Bisimulation Equivalence is Decidable for Basic Parallel Processes

Søren Christensen Yoram Hirshfeld*

Faron Moller†

Laboratory for Foundations of Computer Science

University of Edinburgh

Abstract

In a previous paper the authors proved the decidability of bisimulation equivalence over two subclasses of recursive processes involving a parallel composition operator, namely the so-called *normed* and *live* processes. In this paper, we extend this result to the whole class. The decidability proof permits us further to present a complete axiomatisation for this class of basic parallel processes. This result can be viewed as a proper extension of Milner's complete axiomatisation of bisimulation equivalence on regular processes.

1 Introduction

Finite-state systems have been extensively studied, both within standard formal language theory and within the theory of process calculi. Certainly all standard behavioural equivalences, in particular those within the linear time-branching time spectrum of [8], are decidable over finite-state systems, and complete axiomatisations have been developed for various of these equivalences between them (see, *eg*, [22, 18, 16]). Furthermore, there have been many automated tools designed for the analysis of such systems (see, *eg*, [17]).

Recently, questions regarding the decidability of process equivalences on various classes of infinite-state systems have been studied. Certainly when we move to context-free processes, language equivalence becomes undecidable. However, of greater interest in process theory are questions regarding stronger notions of equivalence which take into account for instance the notions of deadlock, livelock, or causality. In [14, 10] it is demonstrated that all of the standard behavioural equivalences besides bisimilarity are undecidable over context-free processes. However in [7] it is shown that bisimilarity is in fact decidable over this class of infinite-state processes. Previous to this, there were several proofs of this result for the subclass of *normed* processes, those processes which may terminate in a finite number of steps at any point during their execution ([1, 3, 13, 9]).

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

†Supported by ESPRIT BRA 7166: CONCUR2.

The class of context-free processes is provided by a standard process calculus which admits of a general sequencing operator, along with atomic actions and choice. However of greater importance in process theory is the inclusion of some form of parallel combinator. Very little work it seems has been spent on exploring the decidability (or otherwise) of equivalences defined over process calculi which include parallel combinators within recursive definitions. Certainly with very little else you can express the full power of Turing machines along with their undecidable problems [19].

In [6] we explored a calculus which includes a simple parallel combinator. We showed that in the case of normed processes, and also for *live* processes (processes which can never perform an infinite number of identical actions in succession uninterrupted), we can prove decidability of bisimulation equivalence. In order to obtain our results, we required cancellation laws for normed and live processes. Such a law is not valid over the whole calculus, so the proof technique presented there is not valid in general.

In this paper, we demonstrate a refinement of the argument presented in [6] which settles the decidability of bisimulation equivalence over the whole calculus. As with [6], the technique which we use to provide the decidability result is based on the use of *tableau systems.* From the rules we present for generating our tableaux we can extract a sound and complete equational theory for our calculus. As our class of processes clearly contains the regular processes this result can be viewed as a proper extension to Milner's equational theory of bisimulation equivalence on regular processes [18].

2 Preliminaries

We presuppose a countably infinite set of *atomic actions* $\Lambda = \{a, b, c, \ldots\}$ as well as a countably infinite set of *process variables* $Var = \{X, Y, Z, \ldots\}$. The class of recursive BPP (Basic Parallel Processes) expressions is defined by the following abstract syntax equation.

$$\begin{array}{rcll} E & ::= & 0 & \text{(inaction)} \\ & | & X & \text{(process variable, } X \in Var\text{)} \\ & | & aE & \text{(action prefix, } a \in \Lambda\text{)} \\ & | & E + E & \text{(choice)} \\ & | & E \| E & \text{(merge)} \end{array}$$

We shall omit trailing 0s from expressions, thus writing the term $a0$ simply as a. Also we shall write E^n to represent the term $E \| \cdots \| E$ consisting of n copies of E combined in parallel.

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

$$\Delta = \left\{ X_i \overset{\text{def}}{=} E_i \mid 1 \leq i \leq n \right\}$$

where the X_i are distinct and the E_i are BPP expressions at most containing the variables $Var(\Delta) = \{X_1, \ldots, X_n\}$. We further assume that every variable occurrence in the E_is are *guarded*, that is, appear within the scope of an action prefix. The variable X_1 is singled out as the *leading variable* and $X_1 = E_1$ is called the *leading equation.*

Any finite family Δ of BPP equations determines a labelled transition system. The transition relations are given as the least relations satisfying the following rules.

