) \setminus \{b\} \approx a.((b.\mathbf{0} \parallel \overline{b}.c.\mathbf{0}) \setminus \{b\}) \approx a.\tau.c.\mathbf{0}$$

by L1, L2, C1, C2, M and D1–D3. We have for the right-hand side that

$$(p \setminus H) \between (q \setminus H) = (a.b.\mathbf{0} \setminus \{b\}) \between (\overline{b}.c.\mathbf{0} \setminus \{b\}) \approx a.\mathbf{0} \between \mathbf{0} \approx a.\mathbf{0}$$

by D1–D3, L2 and L2. It is obvious that $a.\tau.c.\mathbf{0} \not\approx a.\mathbf{0}$.

Ad D6. Instantiate with $H = \{a\}$, $p = a.b.\mathbf{0}$ and $q = \overline{a}.c.\mathbf{0}$. For the left-hand side we have that

$$(p \mid q) \setminus \{a\} = (a.b.\mathbf{0} \mid \overline{a}.c.\mathbf{0}) \setminus \{a\} \approx \tau.(b.c.\mathbf{0} + c.b.\mathbf{0})$$

by L1, L2, C1, C2, L5, M and D1–D3. For the right-hand side we have that

$$(p \setminus \{a\}) \mid (q \setminus \{a\}) = (a.b.\mathbf{0} \setminus \{a\}) \mid (\overline{a}.c.\mathbf{0} \setminus \{a\}) \approx \mathbf{0}$$

by D2 and C1. Again, it is obvious that $\tau.(b.c.\mathbf{0} + c.b.\mathbf{0}) \not\approx \mathbf{0}$.

Alphabet axioms

The previous example shows us that distribution of restriction over communication merge and left merge is not possible. In [4], Baeten et al. present *conditional alphabet axioms*, axioms that work with restriction based on the alphabet of actions that are present in the terms ($\alpha(p)$ for some term p). These conditional axioms could be added to the base as axiom schemata, but they do not consider open terms or co-actions.

If we consider the following conditional alphabet axiom from [4] (for ACP):

$$\text{(CA3)} \quad \alpha(x) \mid (\alpha(y) \cap H) \subseteq H \Rightarrow \quad (x \parallel y) \setminus H \approx (x \parallel (y \setminus H)) \setminus H,$$

then we can adapt it for CCS by leaving out the condition, because the result of a communication in CCS is always τ and can never be part of a restriction. Also using that $q \setminus H \approx q$ if q is a closed term that does not contain any of the actions contained in H (CA3 from [4]), we can introduce more general versions of the conditional axioms for open terms. Table 5.3 contains these axioms, called the *alphabet axioms* from here on, that we add to our proposed equational base.

(DL1)	$(x \between (y \setminus H)) \setminus H \approx (x \setminus H) \between (y \setminus H)$
(DL2)	$((x \setminus H) \between y) \setminus H \approx (x \setminus H) \between (y \setminus H)$
(DC1)	$(x \mid (y \setminus H)) \setminus H \approx (x \setminus H) \mid (y \setminus H)$

Table 5.3: The alphabet axioms for $\mathbf{P}_H$

The result of applying the alphabet axioms is that an "outer" restriction never includes actions present in the nearby restrictions "inside". The restriction sets of nested restrictions can be made disjoint to some extend. For example: $(a.b.\mathbf{0} \mid \overline{a}.\overline{b}.\mathbf{0} \setminus \{b\}) \setminus \{a, b\} \approx (a.b.\mathbf{0} \setminus \{b\} \mid \overline{a}.\overline{b}.\mathbf{0} \setminus \{b\}) \setminus \{a\}$. Hence, we have obtained some form of distribution of restriction, because we are partly able to push restrictions further inside terms.

Note. The axioms DL1, DL2 and DC1 only work in CCS because it has an symmetric communication relation of action and co-action and the fact that a restriction blocks both of them. In ACP, a communication function dictates which actions communicate and also what the resulting communication action is, which can be a member of a restriction set. In CCS τ is never a member of a restriction set.

