Finite Equational Bases for Fragments of CCS with Restriction and Relabelling*

Luca Aceto[1], Anna Ingólfsdóttir[1], Bas Luttik[2], and Paul van Tilburg[2]

[1] School of Computer Science, Reykjavík University, Kringlan 1, 103 Reykjavík, Iceland, {luca,annai}@ru.is
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, {s.p.luttik,p.j.a.v.tilburg}@tue.nl

Abstract We investigate the equational theory of several fragments of CCS modulo (strong) bisimilarity with special attention to restriction and relabelling. The largest fragment we consider includes action prefixing, choice, parallel composition without communication, restriction and relabelling. We present a finite equational base (i.e., a finite ground-complete and omega-complete axiomatisation) for it, including the left merge from ACP as auxiliary operation to facilitate the axiomatisation of parallel composition.

1 Introduction

The Calculus of Communicating Systems (CCS) was developed by Robin Milner in the late 1970s [8]. This calculus introduced a formal language for describing processes, using a transition system to give an operational meaning to the expressions in the language. In this paper we pay special attention to the restriction and relabelling operators of CCS.

The restriction operator takes a process and a set of actions as arguments. It delimits the scope of actions by preventing the execution by the process of the actions in the set. Restriction is often used to specify the communication topology of a system by blocking the execution of interleaving actions of parallel processes so that only the result of (synchronous) communication remains. Restriction is also present in ACP [3], where it is called encapsulation.

The relabelling operator takes a process and a function from actions to actions. It renames the actions in the process according to the function, and can be used to instantiate a generic specification for specific needs. In CCS, relabelling is, e.g., used in defining the so-called linking operation, which is at the core of many of the specifications offered in [9]. Relabelling is not present in ACP, but it can be added and then it increases the expressiveness of the language. Namely, Baeten and Bergstra prove in [2] that the process Queue cannot be specified by means of a finite guarded recur-

* A full version of this paper, including omitted proofs and auxilary lemmas is available as CS-Report 08-08, Eindhoven University of Technology.

sive specification over ACP, whereas it can be specified by means of a finite guarded recursive specification over ACP with renaming.

In [6] (see also [9]), Hennessy and Milner propose an axiomatisation for CCS modulo bisimilarity that they prove ground-complete (i.e., all valid equations involving terms without variables are equationally derivable from it). Their axiomatisation is infinite, which is unavoidable as proved by Moller [11]. For a finite axiomatisation it is necessary to add auxiliary operators, e.g., the left merge and communication merge of ACP [3].

We want to give an equational base (i.e., an axiomatisation that is not just ground-complete but complete also for equations involving terms with variables) for CCS modulo bisimilarity. Perhaps surprisingly, no complete axiomatisations of bisimilarity over languages including restriction and relabelling have been given to date. In [7], Milner studied an algebra of flowgraphs with operations of (parallel) composition, restriction and relabelling, and provided a complete axiomatisation for it. In that reference, however, the notion of equivalence between expressions is purely "structural", since two expressions are equated when they denote the same flowgraph up to isomorphism.

In this paper we present finite equational bases for fragments of CCS modulo bisimilarity that include restriction and relabelling operators. The largest fragment we consider here includes all the operators from recursion-free CCS, but the parallel composition operator is limited to pure interleaving and does not allow for synchronisation between parallel components. Our completeness proofs build on results and techniques developed in [1], where a finite axiomatisation for the fragment of CCS without restriction and relabelling operators is proved complete.

For our completeness proofs we adopt the classic normal form strategy. This entails showing that all process terms can be proved equal to some normal form using the axioms, followed by the construction of a distinguishing valuation that ensures that two normal forms are equal under this valuation only if they can be proved equal. Both the above-mentioned steps involve non-trivial extensions of the techniques from [1] for the languages we consider because, unlike for ground-complete axiomatisations, restriction and relabelling cannot be eliminated from terms. This means that normal forms may contain occurrences of these operations, and their form will be more complicated than that considered thus far in the literature. Moreover, in order to implement the latter step in the above-mentioned proof technique, distinguishing valuations will need to be defined in such a way that they allow us to detect the restrictions and relabellings that occur in the normal forms.

For the shape of the normal forms in the present paper it is crucial that restriction and relabelling distribute over parallel composition. This is the reason that we now only consider an operator for parallel composition that is limited to pure interleaving; neither restriction nor relabelling distribute over parallel composition in the presence of synchronisation. So an obvious avenue for future work is the technically challenging problem of giving a complete axiomatisation of full CCS modulo bisimilarity, with restriction, relabelling and parallel composition that allows for synchronisation.

The paper is organised as follows. In Sect. 2 we introduce the fragments of CCS that will be discussed in this paper. In Sects. 3–5 we propose equational bases for three fragments of CCS: first only with the restriction operator, then only with the relabelling operator, and finally with both operators.

2 Preliminaries

In this section we introduce a process calculus that is obtained from Milner's pure CCS [9] by omitting recursion, replacing parallel composition by an operation for pure interleaving (i.e., which does not include synchronisation between components), and adding the left merge of Bergstra and Klop [3]. The calculus gives rise to a process algebra **P** for which we will present a (finite) axiomatisation. The main result of this paper states that this axiomatisation is complete.

We fix a set of *action labels* $\mathscr{L}$, a set of *co-action labels* $\overline{\mathscr{L}}$ disjoint from $\mathscr{L}$ and a bijection $\bar{\cdot} : \mathscr{L} \to \overline{\mathscr{L}}$. We define the set of *actions* $\mathscr{A}$ as $\mathscr{L} \cup \overline{\mathscr{L}}$. The inverse of $\bar{\cdot}$ we shall also denote by $\bar{\cdot}$, and thus $\overline{\overline{a}} = a$ for each $a \in \mathscr{A}$. In [9], Milner assumes that $\mathscr{L}$ and $\overline{\mathscr{L}}$ are infinite. However, to obtain a finite axiomatisation, we need to require that the sets $\mathscr{L}$ and $\overline{\mathscr{L}}$ are finite. We also fix a countably infinite set of *variables* $\mathscr{V}$. The meta-variables a, b, and c generally range over $\mathscr{A}$; x, y, and z range over $\mathscr{V}$.

A *relabelling function* is a function $f : \mathscr{A} \to \mathscr{A}$ such that $f(\overline{a}) = \overline{f(a)}$ for each $a \in \mathscr{A}$. With every relabelling function $f : \mathscr{A} \to \mathscr{A}$ we associate a function $f^{-1} : \mathscr{P}(\mathscr{A}) \to \mathscr{P}(\mathscr{A})$ such that $f^{-1}(\mathscr{A}') = \{a \mid f(a) \in \mathscr{A}'\}$ for each $\mathscr{A}' \subseteq \mathscr{A}$. The identity relabelling function Id is defined by $Id(a) = a$ for each $a \in \mathscr{A}$. For each relabelling function f and $L \subseteq \mathscr{L}$, we write $f \restriction L$ for the relabelling function defined by

$$(f \restriction L)(a) = \begin{cases} f(a) & \text{if } a \in L \text{ or } \overline{a} \in L, \\ a & \text{otherwise.} \end{cases}$$

The meta-variables f and g generally refer to relabelling functions, and K and L refer to subsets of $\mathscr{L}$.

