Actions Speak Louder than Words: Proving Bisimilarity for Context-Free Processes

Hans Hüttel* Colin Stirling†

Laboratory for Foundations of Computer Science

James Clerk Maxwell Building

University of Edinburgh

Edinburgh EH9 3JZ

SCOTLAND

Abstract

Recently Baeten, Bergstra, and Klop have proved the remarkable result that bisimulation equivalence is decidable for irredundant context-free grammars. In this paper we provide a much simpler and much more direct proof of this result using a tableau decision method involving goal-directed rules. The decision procedure yields an upper bound on the depth of a tableau. Moreover, it provides the essential part of the bisimulation relation between two processes which underlies their equivalence. A second virtue is that it provides a sound and complete equational theory for such processes.

1 Introduction

In [1] (and [2]) Baeten, Bergstra, and Klop prove the remarkable result that bisimulation equivalence is *decidable* for irredundant context-free grammars (without the empty production). Within process calculus theory these grammars correspond to *normed* processes defined by a finite family of guarded recursion equations in the signature of BPA (Basic Process Algebra) [4]. These processes can have infinitely many states (even after quotienting by bisimulation equivalence). Consequently the process calculus approach (as exemplified in [12]) encompasses a much richer class of infinite-state systems that are open to automatic techniques normally associated with finite state systems than all those approaches based on trace, or language, equivalence. Recently, Huynh and Tian [9] have shown that failures and readiness equivalences are *undecidable* for this class of processes. This suggests a new criterion for distinguishing between the computational qualities of the many behavioural equivalences that have been suggested. Moreover, it may throw new light on the classification of equivalences into linear and branching.

However, the proof of decidability in [1, 2] is not easy as it relies on isolating a possibly complex periodicity from the transition graphs of these processes. An alternative, more elegant, proof utilizing rewrite techniques is presented by Caucal [6]. The idea is to show that the maximal bisimulation on a transition graph is given as the least congruence of a canonical and strongly normalizing Thue system and that there are only finitely many candidates for such a system. However, the proof is still very complex and the decision procedure consists of a linear search for the desired Thue system. Neither of the proofs reflects how one

*E-mail: hans@lfcs.ed.ac.uk

†E-mail: cps@lfcs.ed.ac.uk

intuitively would show that two processes are bisimilar.

In this paper we first provide a much simpler and much more direct proof of the decidability result using a *tableau* decision method involving goal-directed rules; the second author has used this technique in the different context of local model checking finite and infinite state transition systems [13, 5]. The decision procedure yields an upper bound on the depth of a tableau. Moreover, it provides the essential part of the bisimulation relation between two processes which underlies their equivalence, which is also a self-bisimulation in the sense of [6]. One virtue is that the technique could be easily implemented. A second virtue is that it provides a sound and complete equational theory for such processes, albeit in a non-standard form: this theory emanates from 'running the tableau method backwards'. We see this as an initial step in extending Milner's axiomatization of regular processes [11] to context-free processes. At the same time it offers an alternative method for designing equational theories which does not depend on appealing to normal forms.

Preliminaries are dealt with in Section 2. In Section 3 we give the tableau decision method while in Section 4 we present the resulting sound and complete equational theory for these normed context-free processes.

2 Preliminaries

2.1 Normed recursive BPA processes

We consider the class of guarded recursive normed BPA (Basic Process Algebra) processes (see e.g. [2, 4]). BPA process expressions are given by the abstract syntax

$$E ::= a \mid X \mid E_1 + E_2 \mid E_1 E_2$$

Here a ranges over a set of atomic actions, and X over a family of variables. The operator $+$ is nondeterministic choice while E_1E_2 is the sequential composition of E_1 and E_2. A process is defined by a finite family Δ of recursive process equations

$$\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid 1 \leq i \leq k\}$$

where the X_i are distinct, and the E_i are guarded BPA expressions with free variables in $Var = \{X_1, \ldots, X_m\}$. A process expression is *guarded* if every variable occurrence is within the scope of an atomic action. In the sequel we let $X, Y, \ldots$ range over variables in Var and $\alpha, \beta, \ldots$ over finite length sequences of variables.