5.2 Normal Forms

Due to the fact that a restriction $\setminus H$ cannot be distributed over $\between$ or $|$, we have considerably more normal forms for $\mathcal{P}_H$ than we have for $\mathcal{P}_F$. Note that although the alphabet axioms of Table 5.3 do not contribute much to the reduction of the amount of normal forms, they do introduce the useful "disjoint restriction set" property for nested restrictions.

As mentioned before in Section 3.1, the normal forms can be considered as the result of repeatedly applying the distributive axioms pushing the restriction inwards. However, this time, there are no distribution axioms for restriction over left merge and communication merge. This results in a large amount of normal forms.

Example 5.3. In this example we examine what kind of normal forms we can expect. We assume that $p, q \in \mathcal{P}_H^{\mathrm{nf}}$.

First of all, we have the normal forms of $\mathcal{P}_F$: $\mathbf{0}$, $a.p$, $p + q$ and $x \setminus H \between p$. Note however that in $\mathbf{P}_F$ the restriction distributes over left merge, $(x \between p) \setminus H \approx (x \setminus H) \between (p \setminus H)$, which is not possible in $\mathbf{P}_H$. So, we probably will have the normal form $((x \setminus H) \between p) \setminus J$ where $H \cap J = \emptyset$ by DL1, DL2.

Secondly, because we can not use L4 when a restriction is placed over $(x \between y)$ we have that the left-hand side of every left merge can again be a left merge under restriction. This can lead to normal forms with an unbounded nesting of left merges under restriction. An example of such a normal form is:

$$\Big(\big(((x \setminus H \between y) \setminus J) \between z\big) \setminus M\Big) \between p.$$

Note that for this normal form we do know that $H \cap J = J \cap M = L \cap M = \emptyset$ by DL1, DL2.

Thirdly, for the communication merge we will have similar problems with C7. However, this is similar to what is described in the previous paragraph. For now, we only consider normal forms with the following communication possibilities, either two variables communicate $(x \setminus H) \mid (y \setminus J)$ or a variable communicates with a prefix $(x \setminus H) \mid a.\mathbf{0}$.

We will not capture all these possibilities in a grammar as we have done before and we have no proof to establish all normal forms.

5.3 Completeness

The complexity of proving the completeness of the equational base of $\mathbf{P}_H$ can be attributed to the large amount of normal forms. A second cause is that if we have two simple normal forms, similar except for the fact that the restriction is distributed over the elements, then they are hard to distinguish.

The counter examples for the soundness of D5 and D6 show an exploitable difference. When restriction is applied directly to a variable (e.g. $(x \setminus H) \mid a.\mathbf{0}$), it is able to prematurely block communication, whereas it is not able to do so when the restriction is applied to a communication merge or merge (e.g. $(x \mid a.\mathbf{0}) \setminus H$ or $(x \parallel a.\mathbf{0}) \setminus H$). A solution for distinguishing these two types

of normal forms is to use this characteristic to adapt the $\diamond_w$ valuation in such a way that it allows all possible communications. The valuation should make sure that it is possible to determine which communication(s) happened from the residual.

Definition 5.4. We define the valuation $\sharp_w$ by assigning a unique identifiable process for each variable $x \in \mathcal{V}$:

$$\sharp_w(x) = \sum_{a \in \mathcal{A}} a.\zeta_{\lceil x \rceil \cdot w, \lfloor a \rfloor} \text{ with } \zeta_{i,k} = \sum_{j=1}^{i} \chi_{j \cdot k} \text{ and } \chi_n = \sum_{a \in \mathcal{A}_\tau} a^n.\mathbf{0}$$

with $w \geq 1$ and injective functions $\lceil \cdot \rceil : \mathcal{V} \to (\mathbb{N} - \{0\})$, $\lfloor \cdot \rfloor : \mathcal{A} \to (\mathbb{N} - \{0\})$.

