Decidable Subsets of CCS

Søren Christensen* Yoram Hirshfeld†

Faron Moller‡

Laboratory for Foundations of Computer Science

University of Edinburgh

Abstract

CCS is a universal formalism: any computable function is computed by some CCS agent. Moreover, one can reduce the halting problem for Turing machines to the problem of deciding bisimilarity of two CCS agents, thus demonstrating the undecidability of the equivalence checking problem. In this paper, we demonstrate the limits of decidability of CCS. In particular, we show that by simply disallowing either of communication or both restriction and relabelling, we arrive at a sublanguage which still describes a rich class of infinite state systems but for which bisimulation is decidable. We also demonstrate complete axiomatisations for these sublanguages. We compare these results with the undecidability of all other common equivalences.

1 Introduction

The study of process algebras has received a great deal of attention over the past 15 years, and has merited this attention by providing a natural framework for describing and analyzing concurrent systems. Milner's CCS [21] is but one of an array of such calculi, and is representative of the area: it provides a language in which concurrent systems can be described naturally with constructs which have intuitive interpretations, such as parallel composition and abstraction.

The goal of such a formalism is to provide techniques for verifying the correctness of systems. Typically this verification takes the form of demonstrating the equivalence between two processes expressed in the formalism, representing the specification of the system and its implementation. However, any reasonable process algebra such as CCS allows the description of any computable function, and the equivalence problem, regardless of the notion of equivalence you consider, is readily seen to be undecidable in general. Much can be accomplished by restricting attention to communicating finite state systems where the equivalence problem is equally quickly seen to be decidable, but most realistic applications involve infinite state systems. Hence much interest lies in the problem of

*Currently at Systematic Software Engineering A/S, Aarhus.

†On Sabbatical leave from The School of Mathematics and Computer Science, Tel Aviv University.

‡Supported by ESPRIT BRA 7166: CONCUR2.

identifying classes of infinite state systems — *e.g.,* subalgebras of CCS — in which the equivalence problem is decidable.

The first point to settle when exploring the decidability of the equivalence checking problem is the notion of equivalence which is going to be considered. In this paper we shall be particularly interested in bisimulation equivalence [23, 21]. Apart from being the fundamental notion of equivalence for CCS, this behavioural equivalence has several pleasing properties, not least of which being that — as we shall discover — it is decidable over process classes for which all other common equivalences remain undecidable.

Proofs of the undecidability of bisimulation equivalence over full CCS [21, 24] involve modelling (the execution of) Turing machines in CCS and reducing the halting problem to equivalence checking. These modellings require explicit and intricate use of the communication capabilities of the calculus in conjunction with the restriction operator, the mechanism for abstraction in CCS, and relabelling. In this paper we demonstrate that this undecidability indeed results from the marriage between communication and restriction/relabelling. That is, we demonstrate the decidability of CCS without restriction and relabelling, as well as the decidability of CCS without communication. In both cases, we can still express nontrivial infinite state systems.

Our decidability result is based on a particular tableau systems. From this tableau system, we construct a syntactic axiomatisation for reasoning about equality for which we demonstrate soundness and completeness. Such an axiomatisation can be considered as an extension of Milner's axiomatisation for finite state systems [20].

We close this paper with a discussion of related work on the decidability of equivalences over classes of infinite state systems.

2 Preliminaries

To define the version of CCS which we shall be considering, we presuppose a set of *atomic labels* $\Lambda = \{a, b, c, \ldots\}$ not containing τ, and we define $Act = \Lambda \cup \{\tau\}$. We assume a complementation function $\bar{\ } : Act \rightarrow Act$ satisfying $\overline{\overline{a}} = a$ and $\overline{\tau} = \tau$. Complementary labels form the basis of the handshake communication used in the calculus. We also presuppose a set of *process variables* $Var = \{X, Y, Z, \ldots\}$.

The language of CCS expressions is defined by the BNF equation

$$E, F \quad ::= \quad \mathbf{0} \;\Big|\; X \;\Big|\; aE \;\Big|\; E + F \;\Big|\; E \mid F \;\Big|\; E\backslash L \;\Big|\; E[f]$$

where we take $a \in Act$, $X \in Var$, $L \subseteq \Lambda$, and $f : Act \rightarrow Act$ ranging over *relabelling* functions which behave as the identity function on all actions outside of a finite set of labels, and satisfy $\overline{f(a)} = f(\overline{a})$. Briefly, the interpretation of these expressions is as follows:

- $\mathbf{0}$ represents the *nil* process, one which performs no events;
- X represents the process bound to the variable in some assumed environment;
- aE represents the process which performs the action a and evolves into the process E;
- $E + F$ represents a choice between behaving as E or as F, with the choice being made at the time of the first action;

- $E \mid F$ represents the (concurrent) composition of the processes E and F, which allows the two processes to evolve independently, or to synchronise on complementary labels;

- $E \backslash L$ represents the process which behaves as E except cannot perform any action a or $\overline{a}$ in which $a \in L$; and

- $E[f]$ represents the process E in which all actions are relabelled by f.

As a point of notation, we shall omit trailing $\mathbf{0}$s from expressions, thus writing the term $a\mathbf{0}$ simply as a. Also we shall write E^n to represent the term $E \mid \cdots \mid E$ consisting of n copies of E combined in parallel.

A CCS *process* is defined by a finite family of recursive process equations

$$\Delta = \left\{ X_i \stackrel{\text{def}}{=} E_i \;:\; 1 \leq i \leq n \right\}$$

where the X_is are distinct and the E_is are CCS expressions at most containing the variables $Var(\Delta) = \{X_1, \ldots, X_n\}$. We further assume that every variable occurrence in the E_is is *guarded*, that is, appears within the scope of an action prefix aE. The variable X_1 is singled out as the *leading variable* and $X_1 = E_1$ is called the *leading equation*. (Strictly speaking, a process is defined by some CCS term coupled with a family of process equations, but we shall ignore this point with hopefully little technical confusion for the reader.)

A fuller description of these constructs is left to [21]. Notice though that a more general syntax is permitted there, allowing unguarded recursions and infinite relabelling functions. However, our restrictions are quite standard, do not hamper the expressive power of the calculus when describing real systems, and are thus usually assumed in practical applications, for example, when using an automated verification tool such as the Edinburgh Concurrency Workbench [10].