Given a family Δ we define its transition behaviour by the following rules (ϵ denotes the empty process with the expected convention that ϵF is just F):

$$\frac{E \stackrel{a}{\rightarrow} E'}{E + F \stackrel{a}{\rightarrow} E'} \qquad \frac{F \stackrel{a}{\rightarrow} F'}{E + F \stackrel{a}{\rightarrow} F'}$$

$$\frac{E \stackrel{a}{\rightarrow} E'}{EF \stackrel{a}{\rightarrow} E'F} \qquad a \stackrel{a}{\rightarrow} \epsilon \quad a \in Act$$

$$\frac{E \stackrel{a}{\rightarrow} E'}{X \stackrel{a}{\rightarrow} E'} \qquad X \stackrel{\text{def}}{=} E \in \Delta$$

A simple extension is the transitive closure of the relations $\{\stackrel{a}{\rightarrow} \mid a \in Act\}$: for $w \in Act^+$ we write $p \stackrel{aw}{\rightarrow} q$ if $p \stackrel{a}{\rightarrow} p'$ and $p' \stackrel{w}{\rightarrow} q$ for some p'.

The important extra restriction on a family Δ is *normedness*. For each variable X_i we assume that there is a w such that $X_i \stackrel{w}{\rightarrow} \epsilon$, and as in [2] we define the *norm* of X_i, $|X_i|$, to be the length of the least such w, i.e. $|X_i| = \min\{length(w) \mid X_i \stackrel{w}{\rightarrow} \epsilon\}$. The norm is easily computed: we have $|a| = 1$, $|p + q| = \min(|p|, |q|)$, and $|pq| = |p| + |q|$.

Example 1 Consider the pair $X \stackrel{\text{def}}{=} a + bXY$; $Y \stackrel{\text{def}}{=} c$. Here $|X| = |Y| = 1$. By the transition rules above X generates the transition system in Figure 1. □

Because of the normedness restriction, this family of processes does not include the regular processes (such as $X \stackrel{\text{def}}{=} aX$). Nevertheless, it is a very rich family, and

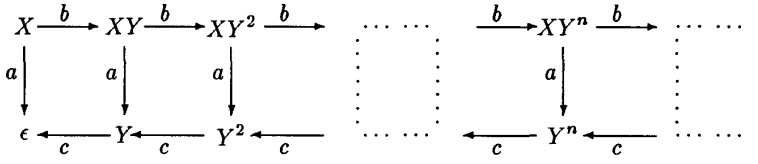

Figure 1: Transition graph for $X \stackrel{\text{def}}{=} a + bXY$; $Y \stackrel{\text{def}}{=} c$ (Example 1)

as illustrated in Figure 1, it contains processes that can have infinitely many states even after quotienting by bisimulation equivalence.

The language $L(X_i)$ accepted by a variable X_i is the set $\{w \mid X_i \stackrel{w}{\rightarrow} \epsilon\}$. For instance, in Example 1 above $L(Y) = \{c\}$ whereas $L(X) = \{b^n a c^n \mid n \geq 0\}$.

2.2 Bisimulation equivalence

Within process calculus theory a variety of equivalences have been proposed in order to capture when two processes may be said to exhibit the same behaviour. The most notable is bisimulation equivalence, as utilized in [12] for example.

Definition 1 *A relation R between processes is a* bisimulation *if whenever pRq then for each $a \in Act$*

1. $p \stackrel{a}{\rightarrow} p' \Rightarrow \exists q' : q \stackrel{a}{\rightarrow} q' \wedge p' R q'$
2. $q \stackrel{a}{\rightarrow} q' \Rightarrow \exists p' : p \stackrel{a}{\rightarrow} p' \wedge p' R q'$

Two processes p and q are said to be bisimulation equivalent *or* bisimilar, *written $p \sim q$, if there is a bisimulation R such that pRq.*

Not only is $\sim$ an equivalence relation, but it is also a congruence relation w.r.t. the process constructs of BPA [4].

Example 2 For an example of bisimilar BPA process expressions take the family $\{X \stackrel{\text{def}}{=} aYX + b, Y \stackrel{\text{def}}{=} bX, A \stackrel{\text{def}}{=} aC + b, C \stackrel{\text{def}}{=} bAA\}$. We have that $X \sim A$; the reader may want to verify that the relation $\{(X^n, A^n) \mid n \geq 0\} \cup \{(YX^{n+1}, CA^n) \mid n \geq 0\}$ is a bisimulation (where X^n here denotes n successive Xs, $X \in V$). □

The following proposition is essential, providing us with a way of removing suffixes of bisimilar BPA expressions.

Proposition 1 *If $EG \sim FG$ then $E \sim F$.*

PROOF: Suppose $EG \sim FG$. Then $\{(E, F) \mid EG \sim FG\}$ is a bisimulation. □

Note that the proof relies on the assumed normedness; a simple counterexample for the unnormed case is that the family $\{X \stackrel{\text{def}}{=} aX, Y \stackrel{\text{def}}{=} a\}$. Here $YYX \sim YX$ but clearly $YY \not\sim Y$.