Note that $\zeta_{i,k}$ always has a branching degree of at least i and an exact length of $i \cdot k$ because the term $\sum_{j=1}^{i} \tau^{j \cdot k}.\mathbf{0}$ is part of $\zeta_{i,k}$ and cannot be removed by any restriction.

The valuation $\sharp_w$ has the same branching degree properties as $\diamond_w$. However, the $\zeta_{i,k}$ process(es) in the residual will have a length that is a specific multiple of the action that was taken before getting this $\zeta_{i,k}$-process. This is useful for detecting which communication happened.

Example 5.5. Consider the two different normal forms $s = (x \mid a.\mathbf{0}) \setminus \{a\}$ and $t = (x \setminus \{a\}) \mid a.\mathbf{0}$. Evaluating these normal forms under $\sharp_w$ with $\lceil x \rceil = 1$ and $w = 2$, we have that

$$\begin{aligned} [\![s]\!]_{\sharp_w} &= ([\![x]\!]_{\sharp_w} \mid a.\mathbf{0}) \setminus \{a\} \approx \tau.(\zeta_{2,\lfloor \overline{a} \rfloor} \setminus \{a\}), \\ [\![t]\!]_{\sharp_w} &= ([\![x]\!]_{\sharp_w} \setminus \{a\}) \mid a.\mathbf{0} \approx \mathbf{0}. \end{aligned}$$

Not only can we see the difference between the two terms, but also we can detect from the residual of $[\![s]\!]_{\sharp_w}$ after performing the action τ, $[\![s]\!]_{\sharp_w} \xrightarrow{\tau} \zeta_{2,\lfloor \overline{a} \rfloor}$, that x was involved in a communication with $\overline{\overline{a}} = a$. This is also still possible if the restriction set contains all actions. Consider $s = (x \mid a.\mathbf{0}) \setminus \{a, b\}$ and $t = (x \mid b.\mathbf{0}) \setminus \{a, b\}$. Then,

$$\begin{aligned} [\![s]\!]_{\sharp_w} &= ([\![x]\!]_{\sharp_w} \mid a.\mathbf{0}) \setminus \{a, b\} \approx \tau.(\zeta_{2,\lfloor \overline{a} \rfloor} \setminus \{a, b\}) \approx \tau.(\tau^{\lfloor \overline{a} \rfloor}.\mathbf{0} + \tau^{2\lfloor \overline{a} \rfloor}.\mathbf{0}), \\ [\![t]\!]_{\sharp_w} &= ([\![x]\!]_{\sharp_w} \mid b.\mathbf{0}) \setminus \{a, b\} \approx \tau.(\zeta_{2,\lfloor \overline{b} \rfloor} \setminus \{a, b\}) \approx \tau.(\tau^{\lfloor \overline{b} \rfloor}.\mathbf{0} + \tau^{2\lfloor \overline{b} \rfloor}.\mathbf{0}). \end{aligned}$$

Note the difference in length of the residuals of $[\![s]\!]_{\sharp_w}$ and $[\![t]\!]_{\sharp_w}$, which is respectively $2\lfloor \overline{a} \rfloor$ and $2\lfloor \overline{b} \rfloor$.

Example 5.6. We now examine a more complicated case with nested restrictions. Given the normal form $u = ((x \setminus \{a\}) \mid (y \setminus \{b\})) \setminus \{c\}$ with $\lceil x \rceil = 1$, $\lceil y \rceil = 2$, $\mathcal{L} = \{a, b, c\}$, and $w = 2$, then we have that

$$\begin{aligned} [\![u]\!]_{\sharp_w} &= (([\![x]\!]_{\sharp_w} \setminus \{a\}) \mid ([\![y]\!]_{\sharp_w} \setminus \{b\})) \setminus \{c\} \\ &\approx \tau.\left(((\zeta_{2,\lfloor \overline{c} \rfloor} \setminus \{a\}) \parallel (\zeta_{4,\lfloor c \rfloor} \setminus \{b\})) \setminus \{c\}\right) \\ &\quad + \tau.\left(((\zeta_{2,\lfloor c \rfloor} \setminus \{a\}) \parallel (\zeta_{4,\lfloor \overline{c} \rfloor} \setminus \{b\})) \setminus \{c\}\right). \end{aligned}$$