Any finite family Δ of CCS equations determines a labelled transition system. The transition relation is given as the least relation satisfying the following rules.

$$aE \stackrel{a}{\longrightarrow} E \qquad \frac{E \stackrel{a}{\longrightarrow} E'}{E \mid F \stackrel{a}{\longrightarrow} E' \mid F} \qquad \frac{E \stackrel{a}{\longrightarrow} E'}{E\backslash L \stackrel{a}{\longrightarrow} E'\backslash L}(a, \overline{a} \notin L)$$

$$\frac{E \stackrel{a}{\longrightarrow} E'}{E + F \stackrel{a}{\longrightarrow} E'} \qquad \frac{F \stackrel{a}{\longrightarrow} F'}{E \mid F \stackrel{a}{\longrightarrow} E \mid F'} \qquad \frac{E \stackrel{a}{\longrightarrow} E'}{E[f] \stackrel{f(a)}{\longrightarrow} E'[f]}$$

$$\frac{F \stackrel{a}{\longrightarrow} F'}{E + F \stackrel{a}{\longrightarrow} F'} \qquad \frac{E \stackrel{a}{\longrightarrow} E', \;\; F \stackrel{\overline{a}}{\longrightarrow} F'}{E \mid F \stackrel{\tau}{\longrightarrow} E' \mid F'}(a \neq \tau) \qquad \frac{E \stackrel{a}{\longrightarrow} E'}{X \stackrel{a}{\longrightarrow} E'}(X \stackrel{\text{def}}{=} E \in \Delta)$$

Strictly speaking, transitions are defined on CCS expressions relative to some family Δ of process equations. However, we shall usually leave the reader to infer the intended family.

Remark 2.1 *It is easy to verify that* CCS *processes generate finite-branching transition graphs, that is, graphs for which the set* $\{F : E \stackrel{a}{\longrightarrow} F\}$ *is finite for each* E *and each* a. *This would not be true if we allowed unguarded expressions. For example, the process* $X \stackrel{\text{def}}{=} a + a \mid X$ *generates an infinite-branching transition graph.*

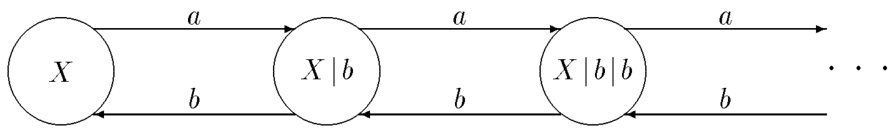

Figure 1: The transition graph for $\left\{X \stackrel{\text{def}}{=} a(X \mid b)\right\}$.

In order to simplify our later analysis, we wish to identify several process expressions. A typical case is that we want to recognize that $\mid$ is commutative and associative. We therefore define the following structural congruence over process expressions.

Definition 2.2 *Let $\equiv$ be the smallest congruence relation over process expressions such that the laws of associativity, commutativity and* **0***-absorption hold for choice and composition.*

When inferring transitions we may ignore harmless **0**-components sitting in parallel. Thus not to be annoyed by such innocent matters we shall always assume that transitions have been inferred modulo the structural congruence $\equiv$. We note that we can safely do so since the semantic equivalence of bisimilarity (which we introduce shortly) satisfies the basic laws underlying the structural congruence $\equiv$.

Example 2.3 *Let Δ be the family $\{X \stackrel{\text{def}}{=} a(X \mid b)\}$. By the transition rules above (modulo $\equiv$) X generates the infinite-state transition graph of Figure 1.*

The equivalence between CCS expressions (states) which we are interested in considering here is bisimilarity [21], defined as follows.

Definition 2.4 *A binary relation $\mathcal{R}$ over* CCS *expressions (states) is a* <u>bisimulation</u> *if whenever $E\mathcal{R}F$ then for each $a \in \Lambda$,*

- *if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' with $E'\mathcal{R}F'$;*
- *if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' with $E'\mathcal{R}F'$.*

Processes E and F are <u>bisimilar</u>, *written $E \sim F$, if they are related by some bisimulation.*

By $Var(\Delta)^{\otimes}$ we denote the set of finite multisets over $Var(\Delta) = \{X_1, \ldots, X_n\}$ and let Greek letters $\alpha, \beta, \ldots$ range over elements of $Var(\Delta)^{\otimes}$. Each such α denotes a CCS process by forming the product of the elements of α, i.e. by combining the elements of α in parallel using the composition operator. We recognise the empty product as **0**, and we ignore the ordering of variables in products, hence identifying processes denoted by elements of $Var(\Delta)^{\otimes}$ up to associativity and commutativity of composition.

Definition 2.5 *A finite family $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \;:\; 1 \leq i \leq n\}$ of guarded* CCS *equations is defined to be in* <u>standard form</u> *iff every expression E_i is of the form*

$$a_1\alpha_1 \; + \; \cdots \; + \; a_m\alpha_m$$

where $\alpha_j \in Var(\Delta)^{\otimes}$ *for each* j. *Again, we recognise the empty sum as* $\mathbf{0}$, *and ignore the ordering of expressions in sums, hence defining the notion of standard form modulo associativity and commutativity of choice.*

In [21] it is shown that any finite family Δ of guarded CCS equations has a unique solution up to bisimilarity. Moreover, we have the following result showing when such a system can be effectively presented in standard form.

Lemma 2.6 *Given any finite family of guarded* CCS *equations* Δ *which either does not involve restriction and relabelling or does not involve communication (so that composition is given by communication-free merge), we can* effectively *construct another finite family of* CCS *equations* Δ' *in standard form in which* $\Delta \sim \Delta'$, *i.e. the leading variables of* Δ *and* Δ' *are bisimilar.*

Proof: See Appendix A. □

For our proof of decidability of bisimulation equivalence we shall rely on the following ordering on $Var(\Delta)^{\otimes}$.

Definition 2.7 *By* $\sqsubset$ *we denote the well-founded ordering on* $Var(\Delta)^{\otimes}$ *given as follows:*

$$X_1^{k_1} \mid X_2^{k_2} \mid \cdots \mid X_n^{k_n} \quad \sqsubset \quad X_1^{l_1} \mid X_2^{l_2} \mid \cdots \mid X_n^{l_n}$$

iff there exists j *such that* $k_j < l_j$ *and for all* $i < j$ *we have* $k_i = l_i$.

It is straightforward to show that $\sqsubset$ is well-founded. We shall furthermore rely on the fact that $\sqsubset$ is *total* in the sense that for any $\alpha, \beta \in Var(\Delta)^{\otimes}$ with $\alpha \not\equiv \beta$ it follows that $\alpha \sqsubset \beta$ or $\beta \sqsubset \alpha$. Also we shall rely on the fact that $\beta \sqsubset \alpha$ implies $\beta \mid \gamma \sqsubset \alpha \mid \gamma$ for any $\gamma \in Var(\Delta)^{\otimes}$. These properties are easily seen to hold for $\sqsubset$.