2.3 Normed recursive BPA processes in Greibach Normal Form

Any family Δ of guarded equations has a unique solution up to bisimulation equivalence [3]. Moreover, in [2] it is shown that any guarded system of equations can be effectively presented in a normal form (2.3), using a family Δ' of the form

$$\{X_i \stackrel{\text{def}}{=} \sum_{j=1}^{n_i} a_{ij}\alpha_{ij} \mid 1 \leq i \leq m\}$$

where each variable sequence α_{ij} has length of at most two. This normal form preserves bisimulation equivalence. Moreover, when Δ only contains normed processes, so does Δ'.

This normal form is called *3-GNF* (3-Greibach Normal Form) by analogy with context-free grammars (without the empty production). There is an obvious correspondence between variables and non-terminals and actions and terminals, and each equation $X_i \stackrel{\text{def}}{=} \sum_{j=1}^{n_i} a_{ij}\alpha_{ij}$ can be viewed as the family of productions $\{X_i \rightarrow a_{ij}\alpha_{ij} \mid 1 \leq j \leq n_i\}$. Normedness corresponds to irredundancy in the grammar. Conversely, any context-free language (without the empty string) is generated by such a grammar (where some variable is designated as the start symbol).

When the variables of two families in 3-GNF Δ and Δ' are disjoint, the family $\Delta \cup \Delta'$ also defines a set of normed processes in normal form. Consequently in general the question of whether $L(X) = L(Y)$ when X, Y are variables in a family Δ is undecidable. However, Baeten, Bergstra and Klop show that despite this the question of whether $X \sim Y$ is decidable. We now present a much simpler proof of this.

3 The tableau decision method

Assume a fixed family Δ of normed 3-GNF equations $\{X_i \stackrel{\text{def}}{=} \sum_{j=1}^{n_i} a_{ij}\alpha_{ij} \mid 1 \leq i \leq m\}$. The bisimulation checker we now present for testing whether $X\alpha \sim Y\beta$ (when all these variables are defined in Δ) is a *tableau system*, a goal-directed proof system. Our proof technique turns out to be similar to the algorithm used in [10] to show that language equivalence of simple context-free grammars is decidable. The rules are built around equations $E\alpha = F\beta$ (where E, F could be the empty sequence of variables). Each rule has the form

$$\frac{E\alpha = F\beta}{E_1\alpha_1 = F_1\alpha_1 \quad \cdots \quad E_n\alpha_n = F_n\alpha_n}$$

(possibly with side conditions). The premise of a rule represents the goal to be achieved (that $E\alpha \sim F\beta$) while the consequents are the subgoals.

We determine whether $X\alpha \sim Y\beta$ by constructing a tableau using the rules presented in Table 1.

A *tableau* for $X\alpha = Y\beta$ is a maximal finite proof tree whose root is labelled $X\alpha = Y\beta$ and where the equations labelling the immediate successors of a node are determined by an application of one of the rules.

The rules are only applied to nodes that are not *terminal*. Terminal nodes are either *successful* or *unsuccessful*. A tableau node is called an *unsuccessful terminal* if it has one of the forms

1. $\alpha = \beta$ with $|\alpha| \neq |\beta|$
2. $a\alpha = b\beta$ with $a \neq b$

Clearly, such nodes cannot relate bisimilar processes. The definition of success is delayed until later.

3.1 Constructing subtableaux

A tableau consists of a number of *eliminating subtableaux*. The rules REC, SUM, and PREFIX are used to construct the subtableaux. Each of these rules is *backwards sound* in that if the consequents are true (their equations relate bisimilar processes) then so is the antecedent.

A subtableau is built from *basic steps*. A basic step for $X\alpha = Y\beta$ is as in Figure 2: it involves one application of REC followed by at most one application of SUM followed by an application of PREFIX to each of its consequents: we assume here that no node in this tree is an unsuccessful terminal. Corresponding to a basic step is a set of single transition steps in the operational semantics, as $X\alpha \stackrel{a_i}{\rightarrow} \alpha_i$ and $Y\beta \stackrel{a_i}{\rightarrow} \beta_i$. An important observation is that $length(\alpha_i) \leq 1 + length(X\alpha)$ as X can only introduce a sequence of at most two variables via REC, and similarly $length(\beta_i) \leq length(Y\beta) + 1$.