The set of *process terms* $\mathscr{T}_{\backslash,[]}$ is generated by the following grammar:

$$\mathrm{T} ::= \mathbf{0} \mid x \mid a.\mathrm{T} \mid \mathrm{T}+\mathrm{T} \mid \mathrm{T} \parallel \mathrm{T} \mid \mathrm{T} \mathbin{\|\!\lfloor} \mathrm{T} \mid \mathrm{T} \backslash L \mid \mathrm{T}[f]$$

where $a \in \mathscr{A}$, $x \in \mathscr{V}$, $L \subseteq \mathscr{L}$, and $f : \mathscr{A} \to \mathscr{A}$ is a relabelling function. The meta-variables p, q and r generally range over $\mathscr{T}_{\backslash,[]}$. We use the following convention for the binding power of the operators in decreasing order: relabelling $\_[f]$ and restriction $\_\backslash L$ (tightest binding), prefixing $a.\_$, parallel composition $\_ \parallel \_$ and left merge $\_ \mathbin{\|\!\lfloor} \_$, alternative composition $\_+\_$. In the remainder of the paper we also need notation for the following subsets of $\mathscr{T}_{\backslash,[]}$: we use $\mathscr{T}_{\backslash}$ to denote the set of all process terms without occurrences of relabelling operators, and $\mathscr{T}_{[]}$ to denote the set of all process terms without occurrences of restriction operators.

Process terms that do not contain any variables are called *closed*. The set of closed process terms is denoted by $\mathscr{T}^{C}_{\backslash,[]}$. We give an operational semantics to closed terms using the binary relations $\xrightarrow{a}$ ($a \in \mathscr{A}$) on $\mathcal{T}^{C}_{\setminus,[]}$ defined by means of the specification in Table 2.1.

$$1\,\frac{}{a.p \xrightarrow{a} p} \qquad 2\,\frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'} \qquad 3\,\frac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$$

$$4\,\frac{p \xrightarrow{a} p'}{p \parallel q \xrightarrow{a} p' \parallel q} \qquad 5\,\frac{q \xrightarrow{a} q'}{p \parallel q \xrightarrow{a} p \parallel q'} \qquad 6\,\frac{p \xrightarrow{a} p'}{p \mathbin{\|\!\!\lfloor} q \xrightarrow{a} p' \parallel q}$$

$$7\,\frac{p \xrightarrow{a} p' \quad a,\bar{a} \notin L}{p \setminus L \xrightarrow{a} p' \setminus L} \qquad 8\,\frac{p \xrightarrow{a} p'}{p[f] \xrightarrow{f(a)} p'[f]}$$

Table 2.1: Operational semantics

If $p \xrightarrow{a} p'$ for some $a \in \mathscr{A}$, then we call p' a *residual* of p. If for a term p and an action a there does not exist a term p' such that $p \xrightarrow{a} p'$, then we write $p \not\xrightarrow{a}$.

It is technically convenient to extend the usage of the rules in Table 2.1 by letting them define binary relations $\xrightarrow{a}$ ($a \in \mathscr{A}$) on the full set of terms $\mathcal{T}_{\setminus,[]}$. (Since there are no operational rules for variables, this effectively means that variables are assigned the "same behaviour" as $\mathbf{0}$.)

The *depth* $\mathrm{d}(p)$ can then be defined for all process terms $p \in \mathcal{T}_{\setminus,[]}$ as the maximum number of consecutive transitions that can be performed starting from p, i.e.,

$$\mathrm{d}(p) = \max\{n \mid \exists_{p_1,\ldots,p_n \in \mathcal{T}_{\setminus,[]}} \text{ s.t. } p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} p_n\}.$$

The operational semantics assigns behaviour to closed terms. The notion of bisimilarity [12] relates closed process terms that exhibit equal behaviour.

Definition 1. A *bisimulation* is a symmetric binary relation $\mathscr{R}$ on $\mathcal{T}^{C}_{\setminus,[]}$ such that $p\,\mathscr{R}\,q$ implies

if $p \xrightarrow{a} p'$, then there exists some $q' \in \mathcal{T}^{C}_{\setminus,[]}$ such that $q \xrightarrow{a} q'$ and $p'\,\mathscr{R}\,q'$.

Closed process terms $p, q \in \mathcal{T}^{C}_{\setminus,[]}$ are said to be *bisimilar* (notation: $p \leftrightarroweq q$) if a bisimulation relation $\mathscr{R}$ exists such that $p\,\mathscr{R}\,q$.

It is well-known that $\leftrightarroweq$ is an equivalence relation. We denote by $[p]$ the *equivalence class* of a closed process term $p \in \mathcal{T}^{C}_{\setminus,[]}$ modulo bisimilarity, and by $\mathcal{T}^{C}_{\setminus,[]}/\!\leftrightarroweq$ the set of all such equivalence classes. The rules in Table 2.1 are all in de Simone's format [13], and from this it follows that bisimilarity is compatible with the syntactic constructs of our process calculus. So $\mathcal{T}^{C}_{\setminus,[]}/\!\leftrightarroweq$ is the universe of a process algebra with a distinguished element $\mathbf{0}$, unary operators $a.\_$ (for all $a \in \mathscr{A}$), $\_[f]$ (for all relabelling functions $f : \mathscr{A} \to \mathscr{A}$), and $\_\setminus L$ (for all $L \subseteq \mathscr{L}$), and binary operators $\_+\_$, $\_\parallel\_$ and $\_\mathbin{\|\!\!\lfloor}\_$ defined as follows:

$$\begin{array}{lll}
\mathbf{0} = [\mathbf{0}]\ , & [p] \parallel [q] = [p \parallel q]\ , & [p] \setminus L = [p \setminus L]\ , \\
a.[p] = [a.p]\ , & [p] \mathbin{\|\!\!\lfloor} [q] = [p \mathbin{\|\!\!\lfloor} q]\ , & [p][f] = [p[f]]\ , \\
[p] + [q] = [p+q]\ . & &
\end{array}$$

Henceforth we shall denote this process algebra by **P**. Members of **P** are called *processes* and will be ranged over by p, q and r like process terms. This convention will not lead to confusion because it will be clear from the context which is meant.

To be able to reason syntactically about **P**, we define how process terms can be used to denote elements of **P** and present an inference system for the derivation of equations between process terms that are valid in **P**.

Definition 2. A *valuation* is a mapping $v : \mathscr{V} \to \mathbf{P}$. Such a mapping may be applied to process terms in $\mathscr{T}_{\backslash,[]}$ using the *evaluation mapping* $[\![\cdot]\!]_v : \mathscr{T}_{\backslash,[]} \to \mathbf{P}$ defined inductively by:

$$\begin{array}{lll} [\![\mathbf{0}]\!]_v = \mathbf{0}\ , & [\![q+r]\!]_v = [\![q]\!]_v + [\![r]\!]_v\ , & [\![q \backslash L]\!]_v = [\![q]\!]_v \backslash L\ , \\ [\![x]\!]_v = v(x)\ , & [\![q \parallel r]\!]_v = [\![q]\!]_v \parallel [\![r]\!]_v\ , & [\![q[f]]\!]_v = [\![q]\!]_v[f]\ , \\ [\![a.q]\!]_v = a.[\![q]\!]_v\ , & [\![q \mathbin{\lfloor\!\lfloor} r]\!]_v = [\![q]\!]_v \mathbin{\lfloor\!\lfloor} [\![r]\!]_v\ . & \end{array}$$

Note that the evaluation mapping maps process terms to members of the algebra **P**, given an assignment of processes to variables. When an evaluation mapping is applied to a closed process term, the assignment is irrelevant and the evaluation mapping amounts to interpreting the syntactic constructs as the corresponding operations of the algebra. Thus, without fixing a specific evaluation mapping, we can use a closed term to denote an element of **P**; this element of **P** is then, of course, the equivalence class that contains the particular closed term. For example, the closed term $a.\mathbf{0}+b.\mathbf{0}$ denotes the element $[\![a.\mathbf{0}+b.\mathbf{0}]\!]_v$ of **P**.