3 Decidability

In this section we fix a finite family $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \;:\; 1 \le i \le n\}$ of guarded CCS equations in standard form. We are interested in deciding for any α and β of $Var(\Delta)^{\otimes}$ whether $\alpha \sim \beta$ is the case or not. The procedure for checking $\alpha \sim \beta$ is based on the *tableau decision method* as for instance utilised by Hüttel and Stirling in [17]. The tableau system is a goal directed proof system. The rules of the tableau system are built around equations $E = F$ where E and F are CCS expressions. Each rule has the form

$$\frac{E = F}{E_1 = F_1 \qquad \cdots \qquad E_n = F_n}$$

possibly with side conditions. The premise of the rule represents the goal to be achieved (that $E \sim F$) whereas the consequents represent (sufficient) subgoals to be established.

A *tableau* for $\alpha = \beta$ is a maximal proof tree whose root is labelled $\alpha = \beta$ and where the labelling of immediate successors of a node are determined according to the rules of the tableau system presented in Table 1. For the presentation of rule Rec we introduce the notation $\mathrm{unf}(\alpha)$ to mean the *unfolding* of α defined as follows: given $Y_i = \sum_{j=1}^{n_i} a_{ij}\alpha_{ij}$ for $1 \le i \le m$,

REC	$\dfrac{\alpha = \beta}{\mathrm{unf}(\alpha) = \mathrm{unf}(\beta)}$	
SUM	$\dfrac{\sum_{i=1}^{n} a_i\alpha_i = \sum_{j=1}^{m} b_j\beta_j}{\{a_i\alpha_i = b_{f(i)}\beta_{f(i)}\}_{i=1}^{n} \qquad \{b_j\beta_j = a_{g(j)}\alpha_{g(j)}\}_{j=1}^{m}}$	where $f : \{1,\dots,n\} \to \{1,\dots,m\}$ $g : \{1,\dots,m\} \to \{1,\dots,n\}$
PREFIX	$\dfrac{a\alpha = a\beta}{\alpha = \beta}$	
SUBL	$\dfrac{\alpha \mid \gamma = \delta}{\beta \mid \gamma = \delta}$	if the dominated node is labelled $\alpha = \beta$ or $\beta = \alpha$ with $\alpha \sqsupset \beta$
SUBR	$\dfrac{\delta = \alpha \mid \gamma}{\delta = \beta \mid \gamma}$	if the dominated node is labelled $\alpha = \beta$ or $\beta = \alpha$ with $\alpha \sqsupset \beta$

Table 1: Rules of the tableau system.

$$\begin{aligned}\mathrm{unf}\Big(\prod_{i=1}^{m} Y_i\Big) \quad &= \quad \sum_{i=1}^{m}\ \sum_{j=1}^{n_i} a_{ij}\Big(\prod_{\substack{k=1\\k\neq i}}^{m} Y_k \mid \alpha_{ij}\Big)\\ &+ \quad \sum_{\substack{i,i'=1\\i\neq i'}}^{m}\ \sum_{j=1}^{n_i}\ \sum_{\substack{j'=1\\a_{ij}=\overline{a}_{i'j'}\neq\tau}}^{n_{i'}} \tau\Big(\prod_{\substack{k=1\\k\neq i,i'}}^{m} Y_k \mid \alpha_{ij} \mid \alpha_{i'j'}\Big)\end{aligned}$$

We adopt some terminology for tableaux. Tableaux are denoted by T (and also by $T(\alpha = \beta)$ to indicate the label of the root). Paths are denoted by π and nodes are denoted by $\mathbf{n}$ (with roots also denoted by $\mathbf{r}$) possibly with subscripts. If a node $\mathbf{n}$ has label $E = F$ we write $\mathbf{n} : E = F$.

Nodes of the form $\mathbf{n} : \alpha = \beta$ are called *basic nodes*. When building tableaux basic nodes might *dominate* other basic nodes; we say a basic node $\mathbf{n} : \alpha \mid \gamma = \delta$ or $\mathbf{n} : \delta = \alpha \mid \gamma$ dominates any node $\mathbf{n}' : \alpha = \beta$ or $\mathbf{n}' : \beta = \alpha$ which appears above $\mathbf{n}$ in the tableau in which $\alpha \sqsupset \beta$ and to which rule REC is applied. Whenever a basic node dominates a previous one, we apply one of the SUB rules to reduce the terms before applying the REC rule. Notice that the side condition for the SUB rules is a condition on tableaux and not on the particular rule.

We shall identify CCS expressions in our tableaux up to the structural congruence $\equiv$, i.e. up to associativity, commutativity and $\mathbf{0}$-absorption of choice and composition. In

$$
\begin{array}{lccr}
 & \multicolumn{2}{c}{X_1 = X_2} & \\
\textsc{Rec} & \multicolumn{2}{c}{\overline{a(X_1 \mid X_4) = aX_3}} & \\
\textsc{Prefix} & \multicolumn{2}{c}{\overline{X_1 \mid X_4 = X_3}} & \\
\textsc{SubL} & \multicolumn{2}{c}{\overline{X_2 \mid X_4 = X_3}} & \\
\textsc{Rec} & \multicolumn{2}{c}{\overline{a(X_3 \mid X_4) + bX_2 = a(X_3 \mid X_4) + bX_2}} & \\
\textsc{Sum} & \overline{a(X_3 \mid X_4) = a(X_3 \mid X_4)} & \overline{bX_2 = bX_2} & \\
\textsc{Prefix} & \overline{X_3 \mid X_4 = X_3 \mid X_4} & \overline{X_2 = X_2} & \textsc{Prefix}
\end{array}
$$

Figure 2: A successful tableau for $X_1 = X_2$.

particular, we always assume that the labels of nodes have been pruned of $\mathbf{0}$ components sitting in parallel or in sum.

Finally, in building tableaux the rules are only applied to nodes that are not *terminal*. A terminal node can either be *successful* or *unsuccessful*. A successful terminal node is one labelled $\alpha = \alpha$, whilst an unsuccessful terminal node is one labelled either $a\alpha = \mathbf{0}$ or $\mathbf{0} = b\beta$ or $a\alpha = b\beta$ with $a \neq b$. A tableau is successful if and only if all terminal nodes are successful; otherwise it is unsuccessful.

Example 3.1 *Let* $\left\{X_1 \stackrel{\text{def}}{=} a(X_1 \mid X_4),\ X_2 \stackrel{\text{def}}{=} aX_3,\ X_3 \stackrel{\text{def}}{=} a(X_3 \mid X_4) + bX_2,\ X_4 \stackrel{\text{def}}{=} b\right\}$ *define a* CCS *process in standard form. In Figure 2 we give a successful tableau for* $X_1 = X_2$.