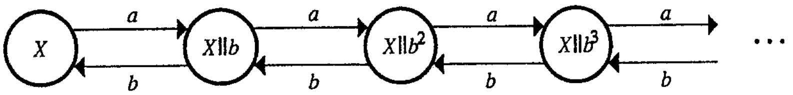

Figure 1: The transition graph for $\left\{X \overset{\text{def}}{=} a(X\|b)\right\}$.

$$aE \xrightarrow{a} E \qquad \frac{E \xrightarrow{a} E'}{E+F \xrightarrow{a} E'} \qquad \frac{E \xrightarrow{a} E'}{E\|F \xrightarrow{a} E'\|F}$$

$$\frac{E \xrightarrow{a} E'}{X \xrightarrow{a} E'}(X \overset{\text{def}}{=} E \in \Delta) \qquad \frac{F \xrightarrow{a} F'}{E+F \xrightarrow{a} F'} \qquad \frac{F \xrightarrow{a} F'}{E\|F \xrightarrow{a} E\|F'}$$

Strictly speaking, transitions are defined on BPP 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* BPP *processes generate finite branching transition graphs, that is, graphs for which the set* $\{F : E \xrightarrow{a} F\}$ *is finite for each* E *and each* a. *This would not be true if we allowed unguarded expressions. For example, the process* $X \overset{\text{def}}{=} a + a\|X$ *generates an infinite-branching transition graph.*

Remark 2.2 *For the purpose of this presentation, we only consider pure merge for our parallel combinator. However, it is easily seen that the results of this paper hold (with the obvious slight modifications) if we allow handshake communication in the style of* CCS. *Hence the calculus we are considering is (guarded)* CCS *without restriction and relabelling.*

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

Definition 2.3 *Let* $\equiv$ *be the smallest congruence relation over process expressions such that the laws of associativity, commutativity and* 0*-absorption hold for choice and merge.*

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

The equivalence between BPP expressions (states) which we are interested in considering here is bisimilarity [19], defined as follows.

Definition 2.5 *A binary relation $\mathcal{R}$ over* BPP *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 BPP process by forming the product of the elements of α, i.e. by combining the elements of α in parallel using the merge operator. We recognise the empty product as $\mathbf{0}$, 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 merge.

Definition 2.6 *A finite family $\Delta = \{X_i \overset{\text{def}}{=} E_i \mid 1 \leq i \leq n\}$ of guarded* BPP *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 for each j we have $\alpha_j \in Var(\Delta)^{\otimes}$. 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 [19] it is shown that any finite family Δ of guarded BPP equations has a unique solution up to bisimilarity. Moreover, in [5] we have the following result showing that any such system can be effectively presented in standard form.

Lemma 2.7 *Given any finite family of guarded* BPP *equations Δ we can effectively construct another finite family of* BPP *equations Δ' in standard form in which $\Delta \sim \Delta'$, i.e. the leading variables of Δ and Δ' are bisimilar.*

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

Definition 2.8 *By $\sqsubset$ we denote the well-founded ordering on $Var(\Delta)^{\otimes}$ given as follows:*

$$X_1^{k_1} \| \cdots \| X_n^{k_n} \quad \sqsubset \quad X_1^{l_1} \| \cdots \| X_n^{l_n}$$

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 $\beta\|\gamma \sqsubset \alpha\|\gamma$ 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 $\Delta = \{X_i \overset{\text{def}}{=} E_i \mid 1 \leq i \leq n\}$ of guarded BPP 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 [13]. 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 BPP expressions. Each rule has the form

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

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 unf(α) to mean the *unfolding* of α defined as follows: given $Y_i = \sum_{j=1}^{n_i} a_{ij}\alpha_{ij}$ for $1 \leq i \leq m$,

$$\text{unf}(Y_1 \| \cdots \| Y_m) = \sum_{i=1}^{m}\sum_{j=1}^{n_i} a_{ij}\big(Y_1 \| \cdots \| Y_{i-1} \| \alpha_{ij} \| Y_{i+1} \| \cdots \| Y_m\big).$$

We shall identify BPP expressions in our tableaux up to the structural congruence $\equiv$, i.e. up to associativity, commutativity and 0-absorption of choice and merge. In particular, we always assume that the labels of nodes have been pruned of $\mathbf{0}$ components sitting in parallel or in sum; rule REC might introduce such innocent components.