The residual of $[\![u]\!]_{\sharp_w}$ is not unique, but either residual gives us the same information:

The residual $((\zeta_{2,\lfloor c \rfloor} \setminus \{a\}) \parallel (\zeta_{4,\lfloor \overline{c} \rfloor} \setminus \{b\})) \setminus \{c\}$ contains two parallel prime components with branching degrees that exceed w, so there must have been two variables in u. The high branching degrees have a typical value that is a multiple of w from which we can determine which variable was evaluated based on the choice of $\lceil \cdot \rceil$.

From the length of the residual, which is a multiple of $\lfloor c \rfloor$ or $\lfloor \overline{c} \rfloor$, we can see that a communication occurred of c with $\overline{c}$. The residual is not able to perform any c actions, though a communication occurred with c, hence there must be a restricting on $\{c\}$ outside. The inner restrictions can be determined based on the actions that each parallel prime component can perform.

Example 5.7. This example illustrates another complication when using the valuation $\sharp_w$. We consider an instantiation of the earlier presented example of a process with nested left merges and restrictions: $((((x \setminus \{b\}) \between y) \setminus \{a\}) \between z) \setminus \{b\}$. If we evaluate the normal form with $\sharp_w$ where $w = 2$, $\lceil x \rceil = 1$, and $\mathcal{L} = \{a, b\}$, then one of its residuals is: $((((\zeta_{2,\lfloor a \rfloor} \setminus \{b\}) \parallel [\![y]\!]_{\sharp_w}) \setminus \{a\}) \parallel [\![z]\!]_{\sharp_w}) \setminus \{b\}$. Before we can say anything about the parallel composition of the residual, we need to get rid of the restrictions. We will not go into detail about the resulting process term, however, it is clear that the result is complex.

We can conclude that the $\sharp_w$-valuation provides us with much more information about the structure of the normal form. Nevertheless, there are some open issues:

- The exact number and structure of normal forms is not yet known. Also, the proof establishing these normal forms has to be done.
- Example 5.6 shows us that it is possible to distinguish a directly applied restriction from a restriction on a communication. However, we need a proof to establish that it covers all the cases.
- How can we distinguish between $((x \mid p) \between q) \setminus H$ and $((x \mid p) \setminus H) \between q$ or even between $(x \setminus H) \between p$ and $(x \between p) \setminus H$?
- Given normal forms with nested left merges under restrictions, how are we going to accurately detect which restriction set has which effect?
- Are there more axioms like DL1, DL2 and DC1 that reduce the number of normal forms?

We believe that the proposed equational base, possibly extended with more alphabet axioms, can be proved complete when the normal forms are known. This result can be obtained with some adaptations of the distinguishing valuation $\sharp_w$ so that it be able to distinguish all normal forms types.

Chapter 6

Conclusion

In this thesis we have established finite equational bases for CCS with restriction. We have proved the soundness of the bases where necessary and used the normal form strategy to prove the completeness of the axiomatisations. Due to the complexity that restriction adds, we first started out with a basic fragment of CCS without any form of parallelism, called BCCSP+Res. Using the distinguishing valuation $*_m$, based on the notion of length, we were able to prove the completeness of the axioms of $\mathbf{P}$.

Then we extended our fragment with parallelism, called $\mathbf{P}_F$, in such a way that it only allows interleaving of actions. The distinguishing valuation was generalised from $*_m$ to $\diamond_w$, basing it on the branching degree of processes rather than the length. Using this valuation, we were able to prove the completeness of the axioms of $\mathbf{P}_F$.