Lemma 3.2 *Every tableau for $\alpha = \beta$ is finite. Furthermore, there is only a finite number of tableaux for $\alpha = \beta$.*

Proof: Let $T(\alpha = \beta)$ be a tableau with root labelled $\alpha = \beta$. It can only be infinite if there exists an infinite path as every node has finite branching degree. Hence suppose π is such an infinite path starting at the root $\mathbf{r} : \alpha = \beta$. The path π can only be infinite if it contains infinitely many basic nodes to which the tableau rule Rec is applied. This is due to the well-foundedness of the ordering $\sqsubset$ on $Var(\Delta)^{\otimes}$ which is decreased through applications of the Sub rules. Thus from the path π we can form an infinite sequence S of nodes $\{\mathbf{n}_i : \alpha_i = \beta_i\}_{i=1}^{\infty}$ by collecting (in order of appearance) the basic nodes along π to which the rule Rec is applied. Hence $\mathbf{n}_1 : \alpha_1 = \beta_1$ represents the root, $\mathbf{n}_2 : \alpha_2 = \beta_2$ represents the second node along π at which Rec is applied, and so on.

An expression α can be viewed as a vector $\tilde{v}$ of $\mathbb{N}^n$: the value of the i^{th} coordinate of $\tilde{v}$, denoted $\tilde{v}(i)$, indicates the number of occurrences of variable X_i in α. Thus we can represent the sequence S by an infinite sequence of vectors $\{\tilde{u}_i\}_{i=1}^{\infty}$ where $\tilde{u}_i \in \mathbb{N}^{2n}$ for all i. The first n coordinates represent α_i and the last coordinates represent β_i.

Consider the infinite sequence $\{\tilde{u}_i(1)\}_{i=1}^{\infty}$ consisting of all the first coordinates of vectors of the sequence S. If this sequence has an upper bound we extract from S an infinite sequence S_1 of vectors $\{\tilde{v}_i\}_{i=1}^{\infty}$ with the property that the first coordinate of $\tilde{v}_i$ remains constant throughout S_1. If the sequence $\{\tilde{u}_i(1)\}_{i=1}^{\infty}$ does not have an upper bound we extract from S an infinite sequence S_1 of vectors $\{\tilde{v}_i\}_{i=1}^{\infty}$ with the property that the first coordinate of $\tilde{v}_i$ is nondecreasing, i.e. $\tilde{v}_i(1) \leq \tilde{v}_j(1)$ whenever $i \leq j$. Continuing in this fashion we arrive at an infinite sequence S_{2n} of vectors $\{\tilde{w}_i\}_{i=1}^{\infty}$ with the property that all coordinate sequences are nondecreasing. But then every node in this sequence is dominated by every node after it, so the rule REC cannot be applied to any of these nodes, as a SUB rule is applicable.

For the proof of the second part, we note that if there were an infinite number of tableaux, then since there are only a finite number of tableaux of a given finite size, there must be an infinite sequence of partial tableaux, each of which being derived from the previous by the application of some rule to the node most recently introduced. But then this sequence provides a tableau with an infinite path through it, which by the first part cannot be. □

We now proceed to show the soundness and completeness of the tableau system.

Theorem 3.3 (Completeness) *If $\alpha \sim \beta$ then there exists a successful tableau with root labelled $\alpha = \beta$.*

Proof: Suppose $\alpha \sim \beta$. If we can construct a tableau $T(\alpha = \beta)$ for $\alpha = \beta$ with the property that any node $\mathbf{n} : E = F$ of $T(\alpha = \beta)$ satisfies $E \sim F$, then by Lemma 3.2 that construction must terminate and each terminal will be successful. Thus the tableau itself will be successful.

We can construct such a $T(\alpha = \beta)$ if we verify that each rule of the tableau system is *forward sound* in the sense that if the antecedent as well as all nodes above relate bisimilar processes then it is possible to find a set of consequents relating bisimilar processes. It is easily verified that the rules are indeed forward sound in this sense. Notice in particular that the rule REC reflects the expansion law for composition [21] and that forward soundness of the SUB rules follows from the fact that bisimilarity is a congruence wrt composition. □

The proof of soundness of the tableau system relies on an alternative characterisation of bisimulation, viz. as a sequence of approximations.

Definition 3.4 *The sequence of bisimulation approximations $\{\sim_n\}_{n=0}^{\infty}$ is defined inductively as follows.*

- *$E \sim_0 F$ for all processes E and F;*
- *$E \sim_{n+1} F$ iff for each $a \in \Lambda$,*
  - *if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' with $E' \sim_n F'$;*
  - *if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' with $E' \sim_n F'$.*

It is a standard result that for finite-branching transition graphs, bisimulation is given as the limit of the above approximations [21]:

$$\sim \;=\; \bigcap_{n=0}^{\infty} \sim_n .$$

As noted in Remark 2.1, CCS processes are finite-branching.

Theorem 3.5 (Soundness) *If there is a successful tableau for $\alpha = \beta$ then $\alpha \sim \beta$.*

Proof: Suppose $T(\alpha = \beta)$ is a tableau for $\alpha = \beta$, and that $\alpha \not\sim \beta$. We shall construct a maximal path $\pi = \{\mathbf{n}_i : E_i = F_i\}$ through this tableau starting at the root $\alpha = \beta$ in which $E_i \not\sim F_i$ for each i. Hence the terminal node of this path cannot be successful, so there can be no successful tableau for $\alpha = \beta$.

While constructing π, we shall at the same time construct the sequence of integers $\{m_i \ : \ E_i \not\sim_{m_i} F_i \text{ and } E_i \sim_j F_i \text{ for all } j < m_i\}$. We shall also prove along the way that this sequence is nonincreasing, and strictly decreasing through applications of the rule Prefix.

Given $\mathbf{n}_i : E_i = F_i$ and m_i, we get $\mathbf{n}_{i+1} : E_{i+1} = F_{i+1}$ and m_{i+1} according to the following cases:

- If Rec is applied to $\mathbf{n}_i$, then the consequent is $\mathbf{n}_{i+1}$ and $m_{i+1} = m_i$.
- If Sum is applied to $\mathbf{n}_i$, then there must be some consequent $\mathbf{n}_{i+1} : E_{i+1} = F_{i+1}$ with $E_{i+1} \not\sim_{m_i} F_{i+1}$ so $m_{i+1} \leq m_i$.
- If Prefix is applied to $\mathbf{n}_i$, then the consequent is $\mathbf{n}_{i+1}$ and $m_{i+1} = m_i - 1$.
- If SubL is applied to $\mathbf{n}_i : E_i = F_i$ then $E_i = F_i$ must be of the form $\alpha \mid \gamma = \delta$ with dominated node $\mathbf{n}_j : \alpha = \beta$ $(\alpha \sqsupset \beta)$. Since between $\mathbf{n}_j$ and $\mathbf{n}_i$ there must have been an intervening application of the rule Prefix, we must have that $m_i < m_j$. We take the node $\mathbf{n}_{i+1} : \beta \mid \gamma = \delta$, and show that we have some valid $m_{i+1} \leq m_i$, that is, that $\beta \mid \gamma \not\sim_{m_i} \delta$. But this follows from $\alpha \sim_{m_i} \beta$ and $\alpha \mid \gamma \not\sim_{m_i} \delta$. The arguments for the other possible applications of the Sub rules are identical.