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 $\mathbf{n} : E = F$.

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 $\alpha = \alpha$, while an unsuccessful terminal node is one labelled either $a\alpha = b\beta$ such that $a \neq b$ or $a\alpha = \mathbf{0}$ or $\mathbf{0} = b\beta$. A tableau is successful if and only if all terminal nodes are successful; otherwise it is unsuccessful.

Tableaux are built from *basic steps*. A basic step for $\alpha = \beta$ consists of an application of REC to $\alpha = \beta$ followed (possibly) by an application of SUM followed by an application of PREFIX to each of its consequents. See Figure 2 for the schema of a basic step. A

$$\begin{array}{lccc c}
\text{REC} & & \dfrac{\alpha = \beta}{\sum a_i\alpha_i = \sum b_i\beta_i} & & \\
\text{SUM} & \multicolumn{3}{c}{\overline{\hspace{12em}}} & \\
\text{PREFIX} & \dfrac{a_1\alpha_1 = b_1\beta_1}{\alpha_1 = \beta_1} & \cdots & \dfrac{a_n\alpha_n = b_n\beta_n}{\alpha_n = \beta_n} & \text{PREFIX}
\end{array}$$

Figure 2: A basic step.

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 $f : \{1,\ldots,n\} \to \{1,\ldots,m\}$ $g : \{1,\ldots,m\} \to \{1,\ldots,n\}$
PREFIX	$\dfrac{a\alpha = a\beta}{\alpha = \beta}$	
SUBL	$\dfrac{\alpha\|\gamma = \delta}{\beta\|\gamma = \delta}$	if the dominated node is labelled $\alpha = \beta$ or $\beta = \alpha$ with $\alpha \sqsupset \beta$
SUBR	$\dfrac{\delta = \alpha\|\gamma}{\delta = \beta\|\gamma}$	if the dominated node is labelled $\alpha = \beta$ or $\beta = \alpha$ with $\alpha \sqsupset \beta$

Table 1: Rules of the tableau system.

basic step represents a set of single transition steps in the operational semantics: for each consequent $\alpha_i = \beta_i$ we have $\alpha \xrightarrow{a_i} \alpha_i$ and $\beta \xrightarrow{a_i} \beta_i$.

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 $\mathbf{n} : \alpha\|\gamma = \delta$ or $\mathbf{n} : \delta = \alpha\|\gamma$ 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.

Example 3.1 *Let* $\{X_1 \overset{\mathrm{def}}{=} a(X_1\|X_4), X_2 \overset{\mathrm{def}}{=} aX_3, X_3 \overset{\mathrm{def}}{=} a(X_3\|X_4) + bX_2, X_4 \overset{\mathrm{def}}{=} b\}$ *be a family of* BPP *processes in standard form. In Figure 3 we give a successful tableau for* $X_1 = X_2$. *Notice that these processes are neither normed nor live, so the techniques described in [6] are inapplicable.*

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

$$
\begin{array}{lcr}
 & \begin{array}{c} X_1 = X_2 \\ \hline a(X_1\|X_4) = aX_3 \\ \hline X_1\|X_4 = X_3 \\ \hline X_2\|X_4 = X_3 \end{array} & \begin{array}{l} \\ \textsc{Rec} \\ \textsc{Prefix} \\ \textsc{SubL} \end{array} \\
\textsc{Rec} & \overline{a(X_3\|X_4) + bX_2 = a(X_3\|X_4) + bX_2} & \\
\textsc{Sum} & \begin{array}{cc} \hline a(X_3\|X_4) = a(X_3\|X_4) & bX_2 = bX_2 \end{array} & \\
\textsc{Prefix} & \begin{array}{cc} \hline X_3\|X_4 = X_3\|X_4 & \quad X_2 = X_2 \end{array} & \textsc{Prefix}
\end{array}
$$

Figure 3: A successful tableau for $X_1 = X_2$.