Furthermore, we have extended the fragment, called $\mathbf{P}_H$, to allow CCS-style communication. We have adapted the conditional alphabet axioms of Baeten et al. to our purely equational setting with open terms. Unfortunately, this led to a number of problems. Firstly, because restriction does not distribute over $\between$ or $|$, there is a large amount of normal forms to consider. Secondly, it is hard to distinguish between these normal forms. We have proposed a new distinguishing valuation $\sharp_w$, based both on length and on branching degree, that can distinguish among some of the normals but not all.

Future work could look into the problems concerning the last mentioned fragment and the proposed $\sharp_w$ valuation. Once solved, it should be easy to generalise the theory with CCS-style communication to ACP-style communication, taking into account that the introduced CCS-specific alphabet axioms no longer apply. Other fragments that can be considered could contain other operators such as sequential composition $(\cdot)$, the empty process ϵ, or renaming and abstraction.

References

[1] Luca Aceto, Wan Fokkink, Anna Ingolfsdottir, and Bas Luttik. Finite equational bases in process algebra: results and open questions. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk, and R.C. de Vrijer, editors, *Processes, terms and cycles: steps on the road to infinity*, volume 3838 of *Lecture Notes in Computer Science*, pages 338–367. Springer, 2005.

[2] Luca Aceto, Wan Fokkink, Anna Ingolfsdottir, and Bas Luttik. A finite equational base for CCS with left merge and communication merge. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Proceedings of ICALP 2006*, volume 2747 of *Lecture Notes in Computer Science*, pages 492–503. Springer, 2006.

[3] J.C.M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2):131–146, 2005.

[4] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Conditional axioms and α/β-calcul in process algebra. In M. Wirsing, editor, *Proceedings of the IFIP TC2/WG2.2 Working Conference on Formal Description of Programming Concepts - III*, pages 53–75. North-Holland, 1985.

[5] J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information and Control*, 60(1-3):109–137, 1984.

[6] G. Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society*, 31:433–454, 1935.

[7] S. Christensen, Y. Hirshfeld, and F. Moller. Decidable subsets of CCS. *The Computer Journal*, 37(4):233–242, 1994.

[8] R. de Simone. Higher level synchronization devices in SCCS-Meije. *Theoretical Computer Science*, 37:245–267, 1985.

[9] J.F. Groote. A new strategy for proving ω-completeness applied to process algebra. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 314–331. Springer, 1990.

[10] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM (JACM)*, 32(1):137–161, 1985.

[11] R. Milner. *A Calculus of Communicating Systems*. Springer, 1980.

[12] R. Milner. *Communication and concurrency.* Prentice-Hall, 1989.

[13] F. Moller. *Axioms for Concurrency.* PhD thesis, 1989.

[14] F. Moller. The nonexistence of finite axiomatisations for CCS congruences. In *Proceedings of LICS'90*, pages 142–153. IEEE Computer Society Press, 1990.

[15] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

Appendix A

Equational Bases

A.1 The Equational Base of P

(A1)	$x + y$	$\approx y + x$	
(A2)	$(x + y) + z$	$\approx x + (y + z)$	
(A3)	$x + x$	$\approx x$	
(A4)	$x + \mathbf{0}$	$\approx x$	
(D1)	$\mathbf{0} \setminus H$	$\approx \mathbf{0}$	
(D2)	$a.x \setminus H$	$\approx \mathbf{0}$	if $a \in H$
(D3)	$a.x \setminus H$	$\approx a.(x \setminus H)$	otherwise
(D4)	$(x + y) \setminus H$	$\approx x \setminus H + y \setminus H$	
(DX1)	$x \setminus \emptyset$	$\approx x$	
(DX2)	$x \setminus \mathcal{A}$	$\approx \mathbf{0}$	
(DX3)	$(x \setminus H) \setminus J$	$\approx x \setminus (H \cup J)$	