That the above conditions hold of the resulting path is now clear. □

We are now in a position to infer the decidability of bisimulation equivalence on CCS processes in standard form. In order to decide the validity of $\alpha = \beta$ we simply start listing tableaux for $\alpha = \beta$ and stop and answer "yes" if a successful tableau has been found. If we list all of the finite number of finite tableaux (systematically, so that we recognise when they have all been listed) and fail to discover a successful one, then we answer "no". By soundness and completeness of the tableau system, we know that this procedure will always give the right answer. Thus the decidability result is established. Combining this with Lemma 2.6 we get the following result.

Theorem 3.6 *Bisimulation equivalence is decidable on* CCS *processes which either do not involve the restriction and relabelling operators or do not involve communication.*

4 An Equational Theory

We now describe a sound and complete equational theory for restriction/relabelling-free or communication-free CCS processes. We restrict attention to processes in standard form, assuming that the axioms presented in Appendix A have been used to establish this. Let

Δ be such a process definition. The theory shall be parameterised by Δ and consists of axioms and inference rules that enable one to derive the roots of successful tableaux.

The theory is in spirit similar to that offered in [17, 16, 7] and is built around sequents of the form $\Gamma \vdash_\Delta E = F$ where Γ is a finite set of *assumptions* of the form $\alpha = \beta$, and E and F are CCS expressions. The semantic interpretation of a sequent $\Gamma \vdash_\Delta E = F$, denoted $\Gamma \models_\Delta E = F$, is as follows: if $\alpha \sim \beta$ for all $(\alpha = \beta) \in \Gamma$, then $E \sim F$. As Δ will remain fixed throughout, we shall omit its subscripted appearance, thus writing $\vdash$ for $\vdash_\Delta$ and $\models$ for $\models_\Delta$. Also we shall omit empty assumption sets, thus writing $\vdash E = F$ and $\models E = F$ for $\emptyset \vdash E = F$ and $\emptyset \models E = F$ respectively. Notice that the relationship $\models E = F$ reduces to $E \sim F$.

The axioms and inference rules are presented in Table 2. We have standard inference rules for equivalence (R1–R3) and congruence (R4–R6). We also have standard axioms for choice (R7–R10) together with standard axioms for parallel composition (R11–R13); notably we have associativity and commutativity for composition. Finally, we have two rules characteristic for this axiomatisation: R14 is an assumption introduction rule underpinning the rôle of the assumption list Γ; and R15 is an assumption elimination rule, and represents a form of fixed point induction. The special form of R15 has been dictated by the rule Rec of the tableau system presented in Table 1. Notice that we do not have an explicit expansion law, as it is incorporated in the assumption elimination rule R15.

Definition 4.1 *A* proof *of* $\Gamma \vdash E = F$*, which we shall denote by that sequent, consists of a proof tree with root labelled* $\Gamma \vdash E = F$*, instances of the axioms* R1 *and* R7–R14 *as leaves and where the father of a set of nodes is determined by an application of one of the inference rules* R2–R6 *or* R15.

Theorem 4.2 (Soundness) *If* $\Gamma \vdash E = F$ *then* $\Gamma \models E = F$*. In particular, if* $\vdash E = F$ *then* $E \sim F$.

Proof: Suppose that $\alpha \sim \beta$ for all $(\alpha = \beta) \in \Gamma$, but that $E \not\sim F$. We shall show that no proof exists for $\Gamma \vdash E = F$.

Suppose then that T is such a proof for $\Gamma \vdash E = F$. We can show that there must be a maximal path $\pi = \{\Gamma_i \vdash E_i = F_i\}$ starting from $\Gamma \vdash E = F$ and leading upwards through T such that $E_i \not\sim F_i$ for all i. This is clear by inspection of the axioms and inference rules. We can furthermore choose π so that the sequence $\{m_i \ : \ E_i \not\sim_{m_i} F_i \text{ and } E_i \sim_{m_i - 1} F_i\}$ is nonincreasing, and strictly decreasing through applications of R4.

The axiom which terminates π, say $\Gamma_l \vdash E_l = F_l$, must be an instance of R14, say of the form $\Gamma', \alpha = \beta \vdash \alpha = \beta$, as otherwise we would have $E_l \sim F_l$.

Since we cannot have $(\alpha = \beta) \in \Gamma$, we must have somewhere in π an application of R15 to eliminate $\alpha = \beta$ from the assumption list. Also, some application of R4 must occur between the axiom and the application of R15, in order for there to be the required guarded expressions on the right of the turnstile at the application of R15. This fact follows from the property that in any sequent $\Gamma \vdash E = F$ of any proof tree, either both E and F are guarded or both E and F are unguarded. Hence the path π is as indicated in Figure 3.

Now we know that $m_l < m_j$; however, this then implies that $\alpha \not\sim_{m_l} \beta$ and $\alpha \sim_{m_l} \beta$, which gives us our required contradiction. $\Box$

Equivalence

R1 $\Gamma \vdash E = E$

R2 $\dfrac{\Gamma \vdash F = E}{\Gamma \vdash E = F}$

R3 $\dfrac{\Gamma \vdash E = F \quad \Gamma \vdash F = G}{\Gamma \vdash E = G}$

Congruence

R4 $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash aE = aF}$

R5 $\dfrac{\Gamma \vdash E_1 = F_1 \quad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 + E_2 = F_1 + F_2}$

R6 $\dfrac{\Gamma \vdash E_1 = F_1 \quad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 \mid E_2 = F_1 \mid F_2}$

Axioms

R7 $\Gamma \vdash E + (F + G) = (E + F) + G$

R8 $\Gamma \vdash E + F = F + E$

R9 $\Gamma \vdash E + E = E$

R10 $\Gamma \vdash E + \mathbf{0} = E$

R11 $\Gamma \vdash E \mid (F \mid G) = (E \mid F) \mid G$

R12 $\Gamma \vdash E \mid F = F \mid E$

R13 $\Gamma \vdash E \mid \mathbf{0} = E$

Assumption Introduction

R14 $\Gamma, \alpha = \beta \vdash \alpha = \beta$

Assumption Elimination

R15 $\dfrac{\Gamma, \alpha = \beta \vdash \mathrm{unf}(\alpha) = \mathrm{unf}(\beta)}{\Gamma \vdash \alpha = \beta}$

Table 2: Axiomatisation.

$$
\begin{array}{lcl}
\text{R14} & \Gamma',\ \alpha=\beta\ \vdash\ \alpha=\beta & (\text{level } l) \\
 & \vdots & \\
\text{R4} & \dfrac{\Gamma',\ \alpha=\beta\ \vdash\ \alpha'=\beta'}{\Gamma',\ \alpha=\beta\ \vdash\ \mu\alpha'=\mu\beta'} & \begin{array}{l}(\text{level } k+1)\\(\text{level } k)\end{array} \\
 & \vdots & \\
\text{R15} & \dfrac{\Gamma'',\ \alpha=\beta\ \vdash\ \text{unf}(\alpha)=\text{unf}(\beta)}{\Gamma''\ \vdash\ \alpha=\beta} & \begin{array}{l}(\text{level } j+1)\\(\text{level } j)\end{array} \\
 & \vdots & \\
 & \Gamma\ \vdash\ E=F & (\text{level } 1)
\end{array}
$$

Figure 3: The path constructed in Lemma 4.2.