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

Consider the infinite sequence $\{\bar{u}_i(1)\}_{i=1}^{\infty}$ 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 $\{\bar{v}_i\}_{i=1}^{\infty}$ with the property that the first coordinate of $\bar{v}_i$ remains constant throughout S_1. If the sequence $\{\bar{u}_i(1)\}_{i=1}^{\infty}$ does not have an upper bound we extract from S an infinite sequence S_1 of vectors $\{\bar{v}_i\}_{i=1}^{\infty}$ with the property that the first coordinate of $\bar{v}_i$ is nondecreasing, i.e. $\bar{v}_i(1) \leq \bar{v}_j(1)$ whenever $i \leq j$. Continuing in this fashion we arrive at an infinite sequence S_{2n} of vectors $\{\bar{w}_i\}_{i=1}^{\infty}$ 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 merge [19] and that forward soundness of the SUB rules follows from the fact that bisimilarity is a congruence wrt merge. □

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 (see for instance [19]) that for finite branching transition graphs, bisimulation is given as the limit of the above approximations:

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

As noted in Remark 2.1, BPP 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 $\{m_i : E_i \not\sim_{m_i} F_i$ and $E_i \sim_j F_i$ 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 $E_{i+1} \not\sim_{m_i} F_{i+1}$ and $E_{i+1} \sim_j F_{i+1}$ for all $j < m_i$, so $m_{i+1} = m_i$.
- 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 $\alpha\|\gamma = \delta$ 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 $\mathbf{n}_{i+1} : \beta\|\gamma = \delta$, and show that we have some valid $m_{i+1} \leq m_i$, that is, that $\beta\|\gamma \not\sim_{m_i} \delta$. But this follows from $\alpha \sim_{m_i} \beta$ and $\alpha\|\gamma \not\sim_{m_i} \delta$. 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 BPP processes. 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.

Theorem 3.6 *Bisimulation equivalence is decidable on* BPP *processes.*

4 An Equational Theory

We now describe a sound and complete equational theory for BPP processes. We restrict attention to processes in standard form. Let Δ be a finite family of such processes. The theory shall be parameterised by Δ and consists of axioms and inference rules that enable one to derive the root of successful tableaux.

The theory is in spirit similar to that offered in [13, 12, 6] 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 BPP expressions. The semantical 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 merge (R11–R13); notably we have associativity and commutativity for merge. 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.

Equivalence

R1 $\Gamma \vdash E = E$

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

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

Congruence

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

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

R6 $$\frac{\Gamma \vdash E_1 = F_1 \quad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 \| E_2 = F_1 \| F_2}$$

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 $\Gamma \vdash E \| (F \| G) = (E \| F) \| G$ R12 $\Gamma \vdash E \| F = F \| E$

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

Assumption Introduction

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

Assumption Elimination

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

Table 2: Axiomatisation.

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

Figure 4: The path constructed in Lemma 4.2.

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 $\{m_i : E_i \not\sim_{m_i} F_i$ 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 4.

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. □

For the completeness proof, we introduce the following notation.

Definition 4.3 *For any node* n *of a tableau,* Recnodes(n) *denotes the set of labels of the nodes above* n *to which the rule* Rec *is applied. In particular,* Recnodes(r) $= \emptyset$ *where* 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(**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(**n**) $\vdash E = F$ by induction on the depth of the subtableau rooted at **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(**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 **n**.

PREFIX: Then $E = F$ is of the form $a\gamma = a\delta$. By the induction hypothesis we have Recnodes(**n**$'$) $\vdash \gamma = \delta$ where **n**$'$ is the son of **n**. By inference rule R4 we therefore conclude that Recnodes(**n**$'$) $\vdash a\gamma = a\delta$. As Recnodes(**n**$'$) = Recnodes(**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 **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(**n**) for all k we get Recnodes(**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(**n**) $\vdash E = F$ as required.