A *process equation* is a pair of process terms (p,q) written as $p \approx q$. The equation $p \approx q$ is *valid* in **P** if $[\![p]\!]_v = [\![q]\!]_v$ for all valuations $v : \mathscr{V} \to \mathbf{P}$. Henceforth, we write $p \underline{\leftrightarrow} q$ if $p \approx q$ is valid in **P**.

(A1)	$x+y \approx y+x$	(LM1)	$x \mathbin{\lfloor\!\lfloor} \mathbf{0} \approx x$
(A2)	$(x+y)+z \approx x+(y+z)$	(LM2)	$\mathbf{0} \mathbin{\lfloor\!\lfloor} x \approx \mathbf{0}$
(A3)	$x+x \approx x$	(LM3)	$a.x \mathbin{\lfloor\!\lfloor} y \approx a.(x \parallel y)$
(A4)	$x+\mathbf{0} \approx x$	(LM4)	$(x+y) \mathbin{\lfloor\!\lfloor} z \approx x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$
		(LM5)	$(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z \approx x \mathbin{\lfloor\!\lfloor} (y \parallel z)$
		(M)	$x \parallel y \approx x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x$
(RS1a)	$x \backslash \emptyset \approx x$	(RL1)	$x[Id] \approx x$
(RS1b)	$x \backslash \mathscr{L} \approx \mathbf{0}$	(RL2)	$\mathbf{0}[f] \approx \mathbf{0}$
(RS2)	$\mathbf{0} \backslash L \approx \mathbf{0}$	(RL3)	$(a.x)[f] \approx f(a).(x[f])$
(RS3)	$a.x \backslash L \approx \begin{cases} \mathbf{0} & \text{if } a,\overline{a} \in L \\ a.(x \backslash L) & \text{if } a,\overline{a} \notin L \end{cases}$	(RL4)	$(x+y)[f] \approx x[f]+y[f]$
		(RL5)	$(x \mathbin{\lfloor\!\lfloor} y)[f] \approx x[f] \mathbin{\lfloor\!\lfloor} y[f]$
(RS4)	$(x+y) \backslash L \approx x \backslash L + y \backslash L$	(RL6)	$(x[f])[g] \approx x[g \circ f]$
(RS5)	$(x \mathbin{\lfloor\!\lfloor} y) \backslash L \approx x \backslash L \mathbin{\lfloor\!\lfloor} y \backslash L$		
(RS6)	$(x \backslash L) \backslash K \approx x \backslash (L \cup K)$		
(RR1)	$x[f] \backslash L \approx (x \backslash f^{-1}(L))[f]$		
(RR2)	$(x \backslash L)[f] \approx (x \backslash L)[g]$ if $f \restriction (\mathscr{L}-L) = g \restriction (\mathscr{L}-L)$		

Table 2.2: The set of axioms $\mathscr{E}$

Table 2.2 presents a set of process equations $\mathscr{E}$ that are all well-known to be valid in $\mathbf{P}$ (see, e.g., [6, 9, 5, 3]). We shall use the process equations in $\mathscr{E}$ as the axioms of an inference system with as rules the familiar rules of equational logic [4]. Henceforth, whenever we write $p \approx q$ we mean that the process equation $p \approx q$ is derivable within this inference system. (In the cases in which we intend to highlight that only a proper subset of the axioms in $\mathscr{E}$ is needed to derive $p \approx q$, we shall explicitly mention the needed axioms.)

Since the axioms are valid in $\mathbf{P}$ and the rules of equational logic preserve validity, we have the following soundness result.

Proposition 1 (Soundness). *For all process terms $p,q \in \mathscr{T}_{\backslash,[]}$, if $p \approx q$, then $p \leftrightarrows q$.*

The main goal of this paper is to prove that the inference system is also complete, i.e., that, for all process terms $p,q \in \mathscr{T}_{\backslash,[]}$, if $p \leftrightarrows q$ then $p \approx q$; if this is the case, then it follows that $\mathscr{E}$ is an *equational base* for the algebra $\mathbf{P}$. Our completeness proof proceeds according to the following strategy:

1. Identify an appropriate notion of normal form and prove that every term in $\mathscr{T}_{\backslash,[]}$ is rewritable according to the axioms in $\mathscr{E}$ to a normal form. To establish completeness, it is then enough to prove that $s \leftrightarrows t$ implies $s \approx t$ for all normal forms s and t.
2. Associate with every two normal forms s and t a distinguishing valuation, i.e., a valuation $* : \mathscr{V} \to \mathbf{P}$ such that if $s \not\approx t$, then $[\![s]\!]_* \neq [\![t]\!]_*$. From this it follows that $s \leftrightarrows t$ implies $s \approx t$ for all normal forms s and t.

The first step is fairly straightforward, even though the normal forms we need to consider involve all the operations in the calculus; the crux of our completeness proof is to find a suitable distinguishing valuation and prove the property described in the second step. Our distinguishing valuation combines several ideas that are best explained separately. To this end, we shall, as stepping stones towards our main result, first apply the aforementioned strategy to obtain completeness results for the fragments $\mathscr{T}_{\backslash}$ and $\mathscr{T}_{[]}$ of our calculus. In Sect. 3 we consider the fragment without relabelling. In Sect. 4 we study the fragment without restriction. Finally, in Sect. 5 we consider the full calculus.

We use the summation $\sum_{i\in I} p_i$ (modulo A1, A2 and A4) to denote an alternative composition of the form $p_1 + p_2 + \ldots$ for a finite set I and processes p_i $(i \in I)$. We also define $\mathbf{0} = \sum_{i\in\emptyset} p_i$ for the empty index set. Furthermore, we shall use an abbreviation for iterated prefixing, defining $a^0.\mathbf{0} = \mathbf{0}$ and $a^{i+1}.\mathbf{0} = a.(a^i.\mathbf{0})$.

We conclude this section with a few properties pertaining to the algebra $\mathbf{P}$ that we shall need in the rest of the paper.

The binary relations $\xrightarrow{a}$ $(a \in \mathscr{A})$ defined earlier for $\mathscr{T}^C_{\backslash,[]}$ induce binary relations $\xrightarrow{a}$ $(a \in \mathscr{A})$ on $\mathbf{P}$ as follows: for all $p, p' \in \mathscr{T}^C_{\backslash,[]}$ we define that $[p] \xrightarrow{a} [p']$ iff for all $q \in [p]$ there exists a $q' \in [p']$ such that $q \xrightarrow{a} q'$.

Proposition 2. *For all $p,q,r \in \mathbf{P}$*

1. *$p = \mathbf{0}$ iff $p \not\xrightarrow{a}$ for all $a \in \mathscr{A}$;*
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. $p \mathbin{\|\!\lfloor} q \xrightarrow{a} r$ *iff there exists some* $p' \in \mathbf{P}$ *such that* $p \xrightarrow{a} p'$ *and* $r = p' \parallel q$*;*
5. $p \parallel q \xrightarrow{a} r$ *iff* $p \mathbin{\|\!\lfloor} q \xrightarrow{a} r$ *or* $q \mathbin{\|\!\lfloor} p \xrightarrow{a} r$*;*
6. $p \setminus L \xrightarrow{a} r$ *iff* $a, \overline{a} \notin L$ *and there exists some* $q \in \mathbf{P}$ *such that* $p \xrightarrow{a} q$ *and* $r = q \setminus L$*;*
7. $p[f] \xrightarrow{b} r$ *iff there exist some* $a \in \mathscr{A}$ *and* $q \in \mathbf{P}$ *such that* $f(a) = b$*,* $p \xrightarrow{a} q$ *and* $r = q[f]$*.*

Bisimulation equivalence preserves the notion of depth (i.e., the closed process terms in an equivalence class have the same depth). Therefore we can define the depth $\mathrm{d}(p)$ of a process $p \in \mathbf{P}$ as the depth of any of its members. As a technical tool we shall also need the notion of *branching degree* $\mathrm{b}(p)$ of a process $p \in \mathbf{P}$ defined by

$$\mathrm{b}(p) = |\{(a, p') \mid p \xrightarrow{a} p'\}|.$$

Lemma 1. *For all* $p, q \in \mathbf{P}$*, it holds that*

1. $\mathrm{b}(\mathbf{0}) = 0$*;*
2. $\mathrm{b}(a.p) = 1$*;*
3. $\mathrm{b}(p+q) \leq \mathrm{b}(p) + \mathrm{b}(q)$*;*
4. $\mathrm{b}(p \mathbin{\|\!\lfloor} q) = \mathrm{b}(p)$*;*
5. $\mathrm{b}(p \parallel q) \geq \mathrm{b}(p)$ *and* $\mathrm{b}(p \parallel q) \geq \mathrm{b}(q)$*.*

An element $p \in \mathbf{P}$ 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 $p = p_1 \parallel \cdots \parallel p_n$. The following theorem and corollary are proved in [10].