For the completeness proof, we introduce the following notation.

Definition 4.3 *For any node* $\mathbf{n}$ *of a tableau,* Recnodes($\mathbf{n}$) *denotes the set of labels of the nodes above* $\mathbf{n}$ *to which the rule* REC *is applied. In particular,* Recnodes($\mathbf{r}$) $= \emptyset$ *where* $\mathbf{r}$ *is the root of the tableau.*

We are now ready to prove our completeness theorem.

Theorem 4.4 (Completeness) *If* $\alpha \sim \beta$ *then* $\vdash \alpha = \beta$.

Proof: If $\alpha \sim \beta$, then there exists a finite successful tableau with root labelled $\alpha = \beta$. Let $T(\alpha = \beta)$ be such a tableau. We shall prove that for any node $\mathbf{n} : E = F$ of $T(\alpha = \beta)$ we have Recnodes($\mathbf{n}$) $\vdash E = F$. In particular, for the root $\mathbf{r} : \alpha = \beta$, this reduces to $\vdash \alpha = \beta$, so we shall have our result.

We prove Recnodes($\mathbf{n}$) $\vdash E = F$ by induction on the depth of the subtableau rooted at $\mathbf{n}$. As the tableau is built modulo the structural congruence $\equiv$ we shall assume that the axioms R7,R8 and R10-R13 are used whenever required to accomplish the proof.

Firstly, if $\mathbf{n} : E = F$ is a terminal node then E and F must be identical terms, so Recnodes($\mathbf{n}$) $\vdash E = F$ follows from R1.

Hence assume that $\mathbf{n} : E = F$ is not a terminal node. We proceed according to the tableau rule applied to $\mathbf{n}$.

PREFIX: Then $E = F$ is of the form $a\gamma = a\delta$. By the induction hypothesis we have Recnodes($\mathbf{n}'$) $\vdash \gamma = \delta$ where $\mathbf{n}'$ is the son of $\mathbf{n}$. By inference rule R4 we therefore conclude that Recnodes($\mathbf{n}'$) $\vdash a\gamma = a\delta$. As Recnodes($\mathbf{n}'$) = Recnodes($\mathbf{n}$) the result follows.

SUM: Then $E = F$ is of the form $\sum a_i\alpha_i = \sum b_j\beta_j$. By the induction hypothesis we have for all sons $\mathbf{n}_k : a_{i_k}\alpha_{i_k} = b_{j_k}\beta_{j_k}$ of $\mathbf{n}$ that Recnodes($\mathbf{n}_k$) $\vdash a_{i_k}\alpha_{i_k} = b_{j_k}\beta_{j_k}$. As Recnodes($\mathbf{n}_k$) = Recnodes($\mathbf{n}$) for all k we get Recnodes($\mathbf{n}$) $\vdash a_{i_k}\alpha_{i_k} = b_{j_k}\beta_{j_k}$. By using rules R5, R7, R8 and R9 we have Recnodes($\mathbf{n}$) $\vdash E = F$ as required.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash X_1 = X_2}\,\text{R14} \qquad \dfrac{}{\Gamma_1 \vdash X_4 = X_4}\,\text{R1}}{\Gamma_1 \vdash X_1 \mid X_4 = X_2 \mid X_4}\,\text{R6} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash X_3 \mid X_4 = X_3 \mid X_4}\,\text{R1}}{\Gamma_2 \vdash a(X_3 \mid X_4) = a(X_3 \mid X_4)}\,\text{R4} \qquad \dfrac{\dfrac{}{\Gamma_2 \vdash X_2 = X_2}\,\text{R1}}{\Gamma_2 \vdash bX_2 = bX_2}\,\text{R4}}{\Gamma_2 \vdash a(X_3 \mid X_4) + bX_2 = a(X_3 \mid X_4) + bX_2}\,\text{R5}}{\Gamma_1 \vdash X_2 \mid X_4 = X_3}\,\text{R15}}{\Gamma_1 \vdash X_1 \mid X_4 = X_3}\,\text{R3}}{\Gamma_1 \vdash a(X_1 \mid X_4) = aX_3}\,\text{R4}}{\vdash X_1 = X_2}\,\text{R15}
$$

Figure 4: A proof of $X_1 = X_2$.

REC: Then $E = F$ is of the form $\alpha = \beta$ and the son $\mathbf{n}'$ of $\mathbf{n}$ is labelled $\mathrm{unf}(\alpha) = \mathrm{unf}(\beta)$. By induction we have $\mathrm{Recnodes}(\mathbf{n}') \vdash \mathrm{unf}(\alpha) = \mathrm{unf}(\beta)$. As $\mathrm{Recnodes}(\mathbf{n}')$ is equal to $\mathrm{Recnodes}(\mathbf{n})$ together with $\alpha = \beta$, by R15 we have $\mathrm{Recnodes}(\mathbf{n}) \vdash E = F$ as required.

SUBL: Say $E = F$ is of the form $\alpha \mid \gamma = \delta$ with the corresponding dominated node $\mathbf{n}'$ labelled $\alpha = \beta$ ($\alpha \sqsupset \beta$) and the son $\mathbf{n}''$ of $\mathbf{n}$ labelled $\beta \mid \gamma = \delta$. By the induction hypothesis we have $\mathrm{Recnodes}(\mathbf{n}'') \vdash \beta \mid \gamma = \delta$. As $\mathrm{Recnodes}(\mathbf{n}'')$ is equal to $\mathrm{Recnodes}(\mathbf{n})$ it follows that $\mathrm{Recnodes}(\mathbf{n}) \vdash \beta \mid \gamma = \delta$. Also, since $(\alpha = \beta) \in \mathrm{Recnodes}(\mathbf{n})$ we have from R14, R6, R1 and R3 that $\mathrm{Recnodes}(\mathbf{n}) \vdash \alpha \mid \gamma = \delta$ as required. The arguments for the other possible applications of the SUB rules are identical.