Assume that $k = \min(|X|, |Y|)$. An eliminating subtableau for $X\alpha = Y\beta$ iterates the construction of basic steps by repeatedly building to depth k these steps and can be pictured as in Figure 3 when $|X| \leq |Y|$. If $\alpha' = \beta'$ is a leaf of an eliminating subtableau then $X\alpha \stackrel{w}{\rightarrow} \alpha'$ and $Y\beta \stackrel{w}{\rightarrow} \beta'$ for some w of length k. By backwards soundness if the leaves of an eliminating subtableau are sound then so is the root.

Rules within subtableaux

REC $\quad \dfrac{X\alpha = Y\beta}{E\alpha = F\beta}$ where $X \stackrel{\text{def}}{=} E$ and $Y \stackrel{\text{def}}{=} F$

PREFIX $\quad \dfrac{a\alpha = a\beta}{\alpha = \beta}$

SUM $\quad \dfrac{(\sum_{i=1}^{m} a_i\alpha_i)\alpha = (\sum_{j=1}^{n} b_j\beta_j)\beta}{\{a_i\alpha_i\alpha = b_{f(i)}\beta_{f(i)}\beta\}_{i=1}^{m}\ \{a_{g(j)}\alpha_{g(j)}\alpha = b_j\beta_j\beta\}_{j=1}^{n}}$

$$\begin{array}{rl} & f : \{1,\ldots,m\} \to \{1,\ldots,n\} \\ \textit{where} & g : \{1,\ldots,n\} \to \{1,\ldots,m\} \\ & \textit{with } m,n \geq 1 \end{array}$$

Rules for new subtableaux

SUBL $\quad \dfrac{\alpha_i\alpha = \beta_i\beta}{\alpha_i\gamma = \beta_i}$ where $\alpha = \gamma\beta$ is the residual

SUBR $\quad \dfrac{\alpha_i\alpha = \beta_i\beta}{\alpha_i = \beta_i\gamma}$ where $\gamma\alpha = \beta$ is the residual

Table 1: The tableau rules

In the case that $|X| \leq |Y|$ each leaf of an eliminating subtableau for $X\alpha = Y\beta$ is either labelled $\alpha = \gamma\beta$, which we call a *residual* of the subtableau, as X is eliminated, or $\alpha_i\alpha = \beta_i\beta$ where α_i and β_i need not be empty. Since the number of iterations of basic steps is $|X|$ there must be at least one residual and α and β must persist as suffixes throughout the subtableau. For any such subtableau we pick one residual node and call it *the* residual. If instead $|Y| < |X|$ similar remarks would apply except that the residual then has the form $\gamma\alpha = \beta$.

The next step is to apply one of the SUB rules of Table 1 to each leaf *other than the residual* of an eliminating subtableau. If the residual is $\alpha = \gamma\beta$ we apply SUBL, and if it is $\gamma\alpha = \beta$ we apply SUBR. So assume $|X| \leq |Y|$ then for each leaf $\alpha_i\alpha = \beta_i\beta$ which is not the residual we obtain

$$\frac{\alpha_i\alpha = \beta_i\beta}{\alpha_i\gamma = \beta_i}\ \text{SUBL} \qquad \alpha = \gamma\beta \text{ is the residual}$$

(If instead $\gamma\alpha = \beta$ is the residual SUBR gives us the consequent $\alpha_i = \beta_i\gamma$). The SUB rules are also backwards sound: for example, in the case of SUBL if $\alpha_i\gamma \sim \beta_i$ and $\alpha \sim \gamma\beta$ then $\alpha_i\alpha \sim \beta_i\beta$. The SUB rules should be thought of as two-step rules consisting of a *substitution* using the residual followed by a *reduction*

$$\dfrac{\dfrac{X\alpha = Y\beta}{E\alpha = F\beta}\ \textsf{REC}}{\dfrac{a_1\alpha_1 = a_1\beta_1}{\alpha_1 = \beta_1}\ \textsf{PREFIX} \quad \cdots \quad \dfrac{a_n\alpha_n = a_n\beta_n}{\alpha_n = \beta_n}\ \textsf{PREFIX}}\ \textsf{SUM}$$

Figure 2: A basic step in the tableau system

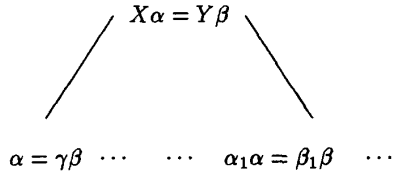

Figure 3: An eliminating subtableau for $X\alpha = Y\beta$.

of the length of the expressions involved according to Proposition 1: for instance, the above application of SUBL can be thought of as substituting $\gamma\beta$ for α, obtaining $\alpha_i\gamma\beta = \beta_i\beta$ and then using Proposition 1 to get $\alpha_i\gamma = \beta$.

