On the Decidability of Non-Interleaving Process Equivalences[1]

Astrid Kiehn
Institut für Informatik, TU München
Arcisstr.21, D–80290 München
email: kiehn@informatik.tu-muenchen.de

Matthew Hennessy
Cognitive and Computing Sciences
University of Sussex
Falmer, Brighton BN1 9QH
email: matthewh@cogs.sussex.ac.uk

Abstract

We develop decision procedures based on proof tableaux for a number of non-interleaving equivalences over processes. The processes considered are those which can be described in a simple extension of *BPP*, Basic Parallel Processes, obtained by omitting the restriction operator from *CCS* . Decision procedures are given for both *strong* and *weak* versions of *causal bisimulation, location equivalence* and *ST-bisimulation.*

1 Introduction

This paper is concerned with the development of automatic verification techniques for process description languages. Typically if P and Q are process descriptions we wish to develop decision procedures for checking if P and Q are semantically equivalent. If P and Q are expressions from *process algebras* or given in terms of *labelled transition systems* then there are already a number of software systems which can automatically check for such semantic identities, [CPS89, SV89]. The main semantic equivalences handled by these tools are variations on *bisimulation equivalence*, [Mil89], and there is also the major restriction that the processes to be checked must be finite state.

More recently techniques have been developed for handling certain kinds of infinite state processes. For example in [CHS92] it was shown that *strong bisimulation* is decidable for context-free processes while [CHM9] contains a similar result for so-called Basic

[1]This work has been supported by the ARC project *An Algebraic Theory for Distributed Systems*

Each of these three equivalences were originally defined using very different meta-languages for expressing and emphasising different intentional features of the behaviour of processes. In [Kie94] it is shown how at least the first two can be expressed in a uniform framework and here we show that this framework can also be used to express *ST-bisimulation.* This is central to our work. We develop one decision procedure based on a tableau method for *local cause bisimulation* which from [Kie94] is known to be equivalent to *location equivalence*, [BCHK9]. We then show how very simple modifications lead to a decision procedure for *global cause bisimulation*, which is known to be equivalent to finite subsets of $\mathcal{C}$. Let $BPP_l(X)$ denote the set of extended processes. As usual the variable x in $rec\,x.\ t$ acts as a *binder* which leads in the standard manner to *free* and *bound* occurrences of variables and a closed process is one with no free occurrence of any variable. We also assume that all occurrences of x in $rec\,x.\ t$ are *guarded*, (see [Mil89] for a formal definition). We use $p, q, \ldots$ to range over CCS, the set of closed processes of $CCS(X)$ and $P, Q \ldots$ to range over BPP_l, the set of closed processes in $BPP_l(X)$. For $T \in BPP_l(X)$ let $cau(T)$ be the set of causes, i.e. elements of $\mathcal{C}$, occurring in T and $cs(T)$, the set of *cause sets* of T, i.e. the set of subsets of $\mathcal{C}$ occurring in T. Obviously $cau(T) = \{\, l \in \Gamma \mid \Gamma \in cs(T)\,\}$. Within $BPP_l(X)$ we represent a CCS processes p as $\emptyset \rhd p$.

Throughout the paper we will use a structural congruence, $\equiv$, over extended processes. This is defined to be the least syntactic congruence generated by the equations

$$\begin{aligned} X \mid Y &= Y \mid X, \\ X \mid (Y \mid Z) &= (X \mid Y) \mid Z, \\ \Gamma \rhd (X \mid Y) &= \Gamma \rhd X \mid \Gamma \rhd Y, \\ \Gamma \rhd (X + Y) &= \Gamma \rhd X + \Gamma \rhd Y. \end{aligned}$$

In other words we work modulo the commutativity and associativity of $\mid$ and we assume that $\Gamma \rhd$ distributes over the two operators $\mid$ and $+$.