Theorem 1. *Every element of* $\mathbf{P}$ *has a unique parallel decomposition.*

Corollary 1. *Let* $p, q, r \in \mathbf{P}$*. If* $p \parallel q = p \parallel r$*, then* $q = r$*.*

3 Restriction

In this section we establish a completeness result for the fragment of our process calculus that includes the restriction operators, but excludes relabelling operators.

The set of normal forms $\mathscr{N}_{\setminus}$ is generated by the following grammar:

$$\mathrm{N} ::= \mathbf{0} \mid a.\mathrm{N} \mid (x \setminus L) \mathbin{\|\!\lfloor} \mathrm{N} \mid \mathrm{N} + \mathrm{N}$$

where $a \in \mathscr{A}$, $x \in \mathscr{V}$, and $L \subset \mathscr{L}$. We refer to $a.s$ and $(x \setminus L) \mathbin{\|\!\lfloor} s$ as simple normal forms.

Lemma 2. *Every process term* $p \in \mathscr{T}_{\setminus}$ *has a normal form* $s \in \mathscr{N}_{\setminus}$ *such that* $p \approx s$ *is provable using RS1a–RS6, LM1–LM5, and M.*

Because of Lemma 2, each term can be written using the following general form:

$$\sum_{i \in I} a_i.s_i + \sum_{j \in J} (x_j \setminus L_j) \mathbin{\|\!\lfloor} s_j \quad \text{(modulo A1, A2 and A4)}$$

for finite index sets I, J and with $a_i \in \mathscr{A}$, $s_i, s_j \in \mathscr{N}_{\setminus}$, $x_j \in \mathscr{V}$, and $L_j \subset \mathscr{L}$.

For our completeness proof, we define a valuation that allows us to distinguish non-bisimilar normal forms. The definitions of the distinguishing valuations we use in this paper are geared towards achieving the properties stated in Lemmas 5 and 6 to follow (or similar lemmas in the subsequent sections). In particular, distinguishing valuations will allow us to tell apart the different types of simple normal forms (Lemma 5).

Definition 3. Let $w \geq 1$ and let $\lceil\cdot\rceil : \mathscr{V} \to (\mathbb{N} - \{0,1\})$ be an injective function. We define the valuation $\diamond_w$ for each variable $x \in \mathscr{V}$ by:

$$\diamond_w(x) = \sum_{a \in \mathscr{L}} a.\xi_{\lceil x \rceil \cdot w} \text{ with } \xi_i = \sum_{a \in \mathscr{L}} \sum_{j=1}^{i} a^i.\mathbf{0}.$$

Note that if s is a simple normal form, then $[\![s]\!]_{\diamond_w}$ has a unique residual. In the following lemmas we establish some special properties pertaining to the valuation $\diamond_w$ when applied to normal forms. These properties will be used to show that $\diamond_w$ is indeed a distinguishing valuation.

First we state two properties of the process $\xi_{\lceil x \rceil \cdot w} \setminus L$, which is a parallel component of the unique residual of $[\![(x \setminus L) \mathbin{|\!\lfloor} s]\!]_{\diamond_w}$.

Lemma 3. *For all $i \geq 1$ and $L \subset \mathscr{L}$, the process $\xi_i \setminus L$ is parallel prime, and its branching degree* $\mathrm{b}(\xi_i \setminus L)$ *is* $i \cdot |\mathscr{L} - L|$.

The valuation $\diamond_w$ is such that if the parameter w is greater than an estimated highest branching degree occurring already in s, then it is possible to determine from the process $[\![s]\!]_{\diamond_w}$ whether s has action prefixing or a left merge as head operator. This will be explained in Lemma 5 below; first we formalise an appropriate estimation of the highest branching degree occurring in a normal form s.

Definition 4. For all $s \in \mathscr{N}_{\setminus}$, the estimated highest branching degree $\mathrm{esb}(s)$ occurring in s is defined inductively as follows:

$$\begin{aligned} &\mathrm{esb}(\mathbf{0}) = 0, && \mathrm{esb}(s+t) = \mathrm{esb}(s) + \mathrm{esb}(t), \\ &\mathrm{esb}(a.t) = \max(1, \mathrm{esb}(t)), && \mathrm{esb}((x \setminus L) \mathbin{|\!\lfloor} t) = \max(|\mathscr{L} - L|, \mathrm{esb}(t)), \end{aligned}$$

with $a \in \mathscr{A}, x \in \mathscr{V}, L \subset \mathscr{L}$ and $t \in \mathscr{N}_{\setminus}$.

Note 1. The lower bound $|\mathscr{L} - L|$ in the definition of $\mathrm{esb}((x \setminus L) \mathbin{|\!\lfloor} t)$ follows from the definition of $\diamond_w$ (see Definition 3), since $[\![x \setminus L]\!]_{\diamond_w} \xrightarrow{a} \xi_{\lceil x \rceil \cdot w} \setminus L$ for all $a \in \mathscr{L} - L$.

The following lemma shows that the estimated branching degree of s is an upper bound on the branching degree of $[\![s]\!]_{\diamond_w}$.

Lemma 4. *For every normal form $s \in \mathscr{N}_{\setminus}$,* $\mathrm{b}([\![s]\!]_{\diamond_w}) \leq \mathrm{esb}(s)$.

Lemma 5. *Let $s, s' \in \mathscr{N}_{\setminus}$ with s simple, and let $x \in \mathscr{V}$, $L \subset \mathscr{L}$ and $w >$* $\mathrm{esb}(s)$.

1. If $s = a.s'$, then the branching degree of the unique residual of $[\![s]\!]_{\diamond_w}$ is smaller than w.

2. *If $s = (x \setminus L) \mathbin{|\!\lfloor} s'$, then the branching degree of the unique residual of $[\![s]\!]_{\diamond_w}$ is larger than w.*

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

1. *If $s = a.s'$, then $p = [\![s']\!]_{\diamond_w}$. Because $\mathrm{esb}(s) < w$, by Definition 4 $\mathrm{esb}(s') \leq \mathrm{esb}(s) < w$. Hence, by Lemma 4, the branching degree of $[\![s']\!]_{\diamond_w}$ is smaller than w.*
2. *If $s = (x \setminus L) \mathbin{|\!\lfloor} s'$, then $p = (\xi_{\lceil x \rceil \cdot w} \setminus L) \parallel [\![s']\!]_{\diamond_w}$. We have by Lemma 3 that $\mathrm{b}(\xi_{\lceil x \rceil \cdot w} \setminus L) = \lceil x \rceil \cdot w \cdot |\mathscr{L} - L| > w$ (given that $L \subset \mathscr{L}$ and $\lceil x \rceil > 1$). Because $[\![s']\!]_{\diamond_w}$ does not decrease the branching degree of the residual p (by Lemma 1), we may conclude that the residual p has a branching degree that exceeds w.* ☐

When it has been determined from the unique residual of $[\![s]\!]_{\diamond_w}$ that s has a left merge as head operator, then the following key lemma allows us to determine which variable occurs in its left argument, and by which proper subset of $\mathscr{L}$ this variable is restricted.