Table A.1: The axioms of **P**

A.2 The Equational Base of $\mathbf{P}_F$

(A1)	$x + y$	$\approx y + x$	
(A2)	$(x + y) + z$	$\approx x + (y + z)$	
(A3)	$x + x$	$\approx x$	
(A4)	$x + \mathbf{0}$	$\approx x$	
(D1)	$\mathbf{0} \setminus H$	$\approx \mathbf{0}$	
(D2)	$a.x \setminus H$	$\approx \mathbf{0}$	if $a \in H$
(D3)	$a.x \setminus H$	$\approx a.(x \setminus H)$	otherwise
(D4)	$(x + y) \setminus H$	$\approx x \setminus H + y \setminus H$	
(D5)	$(x \between y) \setminus H$	$\approx x \setminus H \between y \setminus H$	
(DX1)	$x \setminus \emptyset$	$\approx x$	
(DX2)	$x \setminus \mathcal{A}$	$\approx \mathbf{0}$	
(DX3)	$(x \setminus H) \setminus J$	$\approx x \setminus (H \cup J)$	
(L1)	$\mathbf{0} \between x$	$\approx \mathbf{0}$	
(L2)	$a.x \between y$	$\approx a.(x \parallel y)$	
(L3)	$(x + y) \between z$	$\approx x \between z + y \between z$	
(L4)	$(x \between y) \between z$	$\approx x \between (y \parallel z)$	
(L5)	$x \between \mathbf{0}$	$\approx x$	
(M)	$x \parallel y$	$\approx x \between y + y \between x + x \mid y$	
(F)	$x \mid y$	$\approx \mathbf{0}$	

Table A.2: The axioms of $\mathbf{P}_F$

A.3 The Proposed Equational Base of $\mathbf{P}_H$

(A1)	$x + y$	$\approx y + x$	
(A2)	$(x + y) + z$	$\approx x + (y + z)$	
(A3)	$x + x$	$\approx x$	
(A4)	$x + \mathbf{0}$	$\approx x$	
(D1)	$\mathbf{0} \setminus H$	$\approx \mathbf{0}$	
(D2)	$a.x \setminus H$	$\approx \mathbf{0}$	if $a, \overline{a} \in H$
(D3)	$a.x \setminus H$	$\approx a.(x \setminus H)$	otherwise
(D4)	$(x + y) \setminus H$	$\approx x \setminus H + y \setminus H$	
(DX1)	$x \setminus \emptyset$	$\approx x$	
(DX3)	$(x \setminus H) \setminus J$	$\approx x \setminus (H \cup J)$	
(L1)	$\mathbf{0} \between x$	$\approx \mathbf{0}$	
(L2)	$a.x \between y$	$\approx a.(x \parallel y)$	
(L3)	$(x + y) \between z$	$\approx x \between z + y \between z$	
(L4)	$(x \between y) \between z$	$\approx x \between (y \parallel z)$	
(L5)	$x \between \mathbf{0}$	$\approx x$	
(DL1)	$(x \between (y \setminus H)) \setminus H$	$\approx (x \setminus H) \between (y \setminus H)$	
(DL2)	$((x \setminus H) \between y) \setminus H$	$\approx (x \setminus H) \between (y \setminus H)$	
(C1)	$\mathbf{0} \mid x$	$\approx \mathbf{0}$	
(C2)	$a.x \mid b.y$	$\approx \tau.(x \parallel y)$	if $b = \overline{a}$
(C3)	$a.x \mid b.y$	$\approx \mathbf{0}$	otherwise
(C4)	$(x + y) \mid z$	$\approx x \mid z + y \mid z$	
(C5)	$x \mid y$	$\approx y \mid x$	
(C6)	$(x \mid y) \mid z$	$\approx x \mid (y \mid z)$	
(C7)	$(x \between y) \mid z$	$\approx (x \mid z) \between y$	
(DC1)	$(x \mid (y \setminus H)) \setminus H$	$\approx (x \setminus H) \mid (y \setminus H)$	
(M)	$x \parallel y$	$\approx x \between y + y \between x + x \mid y$	
(H)	$x \mid (y \mid z)$	$\approx \mathbf{0}$	

Table A.3: The axioms of $\mathbf{P}_H$