In this section we give a transition system which formalises the "local causality" between the actions of processes. The natural bisimulation equivalence defined using this transition system is the same as location equivalence, at least for the processes in CCS, [Kie94]. The transition system is given in Figures 1 and 2, and it defines two relations over processes from BPP_l. $P \xrightarrow[\Gamma,l]{a}{}^{(lc)} Q$ means that process P can perform the visible action a, the causes of this action being all the causes in Γ and all future actions which have this occurrence of a as a cause will have l in their set of causes. On the other hand $P \xrightarrow{\tau}{}^{(lc)} Q$ means that P can perform an internal computation and be transformed into Q. Notice also that each visible action, resulting from the application of the rule (LG1), introduces a new cause l. So as a computation proceeds the actions occurring in the computation are recorded as distinct causes in the cause sets of the process. The characteristic τ rule for local cause transitions is the rule for communication (L4). This means that the causes of communications are not accumulated in the cause sets. (L4) uses the somewhat non-standard notation $P[\emptyset/l]$ to denote the result of replacing each cause set Γ occurring in P by the cause set $\Gamma - \{l\}$. More generally we will use $P[\Delta/l]$ to denote the result of replacing each Γ containing l in P by $(\Gamma - \{l\}) \cup \Delta$.

For all $a \in Act$, $\Gamma \in \mathcal{CS}$, $l \in \mathcal{C}$ let $\xrightarrow[\Gamma,l]{a}{}^{(lc)} \subseteq (\ BPP_l \ \times \ BPP_l\ )$ be the least binary relations which satisfy the following axiom and rules.

(LG1) $\quad \Gamma \rhd a.p \xrightarrow[\Gamma,l]{a}{}^{(lc)} \Gamma \cup \{l\} \rhd p \qquad l \notin \Gamma$

(LG2) $\quad P \xrightarrow[\Gamma,l]{a}{}^{(lc)} P',\ l \notin cau(Q)$ implies $P + Q \xrightarrow[\Gamma,l]{a}{}^{(lc)} P'$
$\qquad Q + P \xrightarrow[\Gamma,l]{a}{}^{(lc)} P'$

(LG) $\quad P \xrightarrow[\Gamma,l]{a}{}^{(lc)} P',\ l \notin cau(Q)$ implies $P \mid Q \xrightarrow[\Gamma,l]{a}{}^{(lc)} P' \mid Q$

(LG4) $\quad \Gamma \rhd p[rec\, x.\ p/x] \xrightarrow[\Gamma,l]{a}{}^{(lc)} P'$ implies $\Gamma \rhd rec\, x.\ p \xrightarrow[\Gamma,l]{a}{}^{(lc)} P'$

(LG5) $\quad P \equiv P',\ \ P \xrightarrow[\Gamma,l]{a}{}^{(lc)} Q,$ implies $P' \xrightarrow[\Gamma,l]{a}{}^{(lc)} Q$

Figure 1: Visible Local Cause Transitions

The use of the structural congruence in the rules (LG5) and (L6) enables us to have a relatively simple set of defining rules for the transition systems. For example there is no rule which immediately applies to terms of the form $\Gamma \rhd (a.p \mid Q)$ by virtue of its syntactic form. But (LG) can be applied to $\Gamma \rhd a.p \mid \Gamma \rhd Q$ and since $\Gamma \rhd (a.p \mid Q) \equiv \Gamma \rhd a.p \mid \Gamma \rhd Q$ the rule (LG5) can be used to infer the same transition for $\Gamma \rhd (a.p \mid Q)$.

For every extended process P, every visible action a, location l and cause set Γ let $Der_{\Gamma,l}(P, a)$ be defined as $\{Q \mid P \xrightarrow[\Gamma,l]{a}{}^{(lc)} Q\}$. Because we assume that processes are guarded these sets are always finite. For exactly the same reason the set of τ-derivatives, $Der(P, \tau) = \{Q \mid P \xrightarrow{\tau}{}^{(lc)} Q\}$, is finite. We also let $S(P)$ denote the set of actions from Act_τ which P can perform.

Definition 2.1 [Local Cause Equivalence]
A symmetric relation $R \subseteq \ BPP_l \ \times \ BPP_l$ is called a *local cause bisimulation* iff $R \subseteq G(R)$ where
$(P, Q) \in G(R)$ iff
(i) $\ P \xrightarrow{\tau}{}^{(lc)} P'$ implies $Q \xrightarrow{\tau}{}^{(lc)} Q'$ for some $Q' \in \ BPP_l$ with $(P', Q') \in R$
(ii) $\ P \xrightarrow[A,l]{a}{}^{(lc)} P', l = new(cau(P) \cup cau(Q)),$
implies $Q \xrightarrow[A,l]{a}{}^{(lc)} Q'$ for some $Q' \in \ BPP_l$ with $(P', Q') \in R$.