Lemma 6. *For $i, j \geq 1$ and $K, L \subset \mathscr{L}$, if $\xi_i \setminus K = \xi_j \setminus L$, then $K = L$ and $i = j$.*

Proof. We first show that $K = L$. Assume that $a \in \mathscr{L} - K$. By Definition 3 and Proposition 2(6) there exists some $r \in \mathbf{P}$ such that $\xi_i \setminus K \xrightarrow{a} r$. Therefore $\xi_j \setminus L \xrightarrow{a} r$ also holds. However, by Proposition 2(6) this also means that $a \in \mathscr{L} - L$. The case that $a \in \mathscr{L} - L$ is symmetrical. Hence, since $a \in \mathscr{L} - K$ iff $a \in \mathscr{L} - L$, it follows that $K = L$.

Because $K = L$ and $\xi_i \setminus K = \xi_j \setminus L$, we know that $\mathrm{b}(\xi_i \setminus K) = \mathrm{b}(\xi_j \setminus K)$ and therefore $i \cdot |\mathscr{L} - K| = j \cdot |\mathscr{L} - K|$ by Lemma 3. Since $K \subset \mathscr{L}$, it follows that $i = j$. ☐

The following result states that the valuation $\diamond_w$ is indeed distinguishing.

Theorem 2. *For every two normal forms $s, t \in \mathscr{N}_{\setminus}$ with $w > \mathrm{esb}(s), \mathrm{esb}(t)$, it holds that if $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$, then $s \approx t$ modulo A1–A4.*

Proof. Assume that $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$ holds; we prove that $s \approx t$ is derivable using A1–A4 by induction on the sum of the depths of s and t. We do this by showing that for every summand s_i of s there exists a summand t_j of t 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 must also be a summand t_j of t such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s_i']\!]_{\diamond_w}$. By Lemma 5 we know that t_j must have the form $b.t_j'$, because the branching degree of the unique residual of $[\![t_j]\!]_{\diamond_w}$ does not exceed w.*
   Given that t_j has this form, it can only perform one transition: $[\![t_j]\!]_{\diamond_w} \xrightarrow{b} [\![t_j']\!]_{\diamond_w}$. Since also $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s_i']\!]_{\diamond_w}$ it 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 $s_i = (x \setminus K) \mathbin{⌊\!⌊} s'_i$, then, since $K \subset \mathscr{L}$, $[\![s_i]\!]_{\diamond_w} \xrightarrow{a} p$ for some $a \in \mathscr{L} - K$. We know that also a summand t_j of t exists such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} p$. Definition 3 gives us that $p = (\xi_{\lceil x \rceil \cdot w} \setminus K) \parallel [\![s'_i]\!]_{\diamond_w}$. Similarly to the previous case, by Lemma 5 we also know that t_j must have the form $(y \setminus L) \mathbin{⌊\!⌊} t'_j$ for some $y \in \mathscr{V}$ and $L \subset \mathscr{L}$. The residual of t_j after performing an action $a \in \mathscr{L} - L$ is $(\xi_{\lceil y \rceil \cdot w} \setminus L) \parallel [\![t'_j]\!]_{\diamond_w}$ (also by Definition 3). This residual is equal to p, so we know that $(\xi_{\lceil x \rceil \cdot w} \setminus K) \parallel [\![s'_i]\!]_{\diamond_w} = (\xi_{\lceil y \rceil \cdot w} \setminus L) \parallel [\![t'_j]\!]_{\diamond_w}$. By Lemma 3 we have that the process $\xi_{\lceil x \rceil \cdot w} \setminus K$ is parallel prime and has a branching degree that exceeds w. This process cannot occur in the unique parallel decomposition of $[\![t'_j]\!]_{\diamond_w}$ because, by Lemmas 1 and 4, and the assumption of the theorem that $w > \mathrm{esb}(t)$, the branching degrees of all processes in the decomposition of $[\![t'_j]\!]_{\diamond_w}$ do not exceed w. Conversely, this also holds in a symmetric way for the process $\xi_{\lceil y \rceil \cdot w} \setminus L$ with respect to the unique parallel decomposition of $[\![s'_i]\!]_{\diamond_w}$. Hence, $\xi_{\lceil x \rceil \cdot w} \setminus K = \xi_{\lceil y \rceil \cdot w} \setminus L$.*
   From $\xi_{\lceil x \rceil \cdot w} \setminus K = \xi_{\lceil y \rceil \cdot w} \setminus L$ it follows by Lemma 6 that $K = L$ and $\lceil x \rceil \cdot w = \lceil y \rceil \cdot w$. Therefore, $x = y$ by injectivity of $\lceil \cdot \rceil$.
   We have established that $K = L$ and $x = y$, so $(\xi_{\lceil x \rceil \cdot w} \setminus K) \parallel [\![s'_i]\!]_{\diamond_w} = (\xi_{\lceil y \rceil \cdot w} \setminus L) \parallel [\![t'_j]\!]_{\diamond_w} = (\xi_{\lceil x \rceil \cdot w} \setminus K) \parallel [\![t'_j]\!]_{\diamond_w}$, and hence, by Corollary 1, $[\![s'_i]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$. By induction hypothesis it follows that $s'_i \approx t'_j$ modulo A1–A4, so we may conclude that $s_i = (x \setminus K) \mathbin{⌊\!⌊} s'_i \approx (y \setminus L) \mathbin{⌊\!⌊} t'_j = t_j$ modulo A1–A4.

It follows by a symmetric argument that every summand of t is also provably equal to a summand of s using the above mentioned equations. Hence, $s \approx s + t \approx t$ modulo A1–A4. □

Corollary 2. *For all $p, q \in \mathscr{T}_\setminus$ it holds that $p \approx q$ is provable using A1–A4, RS1a–RS6, LM1–LM5, and M if, and only if, $p \leftrightarroweq q$.*

Proof. The implication from left to right follows from Proposition 1.

For the proof of the implication from the right to the left, we assume that $p \leftrightarroweq q$. By Lemma 2, there are two normal forms s and t such that the equations $p \approx s$ and $q \approx t$ are provable using RS1a–RS6, LM1–LM5, and M. If $p \leftrightarroweq q$, then by Proposition 1 and transitivity of $\leftrightarroweq$ we also know that $s \leftrightarroweq t$ and thus $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$. Hence, by Theorem 2 we know that $s \approx t$ is provable using A1–A4 and we can conclude that $p \approx s \approx t \approx q$. □

4 Relabelling

In this section we establish a completeness result for the fragment of our process calculus that includes relabelling operators, but excludes restriction operators.

The set of normal forms $\mathscr{N}_{[]}$ is generated by the following grammar:

$$\mathrm{N} ::= \mathbf{0} \mid a.\mathrm{N} \mid x[f] \mathbin{⌊\!⌊} \mathrm{N} \mid \mathrm{N} + \mathrm{N},$$

where $a \in \mathscr{A}$, $x \in \mathscr{V}$, and $f : \mathscr{L} \to \mathscr{L}$ is a relabelling function. We refer to $a.\mathrm{N}$ and $x[f] \mathbin{⌊\!⌊} \mathrm{N}$ as simple normal forms.