Notice that for any application of SUB we have that

Proposition 2 *If* $m = \max\{|X| \mid X \in Var\}$ *then*

1. $|\alpha| < |X\alpha|$ *and* $|\gamma\beta| < |Y\beta|$
2. $length(\alpha_i) + length(\gamma) + length(\beta_i) \leq 3m + 1$ *for any application of* SUB.

The latter follows from the observation made earlier about a basic step. Notice that this is completely independent of $length(\alpha)$ and $length(\beta)$.

We say that the residual or the consequent of an application of a SUB rule is a *successful terminal* if it has one of the forms

1. $\alpha = \beta$ where there is another node above it (and an application of PREFIX in between) in either the same or another eliminating subtableau also labelled $\alpha = \beta$.
2. $\alpha = \alpha$

Clearly a node obeying 2 relates bisimilar processes. It turns out that this is also true of 1 in the context of a successful tableau.

When a consequent of SUB or the residual is not a successful terminal we build a new eliminating subtableau with it as root as described above, and continue in this fashion. Therefore, a tableau is defined as successions of eliminating subtableaux as in Figure 4. If at any point in the construction of an eliminating subtableau we reach an unsuccessful terminal then we halt and count the resulting tableau as unsuccessful. A successful tableau on the other hand is a finite-depth proof tree all of whose leaves are successful terminals.

Example 2 (continued) The tableau in Figure 5 is a successful tableau for $X = A$. □

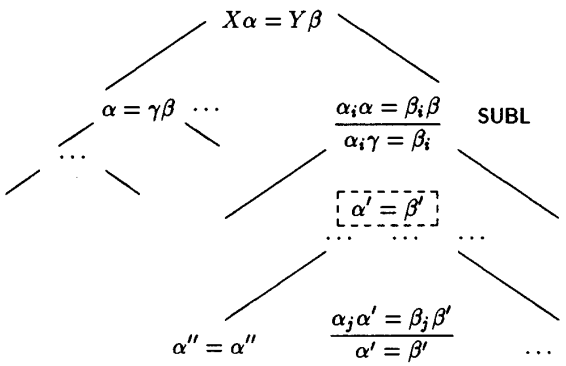

Figure 4: A tableau for $X\alpha = Y\beta$; some successful leaves are shown

3.2 Decidability, soundness, and completeness

We now sketch the proof of correctness of the tableau method.

Theorem 1

1. *Every tableau for $X\alpha = Y\beta$ is finite.*
2. *There are only finitely many tableaux for $X\alpha = Y\beta$.*

PROOF:(sketch) *Part 1:* If a tableau were infinite then it would have an infinite path. By definition such a path could not contain either successful or unsuccessful terminals. By Proposition 2(2) such a path can not pass through infinitely many nodes which are consequents of a SUB rule – for it would then contain a successful terminal infinitely often. Otherwise the path must almost always pass through a residual; but this is impossible as then it must pass infinitely often through unsuccessful terminals as the norm of a residual is strictly less than one directly above it by Proposition 2(1). *Part 2* follows from Part 1 and the observation that there is an upper bound on the number of basic steps along any path, namely $O(m^4k^{3m})$ where k is the number of variables and m the maximal norm of a variable. □

The next theorem states soundness and completeness of the tableau method. The proof of soundness depends on the approximation version of bisimulation equivalence for image-finite transition systems: that $p \sim q$ iff $\forall n \geq 0.\ p \sim_n q$ – see [7] for these definitions.

Theorem 2 *$X\alpha \sim Y\beta$ iff there exists a successful tableau for $X\alpha = Y\beta$.*

PROOF:(sketch) Suppose $X\alpha \sim Y\beta$. Then we build a tableau for $X\alpha = Y\beta$ which preserves the property that for each node $\alpha' = \beta'$ we have $\alpha' \sim \beta'$. Clearly, in the case of the PREFIX and REC rule if the antecedent is true then so is the consequent. Moreover, in the case of SUB if the residual is true and the antecedent is also true then so is the consequent. This just leaves the case of the SUM rule. However, by the definition of bisimulation if $(\sum_i a_i\alpha_i)\alpha \sim (\sum_j b_j\beta_j)\beta$ then for each i there is a j such that $a_i\alpha_i\alpha \sim b_j\beta_j\beta$ and for each j there is an i with the same feature. Now by Theorem 1 this tableau construction must terminate and without unsuccessful terminals.