Let $\xrightarrow{\tau}^{(lc)} \subseteq (BPP_l \times BPP_l$

But since p is closed this reduces to

$$\{rec\, y.(w[p/x])\} \cup Gen(p) \cup Gen(w)[p/x][rec\, y.w[p/x]/y].$$

For the right hand side we have

$$Gen(p) \cup Gen(rec\, y.w)[p/x] = Gen(p) \cup (\{rec\, y.w\} \cup (Gen(w)[rec\, y.w/y]))[p/x]$$

Therefore it remains to show $Gen(w)[p/x][rec\, y.w[p/x]/y] \subseteq Gen(w)[rec\, y.w/y][p/x]$. However these two sets are equal because p is closed. □

For an extended process P we use $Gen(P)$ to denote $Gen(pure(P))$ where $pure(P)$ is the *CCS* process obtained by erasing all cause sets from P. The next lemma shows that the generators of extended processes are closed under transitions.

Lemma 2.4 *If $P \xrightarrow{\tau}^{(lc)} Q$ or $P \xrightarrow[\Gamma,l]{a}^{(lc)} Q$ then $Gen(Q) \subseteq Gen(P)$.*

Proof By induction on the derivation of transitions. The only difficult case is when the transition is inferred using the rules (LG4) or (L5). An immediate corollary of the previous lemma is that $Gen(t[rec\, x.\ t/x]) \subseteq Gen(rec\, x.\ t)$ and both these cases will then follow by an application of induction. □

It is very easy to see that the set of generators of any extended process P is finite and therefore the previous lemma gives us a representation theorem for the "state space" of processes reachable from P. If we ignore cause sets then every such state is equivalent to a parallel product of the generators of P. This is summarised in the next proposition.

Proposition 2.5 *Let $p \in$ CCS.*

1. *The set $Gen(p)$ is finite.*
2. *There is a set $K = \{i_1, \ldots, i_n\}$ with $p_{i_j} \in Gen(p)$ for all $j \in \{1, \ldots, n\}$ such that $p \equiv \prod_{k \in K} p_k$.*
3. *If G is a set of generators and $P \in \mathrm{BPP}_l$ then the following holds: $Gen(P) \subseteq G$ and $P \xrightarrow[\Gamma,l]{a}^{(lc)} Q$ or $P \xrightarrow{\tau}^{(lc)} Q$ imply $Gen(Q) \subseteq G$.*

Proof The first two statements are easily shown by induction on the structure of p. The third is a consequence of the previous lemma. □

This result also shows that the class of processes we consider may be represented by Petri nets. To construct a net equivalent to P we use as places the elements of $Gen(P)$. The transitions between places are defined using the operational semantics of *CCS*. The initial marking is determined by parallel product of elements of $Gen(P)$ yielding P. Note, that this Petri net representation is not truly concurrent in the sense of [Gol88] and [Tau89]. The term $p \equiv (a.nil \mid b.nil) + c.nil$ has as generator set $Gen(p) = \{p, a.nil, b.nil, c.nil, nil\}$. So the net for p contains initially one token on the place p and none on the others. A truly concurrent Petri net for p would initially have at least two tokens to enable the transitions for a and b concurrently. We use the Petri net representation in a Section 5 to show that divergence of processes is decidable. As this problem is independent of concurrent or dependent occurrences of transitions the representation is sufficient for this purpose.

3 Ordering Processes

In this section we develop two distinct orderings on extended processes which will be used in the decision procedure. Both are based on orderings on vectors of natural numbers.

Let $\mathbb{N}^k$ represent the set of vectors-70.80Tfd[199etri

Now $\mathcal{C}$ is ordered as $l_1 < l_2 < l_3 < \ldots$ and this extends naturally to cause sets, finite subsets of $\mathcal{C}$, again lexicographically. Let

> $\{l_{n_1}, \ldots l_{n_k}\} < \{l_{m_1}, \ldots l_{m_{k'}}\}$, where $i < j$ implies $n_i < n_j$ and $m_i < m_j$, if there exists some $j \in \{1, \ldots, k'\}$ such that
>
> 1. $n_i = m_i$ for every $i < j$
> 2. if $j \leq k$ then $n_j < m_j$.