Lemma 7. *Every process term $p \in \mathscr{T}_{[]}$ has a normal form $s \in \mathscr{N}_{[]}$ such that $p \approx s$ is provable using RL1–RL6, LM1–LM5, and M.*

Because of Lemma 7, each term can be written using the following general form:

$$\sum_{i \in I} a_i.s_i + \sum_{j \in J} (x_j[f_j]) \mathbin{\lfloor\!\lfloor} s_j \quad \text{(modulo A1, A2 and A4)}$$

for finite index sets I, J and with $a_i \in \mathscr{A}$, $s_i, s_j \in \mathscr{N}_{[]}$, $x_j \in \mathscr{V}$, and relabelling functions $f_j : \mathscr{L} \to \mathscr{L}$.

Our goal now is to find a distinguishing valuation for each pair of non-bisimilar normal forms. In the following definitions and lemmas $\mathbb{P}$ denotes the set of prime numbers.

Definition 5. Let $\lfloor \cdot \rfloor : \mathscr{L} \to \mathbb{P}$ be an injective function, w a prime number larger than any prime number in the range of $\lfloor \cdot \rfloor$, and let $\lceil \cdot \rceil : \mathscr{V} \to \{m \in \mathbb{P} \mid m > w\}$ be another injective function. We define the valuation $\diamond_w$ for each variable $x \in \mathscr{V}$ by:

$$\diamond_w(x) = a.\zeta_{\lceil x \rceil, w} \text{ with } \zeta_{i,w} = a.\mathbf{0} + \sum_{b \in \mathscr{L}} \sum_{j=1}^{w} b^{i \cdot \lfloor b \rfloor^j}.\mathbf{0},$$

where a is an arbitrary action in $\mathscr{A}$.

Our aim in defining the valuation $\diamond_w$ is again to be able to distinguish the different types of simple normal forms that may occur as summands of a normal form. As in Sect. 3, we will be able to distinguish summands of the form $a.s$ from those of the form $x[f] \mathbin{\lfloor\!\lfloor} s'$ since the unique residual of terms with the latter form will have a larger branching degree than the unique residual of action-prefixed terms—see Lemma 10 to follow. However, in the definition of $\diamond_w$ we also want to ensure that terms of the form $\zeta_i, w[f]$ are prime, and that the sequences of actions those terms afford "encode" the relabelling function f. We obtain the primality of $\zeta_{i,w}[f]$ by means of the summand $a.\mathbf{0}$ of $\zeta_{i,w}$, whereas we encode relabelling functions by taking sequences of actions whose lengths are powers of distinct prime numbers. This is enough to ensure that if $\zeta_{i,w}[f]$ and $\zeta_{i,w}[g]$ are bisimilar, then $f = g$—see Lemma 11 to follow.

Lemma 8. *For all $i \geq 1$ and relabelling functions $f : \mathscr{L} \to \mathscr{L}$, the process $\zeta_{i,w}[f]$ is parallel prime, and its branching degree is $\mathrm{b}(\zeta_{i,w}[f]) = 1 + |\mathscr{L}| \cdot w$.*

Again, the distinguishing ability of the valuation $\diamond_w$ depends on the value of the parameter w being greater than an estimated highest branching degree occurring already in s. This is explained in Lemma 10 below; first we formalise an appropriate estimation of the highest branching degree occurring in a normal form s.

Definition 6. For all $s \in \mathscr{N}_{[]}$, the estimated highest branching degree $\mathrm{esb}(s)$ is defined inductively as follows:

$$\begin{aligned} &\mathrm{esb}(\mathbf{0}) = 0, && \mathrm{esb}(s+t) = \mathrm{esb}(s) + \mathrm{esb}(t), \\ &\mathrm{esb}(a.t) = \max(1, \mathrm{esb}(t)), && \mathrm{esb}(x[f] \mathbin{\lfloor\!\lfloor} t) = \max(1, \mathrm{esb}(t)), \end{aligned}$$

with $a \in \mathscr{A}$, $x \in \mathscr{V}$, relabelling function $f : \mathscr{L} \to \mathscr{L}$ and $t \in \mathscr{N}_{[]}$.

The following lemma shows that the estimated branching degree of s is an upper bound on the branching degree of $[\![s]\!]_{\diamond_w}$.

Lemma 9. *For every normal form $s \in \mathscr{N}_{[]}$, $\mathrm{b}([\![s]\!]_{\diamond_w}) \leq \mathrm{esb}(s)$.*

Lemma 10. *Let $s, s' \in \mathscr{N}_{[]}$ be simple normal forms, $x \in \mathscr{V}$, $f : \mathscr{L} \to \mathscr{L}$ a relabelling function and let $w > \mathrm{esb}(s)$.*

1. If $s = a.s'$, then the unique residual of $[\![s]\!]_{\diamond_w}$ has a branching degree smaller than w.
2. If $s = (x[f]) \mathbin{\|\!\!\lfloor} s'$, then the unique residual of $[\![s]\!]_{\diamond_w}$ has a branching degree larger than w.

When it has been determined from the unique residual of $[\![s]\!]_{\diamond_w}$ that s has a left merge as head operator, then the following key lemma allows us to determine which variable and which relabelling function occur in its left argument.

Lemma 11. *For $i, j \geq 1$ and relabelling functions $f, g : \mathscr{L} \to \mathscr{L}$, if $\zeta_{i,w}[f] = \zeta_{j,w}[g]$, then $i = j$ and $f = g$.*

Proof. *From $\zeta_{i,w}[f] = \zeta_{j,w}[g]$ it follows that $\mathrm{d}(\zeta_{i,w}[f]) = \mathrm{d}(\zeta_{j,w}[g])$ and therefore $i \cdot \lfloor b \rfloor^w = j \cdot \lfloor b \rfloor^w$ for that $b \in \mathscr{L}$ for which $\lfloor b \rfloor$ is largest. Since $\lfloor b \rfloor$ is positive, $i = j$. It remains to prove that $f = g$. Let $b \in \mathscr{L}$. Then $\zeta_{i,w}[f] \xrightarrow{f(b)} \big(b^{(i \cdot \lfloor b \rfloor^w) - 1}\big)[f]$. By the assumption that $\zeta_{i,w}[f] = \zeta_{j,w}[g]$ and since $i = j$, it follows that there also exists some $c \in \mathscr{L}$ such that $f(b) = g(c)$, $\zeta_{i,w}[g] \xrightarrow{f(c)} \big(c^{(i \cdot \lfloor c \rfloor^v) - 1}\big)[f]$ and $\big(b^{(i \cdot \lfloor b \rfloor^w) - 1}\big)[f] = \big(c^{(i \cdot \lfloor c \rfloor^v) - 1}\big)[g]$. Hence $i \cdot \lfloor b \rfloor^w = i \cdot \lfloor c \rfloor^v$ and since $\lfloor b \rfloor$ and $\lfloor c \rfloor$ are prime, it follows that $b = c$ and $w = v$. Therefore, $f(b) = g(b)$.* □

Using the previous lemmas, and reasoning as in the proof of Theorem 2, we can now prove that the valuation defined in Definition 5 is indeed distinguishing.