This completes the proof. □

Example 4.5 *Let* $\left\{X_1 \stackrel{\mathrm{def}}{=} a(X_1 \mid X_4),\ X_2 \stackrel{\mathrm{def}}{=} aX_3,\ X_3 \stackrel{\mathrm{def}}{=} a(X_3 \mid X_4) + bX_2,\ X_4 \stackrel{\mathrm{def}}{=} b\right\}$ *define a* CCS *process in standard form. (This is the process from Example 3.1.) In Figure 4 we give a proof for* $X_1 = X_2$. *In the proof we use the following abbreviations:* $\Gamma_1 = \{X_1 = X_2\}$ *and* $\Gamma_2 = \Gamma_1 \cup \{X_2 \mid X_4 = X_3\}$.

Since CCS contains the regular processes, i.e. processes for which the transition graphs are finite, our theory can be seen as a proper extension of Milner's equational theory for bisimulation equivalence on regular processes. However, as our theory is *sequent based* it is very different from Milner's elegant theory which is build around a few laws for recursion given by an explicit fixed point operator μ [20]. It is not known how to extend Milner's equational theory for regular processes to our class of standard form CCS.

Whether one prefers a theory in the style of Milner's or a sequent based theory is perhaps a matter of taste. One might argue in favour of Milner's theory due to its elegant and natural laws for recursion. On the other hand, the advantage of the sequent based equational theory is that it offers a very natural and direct method of presenting proofs (via the corresponding tableau system). Using Milner's equational theory, it is less straightforward how proofs are built up. Finally, in favour of the sequent based equational theory is the fact that it has proven applicable to a wider range of process classes such as a rich subclass of context-free processes [17] and also standard form CCS as we have demonstrated here.

5 Related Work

The work reported here is based on the results presented in [7, 8], where we consider the restriction-, relabelling- and communication-free subset of CCS, which we termed BPP for *Basic Parallel Processes*. The work is similar in spirit to the work in [9, 17] on context-free processes. However, it is worth noting that the classes of processes which we study are incomparable to the class of context-free processes. For example, in [6] it is demonstrated that the context-free process defined by $X \stackrel{\text{def}}{=} a(Xb + b)$ which generates nonempty strings of the form $a^n b^n$ cannot be expressed by a BPP process; and in [1] it is demonstrated that the BPP process defined by $X \stackrel{\text{def}}{=} a(X \mid b) + c(X \mid d)$ cannot be expressed as a context-free process.

In our work, we have only considered the decidability of bisimilarity. However, many other equivalences have been suggested in the area of process algebra. In [5] it is shown that distributed bisimilarity as defined by Castellani [3] is decidable over BPP. But of more interest are the equivalences within the linear time-branching time spectrum of [11]. In [14] it is demonstrated that language equivalence is undecidable over BPP, as well as the simulation preorder. Based on the techniques presented there, [15] demonstrates the undecidability of all equivalences in the linear time-branching time spectrum over BPP apart from bisimulation. Thus the situation is analogous to that of BPA, for which bisimulation equivalence is decidable [9] but no other equivalence in the spectrum is decidable [18, 13].

Another natural question regards whether we can extend the expressive power of our standard form CCS whilst maintaining a decidable theory for bisimilarity. By relying on Jančar's recent result on the undecidability of bisimilarity on labelled Petri nets [19] we can show that adding a notion of *forced* binary synchronisation will prevent bisimilarity from being decidable. So it still remains to find natural extensions to BPP (for instance by including sequential composition) for which we could demonstrate the decidability of bisimilarity.

In [22] a natural notion of prime decomposition was defined, and the question was raised as to when processes can be uniquely decomposed into prime components; this question can be asked for both sequential composition as well as parallel composition. It is demonstrated in [7] that so-called *normed* BPP processes — those processes which at any point may follow a terminating path — have unique decompositions, and the decidability of bisimulation equivalence over this process class is demonstrated by exploiting a corresponding cancellation lemma. Indeed this property holds similarly for normed BPA processes, and several proofs of the decidability of bisimulation equivalence in this class exploit this property [2, 4, 17, 12]. Of course the results concerning normed BPP and normed BPA are made redundant by the results for the unnormed classes; however, unique decomposability allows for techniques which could be exploited for wider classes of processes such as normed PA, the algebra which includes both sequential and parallel composition.

References

[1] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In Proceedings of ICALP'84, J. Paredaens (ed), *Lecture Notes in Computer Science* 172, pp82–94. Springer-Verlag, 1984.

[2] J.C.M. Baeten, J.A. Bergstra and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In Proceedings of PARLE'87, J.W. de Bakker, A.J. Nijman, P.C. Treleaven (eds), *Lecture Notes in Computer Science* 259, pp93–114. Springer-Verlag, 1987.

[3] I. Castellani. *Bisimulations for Concurrency.* PhD thesis CST-51-88. University of Edinburgh, 1988.

[4] D. Caucal. Graphes canoniques des graphes algébriques. *Informatique Théorique et Applications (RAIRO)* 24(4), pp339–352, 1990.

[5] S. Christensen. Distributed bisimilarity is decidable for a class of infinite-state systems. In Proceedings of CONCUR'92, W.R. Cleaveland (ed), *Lecture Notes in Computer Science* 630, pp148–161. Springer-Verlag, 1992.

[6] S. Christensen. *Decidability and Decomposition in Process Algebras.* PhD thesis CST-105-93. University of Edinburgh, September 1993.

[7] S. Christensen, Y. Hirschfeld and F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In Proceedings of LICS'93. IEEE Computer Society Press, 1993.

[8] S. Christensen, Y. Hirschfeld and F. Moller. Bisimulation equivalence is decidable for basic parallel processes. In Proceedings of CONCUR'93, E. Best (ed), *Lecture Notes in Computer Science* 715, pp143–157. Springer-Verlag, 1993.

[9] S. Christensen, H. Hüttel and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. In Proceedings of CONCUR'92, W.R. Cleaveland (ed), *Lecture Notes in Computer Science* 630, pp138–147. Springer-Verlag, 1992.

[10] R. Cleaveland, J. Parrow, B. Steffen. The Concurrency Workbench: A semantics-based verification tool for finite-state systems. In Proceedings of a workshop on automated verification methods for finite-state systems, *Lecture Notes in Computer Science* 407, Springer-Verlag, 1989.

[11] R.J. van Glabbeek. The linear time-branching time spectrum. In Proceedings of CONCUR'90, J. Baeten, J.W. Klop (eds), *Lecture Notes in Computer Science* 458, pp278–297. Springer-Verlag, 1990.

[12] J.F. Groote. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters* 42, pp167–171, 1991.