This is a total well-founded ordering on cause sets.

Definition 3.1 If G is a finite set of generators then a *G-parform* is any extended term of the form $\prod_{j \in J} \Gamma_j \rhd p_j$, where each p_j is a polynomial over G, which satisfies $i < j$ implies $\Gamma_i < \Gamma_j$. □

Lemma 3.2 *For every extended process P such that $Gen(P) \subseteq G$ there is a G-parform Q such that $P \equiv Q$.*

Proof By structural induction on P. □

Since the set of generators G is fixed for the remainder of the paper we will refer to G-parforms as simply parforms and we use $pf(P)$ to denote the parform to which P can be reduced. This is a slight abuse of notation as P may be reduced to two parforms which are not syntactically identical. But they will be equivalent up to the associativity and commutativity of $|$ and therefore they are "essentially" the same.

We now extend the ordering $\sqsubset_{lex}$ to parforms. For any parform $P := \prod_{i \in I} \Gamma_i \rhd p_i$ and any cause set Γ the vector $\Gamma(P)$ is defined as follows:

$$\Gamma(P) = \begin{cases} \alpha(p_i), & \Gamma = \Gamma_i \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

(Here $\mathbf{0}$ is the vector which consists only of 0s.) Note that this is well-defined for parforms since every cause set appears at most once in these terms. This notation is used in the following definition.

Definition 3.3i

, wThDefinitiolaj1 26.88(92)202po0-defined1m69. 5

Proposition 3.5 *Let P, Q, R be* parforms *with $P \sqsubset_{lex} Q$. Then*

1. *$pf(P \mid R) \sqsubset_{lex} pf(Q \mid R)$,*
2. *if π is a cause set renaming preserving the natural order on $cs(P) \cup cs(Q)$ then $\pi(P) \sqsubset_{lex} \pi(Q)$.* $\square$

This is the first ordering required in the decision procedure. The second also comes from an ordering on $\mathbb{N}^k$:

> for $\alpha, \beta \in \mathbb{N}^k$ let $\alpha \leq \beta$ if $\alpha_i \leq \beta_i$ for every $1 \leq i \leq k$.

However we first use this ordering to induce an ordering on *words* over $\mathbb{N}^k$:

> For each $v, w \in (\mathbb{N}^k)^*$ let $v \preceq w$ whenever there is an injection $f : \{1, \ldots, |v|\} \rightarrow \{1, \ldots, |w|\}$ such that $i_1 < i_2$ implies $f(i_1) < f(i_2)$ and $v[i] \leq w[f(i)]$

where $w[i]$ denotes the i^{th} letter of w. The main property of this ordering is given by

Theorem 3.6 *Let $(u_i)_{i \in \mathbb{N}}$, $u_i \in (\mathbb{N}^k)^\star$ be an infinite sequence of words over $\mathbb{N}^k$. Then there exists some $i, j \in \mathbb{N}$ such that $u_i \preceq u_j$.*

Proof This is a variation on Higman's Theorem, [Lot8]. Assume that there exist infinite sequences $(u_i)_{i \in \mathbb{N}}$, $u_i \in (\mathbb{N}^k)^\star$, such that $u_j \not\preceq u_i$ whenever $j < i$. Using the axiom of choice we can select an "earliest" of such sequences $(x_i)_{i \in \mathbb{N}}$ that is x_1 is the shortest word beginning such a sequence, x_2 is the shortest word such that x_1, x_2 is beginning such a sequence and so on. The sequence $(x_i)_{i \in \mathbb{N}}$ contains an infinite subsequence $(x_{i_j})_{j \in \mathbb{N}}$ with $x_{i_k}[1] \leq x_{i_l}[1]$ whenever $k < l$. Let $w[2 \ldots]$ denote the word w with the firstQ

…]Tj [illegible]

$(u_i)_{i \in \mathbb{N}}$,

For example

$$\{1\} \rhd (a.p) \mid \{1,2\} \rhd (b.nil \mid a.p)$$
$$\preceq$$
$$\{1\} \rhd (a.p \mid a.p) \mid \{1,2\} \rhd (c.nil) \mid \{2\} \rhd (a.p \mid b.nil \mid c.nil).$$

because $\alpha(a.p) \leq \alpha(a.p \mid a.p)$, $\alpha(b.nil \mid a.p) \leq \alpha(a.p \mid b.nil \mid c.nil)$.