Theorem 3. *For every two normal forms $s, t \in \mathscr{N}_{[]}$ with $w > \mathrm{esb}(s), \mathrm{esb}(t)$, it holds that if $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$, then $s \approx t$ modulo A1–A4.*

Proof. *Assume that $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$ holds; we prove that $s \approx t$ is derivable using A1–A4 by induction on the sum of the depths of s and t. We do this by showing that for each summand s_i of s there exists a summand t_j of t such that $s_i \approx t_j$ modulo A1–A4. By symmetry this suffices to prove the claim.*

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 must also be a summand t_j of t such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s_j']\!]_{\diamond_w}$. By Lemma 10 we know that t_j must have the form $b.t_j'$, because the branching degree of the unique residual of $[\![t_j']\!]_{\diamond_w}$ does not exceed w.
Given that t_j has this form, it can only perform one transition: $[\![t_j]\!]_{\diamond_w} \xrightarrow{b} [\![t_j']\!]_{\diamond_w}$. Since also $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s_i']\!]_{\diamond_w}$ it 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 $s_i = x[f] \mathbin{\|\!\lfloor} s'_i$, then $[\![s_i]\!]_{\diamond_w} \xrightarrow{f(a)} \zeta_{\lceil x\rceil,w}[f] \parallel [\![s'_i]\!]_{\diamond_w} = p$. Since $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$, there must be a summand $t_j = y[g] \mathbin{\|\!\lfloor} t'_j$ of t such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{g(b)} \zeta_{\lceil y\rceil,w}[g] \parallel [\![t'_j]\!]_{\diamond_w} = q$ and $p = q$. By Lemma 9, the right-hand side parallel components of p and q have branching degrees not exceeding w whereas, by Lemma 8, the left-hand side parallel components are parallel prime and have branching degree $1 + |\mathscr{L}| \cdot w$. Using Theorem 1 it follows that $\zeta_{\lceil x\rceil,w}[f] = \zeta_{\lceil y\rceil,w}[g]$ and $[\![s'_j]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$. By Lemma 11 we have that $\lceil x\rceil = \lceil y\rceil$ and $f = g$. Hence, $x = y$ by injectivity of $\lceil\cdot\rceil$. By induction, we have that $s'_i \approx t'_j$ modulo A1–A4. Therefore $x[f] \mathbin{\|\!\lfloor} s'_j$ is provably equal to a summand of t.*

It follows by a symmetric argument that every summand of t is also provably equal to a summand of s using the above mentioned equations. Hence, $s \approx s + t \approx t$ modulo A1–A4. □

Corollary 3. *For all process terms $p, q \in \mathscr{T}_{[]}$ it holds that $p \approx q$ is provable using A1–A4, RL1–RL6, LM1–LM5, and M if, and only if, $p \leftrightarroweq q$.*

5 Restriction and Relabelling

In this section, we consider the language that includes both restriction and relabelling operators.

The set of normal forms $\mathscr{N}_{\backslash,[]}$ is generated by the following grammar:

$$\mathrm{N} ::= \mathbf{0} \mid a.\mathrm{N} \mid (x \setminus L)[f] \mathbin{\|\!\lfloor} \mathrm{N} \mid \mathrm{N}+\mathrm{N}$$

where $a \in \mathscr{A}$, $x \in \mathscr{V}$, $L \subset \mathscr{L}$, and $f : \mathscr{L} \to \mathscr{L}$ is a relabelling function satisfying $f = f \restriction (\mathscr{L} - L)$ (i.e., f is the identity on all $a \in L$). We refer to the normal forms $a.\mathrm{N}$ and $(x \setminus L)[f] \mathbin{\|\!\lfloor} \mathrm{N}$ as simple normal forms.

Lemma 12. *Every process term $p \in \mathscr{T}_{\backslash,[]}$ has a normal form $s \in \mathscr{N}_{\backslash,[]}$ such that $p \approx s$ is provable using RS1a–RS6, RL1–RL6, RR1, RR2, LM1–LM5, and M.*

Now, using the previous lemma, each term can be written using the following general form:

$$\sum_{i\in I} a_i.s_i + \sum_{j\in K} (x_j \setminus L_j)[f_j] \mathbin{\|\!\lfloor} s_j \quad \text{(modulo A1, A2, A4, and RR2)}$$

for finite index sets I, J and with $a_i \in \mathscr{A}$, $s_i, s_j \in \mathscr{N}_{\backslash,[]}$, $x_j \in \mathscr{V}$, $L_j \subset \mathscr{L}$, and relabelling functions $f_j : \mathscr{L} \to \mathscr{L}$ with $f_j = f_j \restriction (\mathscr{L} - L_j)$.

A valuation that distinguishes an action prefix from a variable under restriction and relabelling can be constructed by combining the ideas underlying the valuations presented in Definitions 3 and 5. The result shown below uses powers of distinct prime numbers to “encode” the relabelling function and employs a summation over all actions to allow for the detection of the restricting set.

Definition 7. Let $\lfloor\cdot\rfloor : \mathscr{L} \to \mathbb{P}$ be an injective function, w a prime number larger than any prime number in the range of $\lfloor\cdot\rfloor$, and let $\lceil\cdot\rceil : \mathscr{V} \to \{m \in \mathbb{P} \mid m > w\}$ be another injective function. We define the valuation $\diamond_w$ for each variable $x \in \mathscr{V}$ by:

$$\diamond_w(x) = \sum_{a\in\mathscr{L}} a.\chi_{\lceil x\rceil,w} \text{ with } \chi_{i,w} = \sum_{a\in\mathscr{L}} \left(a.\mathbf{0} + \sum_{j=1}^{w} a^{i\cdot\lfloor a\rfloor^j}.\mathbf{0} \right).$$

First, we establish two properties of the process $(\chi_{\lceil x\rceil,w} \setminus L)[f]$, which is a parallel component of the unique residual of $[\![(x\setminus L)[f] \mathbin{|\!\lfloor} s]\!]_{\diamond_w}$.

Lemma 13. *For all $i > 1$, $L \subset \mathscr{L}$, and relabelling functions $f : \mathscr{L} \to \mathscr{L}$, the process $(\chi_{i,w} \setminus L)[f]$ is parallel prime, and its branching degree is $|f(\mathscr{L} - L)| + |\mathscr{L} - L| \cdot w$.*

To enable the valuation $\diamond_w$ to distinguish between an action prefix and a term with the left merge as head operator, as explained in Lemma 15 below, we need an appropriate estimation of the highest branching degree occurring in a normal form s.

Definition 8. For all $s \in \mathscr{N}_{\setminus,[]}$, the lower bound estimate of the branching degree of s, denoted with $\mathrm{esb}(s)$, is defined inductively as follows:

$$\begin{array}{ll} \mathrm{esb}(\mathbf{0}) = 0, & \mathrm{esb}(s+t) = \mathrm{esb}(s) + \mathrm{esb}(t), \\ \mathrm{esb}(a.t) = \max(1, \mathrm{esb}(t)), & \mathrm{esb}((x\setminus L)[f] \mathbin{|\!\lfloor} t) = \max(|\mathscr{L}|, \mathrm{esb}(t)). \end{array}$$

with $a \in \mathscr{A}, x \in \mathscr{V}, L \subset \mathscr{L}$, relabelling function $f : \mathscr{L} \to \mathscr{L}$ and $t \in \mathscr{N}_{\setminus,[]}$.

The following lemma shows that the estimated branching degree of s is an upper bound on the branching degree of $[\![s]\!]_{\diamond_w}$.

Lemma 14. *For every normal form $s \in \mathscr{N}_{\setminus,[]}$, $\mathrm{b}([\![s]\!]_{\diamond_w}) \leq \mathrm{esb}(s)$.*

Lemma 15. *Let $s, s' \in \mathscr{N}_{\setminus,[]}$ be simple normal forms, $x \in \mathscr{V}$, $L \subset \mathscr{L}$, $f : \mathscr{L} \to \mathscr{L}$ a relabelling function and let $w > \mathrm{esb}(s)$.*

1. If $s = a.s'$, then the unique residual of $[\![s]\!]_{\diamond_w}$ has a branching degree smaller than w.
2. If $s = (x\setminus L)[f] \mathbin{|\!\lfloor} s'$, then the unique residual of $[\![s]\!]_{\diamond_w}$ has a branching degree larger than w.

The following lemma allows us to determine the variable, the restriction set and relabelling function in a simple normal form of the shape $(x\setminus L)[f] \mathbin{|\!\lfloor} s$.