Suppose we had a successful tableau for $X\alpha \sim Y\beta$ but $X\alpha \not\sim Y\beta$. As the transition systems for $X\alpha$ and $Y\beta$ are image-finite, there is a least n such that $X\alpha \not\sim_n Y\beta$ and $\forall m < n.\ X\alpha \sim_m Y\beta$. As the rules for basic steps are backwards sound with respect to each $\sim_n$

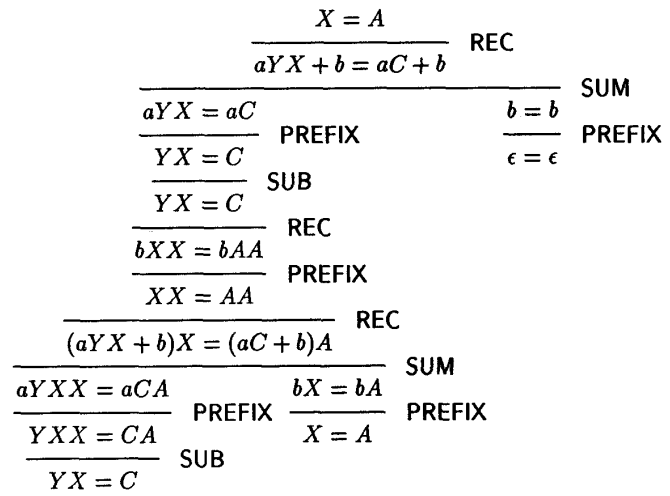

Figure 5: A successful tableau for $X = A$ in Example 2

we know that there is a leaf $\alpha' = \beta'$ of the eliminating subtableau for $X\alpha = Y\beta$ with the property $\alpha' \not\sim_m \beta'$ and $m < n$ (because PREFIX must have been applied at least once). Moreover, after the application of SUB there is at least one new root $\alpha'' = \beta''$ of the subsequent eliminating subtableau with the same property $\alpha'' \not\sim_m \beta''$. Choose such a root $X_1\alpha_1 = Y_1\beta_1$ which has least m such that $X_1\alpha_1 \not\sim_m Y_1\beta_1$. Continue this process down the tableau. We must therefore derive a contradiction, for we must eventually hit a successful terminal: it can not be of the form $\alpha = \alpha$ because $\alpha \sim \alpha$, and moreover it can not have the form $\alpha_i = \beta_i$ with $\alpha_i \not\sim_{m_i} \beta_i$ for we know that the node above it labelled $\alpha_i = \beta_i$ must have the property that $\alpha_i \sim_{m_i} \beta_i$. □

These theorems guarantee that bisimulation equivalence is decidable for normed recursive BPA processes. To test for the equivalence of two such processes p and q, one first expresses them in 3-GNF. Secondly, one tries to construct a successful tableau for them. By Theorem 1 we can do this exhaustively, and by Theorem 2 they are bisimilar iff we build a successful tableau at any point. Not only is the proof here much simpler than in [1, 2, 6] but also it appeals directly to an intuitive way of showing that two processes are bisimilar. Moreover, given a successful tableau for $X\alpha = Y\beta$, then the *finite* relation R defined by $\{(E\alpha', F\beta') \mid E\alpha' = F\beta' \text{ is a node}\}$ is the *essential* part of the bisimulation relation that underlies the equivalence $X\alpha \sim Y\beta$, in the sense that if for some pair $(E\alpha, F\beta) \in R$ we have $E\alpha \not\sim_n F\beta$ then there is another pair $(E'\alpha', F'\beta') \in R$ with the feature that $E\alpha \not\sim_m F\beta$ for some $m < n$.

An alternative understanding of R is that it is a *self-bisimulation* in the sense of [6]. This also gives us a different proof of the soundness half of Theorem 2.

A self-bisimulation is a bisimulation up to congruence w.r.t. sequential composition.