[13] J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 1994. *(To appear.)*

[14] Y. Hirshfeld. Petri nets and the equivalence problem. In Proceedings of CSL'93, Swansea, May 1993.

[15] H. Hüttel. Undecidable equivalences for basic parallel processes. Research report ECS-LFCS-93-275. University of Edinburgh, October 1993.

[16] H. Hüttel. *Decidability, Behavioural Equivalences and Infinite Transition Graphs.* PhD thesis CST-86-91. University of Edinburgh, December 1991.

[17] H. Hüttel and C. Stirling. Actions speak louder than words: proving bisimilarity for context-free processes. In Proceedings of LICS'91, pp376–386. IEEE Computer Society Press, 1991.

[18] D.T. Huynh and L. Tian. On deciding readiness and failures equivalences for processes. Research report UTDCS-31-90. University of Texas at Dallas, September 1990.

[19] P. Jančar. Decidability questions for bisimilarity of Petri nets and some related problems. Research report ECS-LFCS-93-261. University of Edinburgh, April 1993.

[20] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences* 28, pp439–466, 1984.

[21] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[22] R. Milner and F. Moller. Unique decomposition of processes. *Journal of Theoretical Computer Science* 107, pp357-363, 1993.

[23] D. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science* 104, pp168–183. Springer-Verlag, 1981.

[24] D. Taubner. Finite representations of CCS and TCSP programs by automata and Petri nets. *Lecture Notes in Computer Science* 369. Springer-Verlag, 1989.

A Proof of the Standard Form Lemma

In this appendix, we demonstrate the validity of Lemma 2.6, that any restriction/relabelling-free or communication-free CCS process can be effectively transformed into standard form. We do this by using a collection of sound axioms to eliminate instances of the relabelling operator in the absence of restriction and communication, and of the restriction operator in the absence of communication. We then point out the transformation from these simpler processes into standard form.

We start by including the recursion axioms from [21].

Fact A.1 *Let $\Delta = \left\{ X_i \stackrel{\text{def}}{=} E_i \;:\; 1 \leq i \leq n \right\}$ be an arbitrary* CCS *process. (Recall that by definition, the X_is are guarded in the E_is.) Then*

- $X_i \sim E_i$ *for $1 \leq i \leq n$.*

- *If* $P_i \sim E_i\{\widetilde{P}/\widetilde{X}\}$ *for each* $1 \le i \le n$ *then* $P_i \sim X_i$ *for each* $1 \le i \le n$.

Lemma A.2 *Let* $\Delta = \left\{X_i \stackrel{\text{def}}{=} E_i \ : \ 1 \le i \le n\right\}$ *be a CCS process which does not involve communication or the restriction operator. Then we can effectively construct an equivalent* CCS *process* Δ' *which also does not involve the relabelling operator.*

Proof: Let $\mathcal{F} = \{f_1, f_2, \ldots, f_k\}$ be the smallest collection of relabelling functions which is closed under composition and contains the identity function $f_1 = \mathtt{id}$ as well as all those appearing in Δ.

Let $\Delta' = \left\{Y_{ij} \stackrel{\text{def}}{=} F_{ij} \ : \ 1 \le i \le n,\ 1 \le j \le k\right\}$ where for each s and t, $F_{st}\left\{X_i[f_j]/Y_{ij}\right\}$ is the term $E_i[f_j]$ in which the following sound axioms have been successively applied as rewrite rules, thus pushing all relabellings down to variable level.

$$\begin{array}{rclcrcl}
\mathbf{0}[f] & \sim & \mathbf{0} & \qquad & (E+F)[f] & \sim & E[f] \ + \ F[f] \\
(aE)[f] & \sim & f(a)E[f] & & (E \mid F)[f] & \sim & E[f] \ \mid \ F[f] \\
 & & & & E[g][f] & \sim & E[g \circ f]
\end{array}$$

Then by Fact A.1, $X_s[f_t] \sim F_{st}\left\{X_i[f_j]/Y_{ij}\right\}$ for each s and t, so $X_s[f_t] \sim Y_{st}$ for each s and t. In particular, $X_1 \sim Y_{11}$. □

(Note that the axiom for distributivity over composition is only assured to be sound in the event that the relabelling function does not create any potential communication actions, which is certainly the case when there are no such actions.)

Lemma A.3 *Let* $\Delta = \left\{X_i \stackrel{\text{def}}{=} E_i \ : \ 1 \le i \le n\right\}$ *be a CCS process which does not involve communication. Then we can effectively construct an equivalent* CCS *process* Δ' *which also does not involve the restriction operator. Hence by the previous lemma we can construct an equivalent* CCS *process which also does not include the relabelling operator.*

Proof: Let $\mathcal{L} = \{L_1, L_2, \ldots, L_k\}$ be the smallest collection of restriction sets which is closed under union and contains the empty set $L_1 = \emptyset$ as well as all those appearing in Δ, and contains $f^{-1}(L)$ whenever it contains L and f appears as a relabelling function in Δ.

Let $\Delta' = \left\{Y_{ij} \stackrel{\text{def}}{=} F_{ij} \ : \ 1 \le i \le n,\ 1 \le j \le k\right\}$ where for each s and t, $F_{st}\left\{X_i \backslash L_j/Y_{ij}\right\}$ is the term $E_i \backslash L_j$ in which the following sound axioms have been successively applied as rewrite rules, thus pushing all relabellings down to variable level.

$$\begin{array}{rclcrcl}
\mathbf{0}\backslash L & \sim & \mathbf{0} & \qquad & (E+F)\backslash L & \sim & E\backslash L \ + \ F\backslash L \\
(aE)\backslash L & \sim & \begin{cases} aE\backslash L, & \text{if } a, \bar{a} \notin L \\ \mathbf{0}, & \text{otherwise} \end{cases} & & (E \mid F)\backslash L & \sim & E\backslash L \ \mid \ F\backslash L \\
E[f]\backslash L & \sim & E\backslash f^{-1}(L)[f] & & E\backslash L\backslash M & \sim & E\backslash(L \cup M)
\end{array}$$

(Again note that the axiom for distributivity over composition is only assured to be sound in the event that the restriction set does not contain any potential communication actions, which is certainly the case when there are no such actions.)

Then by Fact A.1, $X_s\backslash L_t \sim F_{st}\left\{X_i\backslash L_j/Y_{ij}\right\}$ for each s and t, so $X_s\backslash L_t \sim Y_{st}$ for each s and t. In particular, $X_1 \sim Y_{11}$. □

We are now left needing to justify that a relabelling- and restriction-free CCS process can be effectively transformed into standard form. A lengthy proof of this result can be found in [6], and is simply a modification of the Greibach normal form lemma for context-free grammars, where composition is now commutative and permits binary synchronisation, and where the transformation steps are designed to preserve not just language equivalence but bisimilarity as well.