Lemma 16. *For $w \in \mathbb{P}$, $i, j \in \{m \in \mathbb{P} \mid m > w\}$, $K, L \subset \mathscr{L}$, and relabelling functions $f, g : \mathscr{L} \to \mathscr{L}$, if $(\chi_{i,w} \setminus K)[f] = (\chi_{j,w} \setminus L)[g]$, then $K = L$, $f \upharpoonright (\mathscr{L} - K) = g \upharpoonright (\mathscr{L} - K)$ and $i = j$.*

By following the strategy we adopted in the proofs of Theorems 2 and 3, we can show that the valuation defined above is indeed distinguishing.

Theorem 4. *For every two normal forms $s, t \in \mathscr{N}_{\setminus,[]}$ with $w > \mathrm{esb}(s), \mathrm{esb}(t)$, it holds that if $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$, then $s \approx t$ modulo A1–A4 and RR2.*

Proof. *We now prove that $s \approx t$ assuming that $[\![s]\!]_{\diamond_w} = [\![t]\!]_{\diamond_w}$ by induction on the sum of the depths of s and t. We do so 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 and RR2. 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 summand t_j of t such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{a} [\![s'_i]\!]_{\diamond_w}$. By Lemma 15 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 transition: $[\![t_j]\!]_{\diamond_w} \xrightarrow{b} [\![t'_j]\!]_{\diamond_w}$. 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 and RR2. Hence, we may conclude that $s_i = a.s'_i \approx b.t'_j = t_j$.

2. *If $s_i = (x \setminus K)[f] \mathbin{|\!\lfloor} s'_i$, then, since $K \subset \mathscr{L}$, $[\![s_i]\!]_{\diamond_w} \xrightarrow{f(a)} (\chi_{\lceil x \rceil,w} \setminus K)[f] \parallel [\![s'_i]\!]_{\diamond_w} = p$ (by Definition 7) for some $a \in \mathscr{L} - K$. We know that also a summand t_j of t exists such that $[\![t_j]\!]_{\diamond_w} \xrightarrow{f(a)} p$. Similarly as in previous case, by Lemma 15, we also know that t_j must have the form $(y \setminus L)[g] \mathbin{|\!\lfloor} t'_j$ for some $y \in \mathscr{V}$, $L \subset \mathscr{L}$, $g : \mathscr{L} \to \mathscr{L}$ such that $g(b) = f(a)$ for some $b \in \mathscr{L} - L$, and t'_j. The residual of t_j after performing an action $g(b)$ with $b \in \mathscr{L} - L$ is $(\chi_{\lceil y \rceil,w} \setminus L) \parallel [\![t'_j]\!]_{\diamond_w}$ (also by Definition 7). This residual is equal to p, so we know that $(\chi_{\lceil x \rceil,w} \setminus K)[f] \parallel [\![s'_i]\!]_{\diamond_w} = (\chi_{\lceil y \rceil,w} \setminus L)[g] \parallel [\![t'_j]\!]_{\diamond_w}$.*
   By Lemma 13 we have that the process $(\chi_{\lceil x \rceil,w} \setminus K)[f]$ is parallel prime and has a branching degree that exceeds w. This process cannot occur in the unique parallel decomposition of $[\![t'_j]\!]_{\diamond_w}$ because, by Lemma 1 and the fact that $w > \mathrm{esb}(t) \geq \mathrm{esb}(t'_j)$, the branching degrees of all processes in the parallel decomposition of $[\![t'_j]\!]_{\diamond_w}$ do not exceed w. Conversely, this also holds in a symmetric way for the process $(\chi_{\lceil y \rceil,w} \setminus L)[g]$ with respect to the unique parallel decomposition of $[\![s'_i]\!]_{\diamond_w}$. Hence by Theorem 1, $(\chi_{\lceil x \rceil,w} \setminus K)[f] = (\chi_{\lceil y \rceil,w} \setminus L)[g]$ and $[\![s'_i]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$.
   From $(\chi_{\lceil x \rceil,w} \setminus K)[f] = (\chi_{\lceil y \rceil,w} \setminus L)[g]$ it follows by Lemma 16 that $\lceil x \rceil = \lceil y \rceil$, $K = L$ and $f \restriction (\mathscr{L} - K) = g \restriction (\mathscr{L} - K)$. By the injectivity of $\lceil \cdot \rceil$ we know also that $x = y$.
   Since $[\![s'_i]\!]_{\diamond_w} = [\![t'_j]\!]_{\diamond_w}$, by induction hypothesis it follows that $s'_i \approx t'_j$ modulo A1–A4 and RR2.
   Summing up, we have established that $K = L$, $f \restriction (\mathscr{L} - K) = g \restriction (\mathscr{L} - K)$, $x = y$, and $s'_i \approx t'_j$ modulo A1–A4 and RR2, We may therefore conclude that $s_i = (x \setminus K)[f] \mathbin{|\!\lfloor} s'_i \approx (y \setminus L)[g] \mathbin{|\!\lfloor} t'_j = t_j$.

The above analysis shows that for each summand s_i of s there exists a summand t_j of t such that $s_i \approx t_i$ modulo A1–A4 and RR2. It follows by a symmetric argument that every summand of t is also provably equal to a summand of s using the above mentioned equations. Hence, $s \approx s + t \approx t$ modulo A1–A4 and RR2. ☐

Corollary 4. *For all process terms $p, q \in \mathscr{T}_{\setminus,[]}$ it holds that $p \approx q$ if, and only if, $p \leftrightarrows q$.*

Acknowledgements

The work of Aceto and Ingólfsdóttir has been partially supported by the projects “The Equational Logic of Parallel Processes” (nr. 060013021) and “New Developments in Operational Semantics” (nr. 080039021) of the Icelandic Research Fund.

The research of van Tilburg was supported by the project “Models of Computation: Automata and Processes” (nr. 612.000.630) of the Netherlands Organisation for Scientific Research (NWO).

References

1. Aceto, L., Fokkink, W., Ingólfsdóttir, A., Luttik, B.: A finite equational base for CCS with left merge and communication merge. ACM Trans. Comput. Log. (2008). To appear (available as `http://tocl.acm.org/accepted/310luttik.pdf`).
2. Baeten, J.C.M., Bergstra, J.A.: Global renaming operators in concrete process algebra. Information and Computation **78**(3), 205–245 (1988)
3. Bergstra, J., Klop, J.: Process algebra for synchronous communication. Information and Control **60**(1-3), 109–137 (1984)
4. Birkhoff, G.: On the structure of abstract algebras. Proceedings of the Cambridge Philosophical Society **31**, 433–454 (1935)
5. Christensen, S., Hirshfeld, Y., Moller, F.: Decidable subsets of CCS. The Computer Journal **37**(4), 233–242 (1994)
6. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of the ACM (JACM) **32**(1), 137–161 (1985)
7. Milner, R.: Flowgraphs and flow algebras. Journal of the ACM (JACM) **26**(4), 794–818 (1979)
8. Milner, R.: A Calculus of Communicating Systems. Springer (1980)
9. Milner, R.: Communication and Concurrency. Prentice-Hall (1989)
10. Milner, R., Moller, F.: Unique decomposition of processes. Theoret. Comput. Sci. **107**, 357–363 (1993)
11. Moller, F.: Axioms for Concurrency. Ph.D. thesis (1989)
12. Park, D.: Concurrency and automata on infinite sequences. In: P. Deussen (ed.) 5^{h} GI Conference, *Lecture Notes in Computer Science*, vol. 104, pp. 167–183. Springer (1981)
13. de Simone, R.: Higher level synchronization devices in SCCS-Meije. Theoretical Computer Science **37**, 245–267 (1985)