Definition 2 *For any binary relation R on Var^*, $\underset{R}{\rightarrow}$ is the least precongruence w.r.t. sequential composition that contains R, $\underset{R}{\longleftrightarrow}$ the symmetric congruence of $\underset{R}{\rightarrow}$ and $\underset{R}{\longleftrightarrow}^*$ the transitive closure of $\underset{R}{\longleftrightarrow}$ and thus the least congruence w.r.t. sequential composition containing R.*

Definition 3 *A relation $R \subseteq Var^* \times Var^*$ is called* a self-bisimulation *iff $\alpha R \beta$ implies that*

1. *$\alpha \xrightarrow{a} \alpha'$ implies $\beta \xrightarrow{a} \beta'$ for some β' s.t. $\alpha' \underset{R}{\longleftrightarrow}^* \beta'$*

2. $\beta \xrightarrow{a} \beta'$ *implies* $\alpha \xrightarrow{a} \alpha'$ *for some* β' *s.t.* $\alpha' \underset{R}{\longleftrightarrow}^* \beta'$

Lemma 1 [6] *If R is a self-bisimulation then* $\underset{R}{\longleftrightarrow}^* \subseteq \sim$.

Proposition 3
For any successful tableau $\mathbf{T}$ *for* $X\alpha = Y\beta$, $R_{\mathbf{T}} = \{(\alpha, \beta) \mid \alpha = \beta \text{ is a node in } \mathbf{T}, \alpha, \beta \in Var^*\}$ *is a self-bisimulation.*

3.3 Further work

The tableau method clearly prompts many outstanding issues. Does it hold for *all* recursive BPA processes including the unnormed ? Does it hold for weak bisimulation equivalence ? Does it hold when the process algebra contains more operators ? These are currently active lines of research. Recently, the first author [8] has shown that the decidability result can be extended to branching bismulation equivalence [14] for a class of normed BPA processes with silent actions. And Huynh and Tian [9] have proved the negative result that like language equivalence, failures and readiness equivalences are undecidable for this class of processes.

4 An equational theory

Besides yielding a straightforward decision procedure, the tableau technique can also be used to build a sound and complete equational theory for bisimulation equivalence of normed recursive BPA processes (in 3-GNF). For all that is required is a family of sound rules that permit one to derive the roots of successful tableaux.

We now formalize such an equational theory. Attention is restricted to processes in 3-GNF. Let Δ be a family of such processes $\{X_i \stackrel{\text{def}}{=} \sum_{j=1}^{n_i} a_{ij}\alpha_{ij} \mid 1 \leq i \leq k\}$. The proof system we present appeals to *assumptions* of the form $X\alpha = Y\beta$. The basic sequent of the system has the form $\Gamma \vdash_\Delta E = F$ where Γ is a set of assumptions, Δ is a family of processes, and E, F range over BPA expressions. Semantically, such a sequent is to be understood as follows: if the relation R given by $\{(X\alpha, Y\beta) \mid X\alpha = Y\beta \in \Gamma\} \cup \{(X_i, E_i) \mid X_i \stackrel{\text{def}}{=} E_i \in \Delta\}$ is part of a bisimulation then $E \sim F$. Thus, in the special case when Γ is empty then $\Gamma \vdash_\Delta E = F$ is intended to represent that $E \sim F$ (relative to the family Δ).

The rules are given in Table 2. Incorporated are the BPA axioms [4] as the rules R6-10. Equivalence and congruence rules are R1-5. So the only notable rules are those that deal with recursion, which have been dictated by the tableau method. First, R11 is an assumption *introduction* rule, justified by the interpretation of sequents described above. Secondly, R12 is an assumption *elimination* or discharge rule, which at the same time is a version of fixed point induction: notice that it is contextual in character, involving the BPA contexts $[\,]\alpha$ and $[\,]\beta$ where $[\,]$ is a 'hole'.

As expected, $\Gamma \vdash_\Delta E = F$ if there is a finite sequence of sequents $\sigma_1, \ldots, \sigma_n$ with $\sigma_n = \Gamma \vdash_\Delta E = F$ and where each σ_i is either an axiom instance (an instance of R1,R6-10, or R11) or is the result of one of the other rules applied to premises contained in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. The next result states weak soundness and completeness of the equational theory; however, the proof of soundness relies on showing strong soundness.