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

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

This completes the proof. □

Example 4.5 *Let* $\{X_1 \stackrel{\text{def}}{=} a(X_1\|X_4), X_2 \stackrel{\text{def}}{=} aX_3, X_3 \stackrel{\text{def}}{=} a(X_3\|X_4) + bX_2, X_4 \stackrel{\text{def}}{=} b\}$ *be a family of* BPP *processes in full standard form. (This is the family of processes from Example 3.1.) In Figure 5 we give a proof for* $X_1 = X_2$. *In the proof we use the following abbreviations:* $\Gamma_1 = \{X_1 = X_2\}$ *and* $\Gamma_2 = \Gamma_1 \cup \{X_2\|X_4 = X_3\}$.

Since BPP 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 (see [18]). It is not known how to extend Milner's theory for regular processes to the class BPP.

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

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

Figure 5: A proof of $X_1 = X_2$.

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 normed context-free processes (see [13]) and also BPP as we have demonstrated here.

5 Related Work

The work reported here is similar in spirit to the work in [7, 13] 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 [5] it is demonstrated that the context-free process defined by $X \overset{\text{def}}{=} a(Xb + b)$ which generates nonempty strings of the form $a^n b^n$ cannot be expressed in our calculus BPP; and the BPP process defined by

$$X \overset{\text{def}}{=} a(b\|c\|X + b\|c) + b(a\|c\|X + a\|c) + c(a\|b\|X + a\|b)$$

which generates all nonempty strings over the alphabet $\{a, b, c\}$ which contain equal numbers of a's, b's and c's cannot be expressed as a context-free process. It would be interesting to combine the two calculi into one which admits both general sequencing and parallelism.

We have only considered decidability on BPP of bisimilarity. However, many more equivalences have been suggested in the area of process algebra. In [4] it is shown that distributed bisimilarity as defined by Castellani (see [2]) is decidable on BPP. But of more interest are the equivalences within the linear time-branching time spectrum of [8]. It would be nice to have a picture as complete as that for BPA where bisimilarity is decidable while all the other equivalences are undecidable. A very recent result by Hirshfeld demonstrates that language equivalence is undecidable on BPP [11]. Using this result it might be possible to show (some of) the other equivalences within the linear time-branching time spectrum to be undecidable (this is the case for BPA where the well-known undecidability of language equivalence is reduced to a number of the other equivalences [10]).

Another natural question regards whether we can extend the expressive power of BPP while maintaining a decidable theory for bisimilarity. Certainly, by replacing the merge operator with CCS's parallel combinator (thus allowing for synchronisation on complementary ports) we still have decidability of bisimilarity; the arguments are more or less as presented here and the details appear in [5]. However, by relying on Jančar's recent result on the undecidability of bisimilarity on labelled Petri nets [15] we can show that adding a notion of *forced* binary synchronisation on top of BPP will prevent bisimilarity from being decidable. So it still remains to find natural extensions to BPP (besides including sequential composition) for which we should start searching for decidable theories of bisimilarity (or otherwise).

References

[1] 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.

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

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

[4] 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.

[5] S. Christensen. Forthcoming PhD thesis. University of Edinburgh, 1993.

[6] S. Christensen, Y. Hirschfeld and F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In Proceedings of LICS93. IEEE Computer Society Press, 1993.

[7] 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.

[8] 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.

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

[10] J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. Research report ECS-LFCS-91-169. University of Edinburgh, August 1991.

[11] Y. Hirshfeld. Finitely generated processes, Petri nets and the equivalence problem. Unpublished notes, University of Edinburgh, May 1993.

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

[13] 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.

[14] 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.

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

[16] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In Proceedings of LICS 91, pp214–225. IEEE Computer Society Press, 1991.

[17] Madelaine, E., Verification tools from the CONCUR project. *Bulletin of the European Association of Theoretical Computer Science* 47, pp110-126, June 1992.

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

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

[20] R. Milner and F. Moller. Unique decomposition of processes. *Bulletin of the European Association for Theoretical Computer Science* 41, pp226–232, 1990.

[21] F. Moller. *Axioms for Concurrency.* PhD thesis CST-59-89. University of Edinburgh, 1989.

[22] A. Salomaa, Two complete axiom systems for the algebra of regular events. *Journal of the Association of Computing Machinery* 13, pp158–169, 1966.