Theorem 3 $\emptyset \vdash_\Delta X\alpha = Y\beta$ *iff* $X\alpha \sim Y\beta$ *(with respect to Δ).*

PROOF:(sketch) For soundness, it suffices to interpret $\Gamma \vdash_\Delta E = F$ as described above: if the relation R given by $\{(X\alpha, Y\beta) \mid X\alpha = Y\beta \in \Gamma\} \cup \{(X_i, E_i) \mid X_i = E_i \in \Delta\}$ is part of a bisimulation then $E \sim F$. When Γ is empty this implies $E \sim F$. The axioms of Table 2 are true under this interpretation, and the rules of Table 2 preserve truth. For completeness, the proof proceeds on a successful tableau whose root is $X\alpha = Y\beta$. For each node of the tableau $E\alpha_1 = F\beta_1$ there is an equational proof $\Gamma \vdash_\Delta E\alpha_1 = F\beta_1$ when Γ is the set of pairs $\{(X_i\alpha_i, Y_i\beta_i)\}$ such that $X_i\alpha_i = Y_i\beta_i$ lables a node above $E\alpha_1 = F\beta_1$. □

This equational theory is somewhat non-standard in

the arena of process algebras. As it depends on assumptions, it is different in style both from Milner's elegant equational theory for regular processes with an explicit fixed point operator μ [11] and the version in [4] without μ. The crucial difficulty in extending Milner's theory to full normed and unnormed context-free processes with μ centres on the appropriate fixed-point induction rule. A case to ponder on is that $(\mu X.aX)E \sim (\mu X.aXXF)$ for *any* E and F.

References

[1] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *LNCS 259*, pages 93–114. Springer-Verlag, 1987.

[2] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. Technical Report CS-R8632, CWI, September 1987.

[3] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Inf. and Contr.*, 60:109–137, 1984.

[4] J.A. Bergstra and J.W. Klop. Process theory based on bisimulation semantics. In J.W. de Bakker, W.P de Roever, and G. Rozenberg, editors, *LNCS 354*, pages 50–122. Springer-Verlag, 1988.

[5] J. Bradfield and C. Stirling. Verifying temporal properties of processes. In *LNCS 458*, pages 115–125. Springer-Verlag, 1990.

[6] D. Caucal. Graphes canoniques de graphes algebriques. Rapport de Recherche 872, INRIA, Juillet 1988.

[7] M. Hennessy and A.J.R.G Milner. Algebraic laws for nondeterminism and concurrency. *JACM*, 32(1):137–161, January 1985.

[8] H. Hüttel. Silence is golden: Branching bisimilarity is decidable for context-free processes. Submitted for CAV91, 1991.

[9] Dung T. Huynh and Lu Tian. On deciding readiness and failure equivalences for processes. Technical Report UTDCS-31-90, University of Texas at Dallas, September 1990.

[10] A.J. Korenjak and J.E. Hopcroft. Simple deterministic languages. In *Proc. Sevent Annual IEEE Symposium on Switching and Automata Theory*, pages 36–46, 1966.

[11] R. Milner. A complete inference system for a class of regular behaviours. *JCSS*, 28:439–466, 1984.

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

[13] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. In *LNCS 351*, pages 369–383. Springer-Verlag, 1989.

[14] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, CWI/Vrije Universiteit, 1990.

Equivalence	
R1	$\Gamma \vdash_\Delta E = E$
R2	$\dfrac{\Gamma \vdash_\Delta E = F}{\Gamma \vdash_\Delta F = E}$
R3	$\dfrac{\Gamma \vdash_\Delta E = F \quad \Gamma \vdash_\Delta F = G}{\Gamma \vdash_\Delta E = G}$
Congruence	
R4	$\dfrac{\Gamma \vdash_\Delta E_1 = F_1 \quad \Gamma \vdash_\Delta E_2 = F_2}{\Gamma \vdash_\Delta E_1 + E_2 = F_1 + F_2}$
R5	$\dfrac{\Gamma \vdash_\Delta E_1 = F_1 \quad \Gamma \vdash_\Delta E_2 = F_2}{\Gamma \vdash_\Delta E_1 E_2 = F_1 F_2}$
BPA axioms	
R6	$\Gamma \vdash_\Delta E + F = F + E$
R7	$\Gamma \vdash_\Delta (E + F) + G = E + (F + G)$
R8	$\Gamma \vdash_\Delta E + E = E$
R9	$\Gamma \vdash_\Delta (E + F)G = EG + FG$
R10	$\Gamma \vdash_\Delta E(FG) = (EF)G$
Recursion	
R11	$\Gamma, X\alpha = Y\beta \vdash_\Delta X\alpha = Y\beta$
R12	$\dfrac{\Gamma, X\alpha = Y\beta \vdash_\Delta E\alpha = F\beta}{\Gamma \vdash_\Delta X\alpha = Y\beta}$ $\quad X \stackrel{\text{def}}{=} E,\ Y \stackrel{\text{def}}{=} F \in \Delta$

Table 2: Rules of inference in the equational theory