There is an alternative characterisation of this ordering:

Proposition 3.8 *$P \preceq Q$ iff $Q \equiv Q_1 \mid Q_2$ and there is a cause set bijection $\pi\colon cs(P) \longrightarrow cs(Q_1)$ preserving the natural order on $cs(P)$ such that $Q_1 \equiv \pi(P)$.*

Proof Let $P' := \prod_{i \in I} \Gamma_i \rhd p_i$ and $Q' := \prod_{j \in J} \Delta_j \rhd q_j$ be G-parforms of P and Q respectively.

First suppose that $P \preceq Q$, i.e. $\omega(P') \preceq \omega(Q')$. Then there is an injection $f : I \longrightarrow J$ such that $i_1 < i_2$ implies $f(i_1) < f(i_2)$ and $\alpha(p_i) \leq \alpha(q_{f(i)})$ This means for each $i \in I$ that $q_{f(i)} \equiv p_i \mid r_i$ for some r_i.
Let Q_1, Q_2 denote $\prod_{i \in I} \Delta_{f(i)} \rhd p_i$ and $\prod_{i \in I} \Delta_{f(i)} \rhd r_i \mid \prod_{j \in J \setminus f(I)} \Delta_j \rhd q_j$ respectively. Then $Q \equiv Q_1 \mid Q_2$, and the required bijection $\pi : \{\Gamma_{i_1}, \ldots, \Gamma_{i_{|I|}}\} \longrightarrow \{\Delta_{f(i_1)}, \ldots, \Delta_{f(i_{|I|})}\}$ is given by $\pi(\Gamma_i) := \Delta_{f(i)}$.

The converse is similar. □

4 The Decidability Algorithm

In order to decide for two processes $P, Q \in BPP_l$,where $cs(P) = cs(Q)$, whether they are local cause equivalent we build up a tableau $T(P = Q)$. A tableau $T(P = Q)$ is a proof tree whose root is labelled $P = Q$ and whose proper nodes (there are also intermediate nodes, see below) are labelled with expressions of the form $P' = Q'$ where P', Q' are extended processes whose generators are in $Gen(P) \cup Gen(Q)$ and which satisfy $cs(P') = cs(Q')$. Each rule for extending a tableau has the form

$$\frac{P = Q}{P_1 = Q_1, \ldots, P_n = Q_n}$$

with possible side conditions. Intuitively the premise of such a rule can be viewed as a goal to achieve while the consequents represent sufficient subgoals to be established. We distinguish between proper and intermediate nodes. All proper nodes in the proof tree are either terminal or non-terminal and a proof tree can be extended by applying one of the rules to a non-terminal node, thereby introducing n new nodes. It may be that the application of a rule will violate the condition that labels must be of the form $R = S$ with $cs(R) = cs(S)$. In such cases we can simply add a $\Gamma \rhd \mathit{nil}$ factor to R for each $\Gamma \in cs(S) \setminus cs(R)$ and similarly for S. Extended processes are considered up to $\equiv$.

A node is terminal if it has one of the following forms:

- $P = Q$ where $P \equiv Q$; in which case the node

(UNWIND)

$$\frac{P \quad = \quad Q}{\{Der_{\Gamma,l}(P,a) = Der_{\Gamma,l}(Q,a)\} \qquad Der(P,\tau) = Der(Q,\tau)}$$

where $l = new(cau(P)) = new(cau(Q))$
$\Gamma \in cs(P) = cs(Q)$ and $a \in act(P) \cup act(Q)$

(SUM)

$$\frac{\{P_1,\ldots,P_n\} \quad = \quad \{Q_1,\ldots,Q_m\}}{\{\,P_i = Q_{f(i)}\}_{i\in\{1,\ldots,n\}} \qquad \{P_{g(j)} = Q_j\,\}_{j\in\{1,\ldots,m\}}}$$

where f and g are mappings $f : \{1,\ldots n\} \longrightarrow \{1,\ldots m\}$
and $g : \{1,\ldots,m\} \longrightarrow \{1,\ldots,n\}$

(SUBL)

$$\frac{\pi(P_1) \mid P_2 \quad = \quad Q}{\pi(P_1') \mid P_2 \quad = \quad Q}$$

where π is a cause set renaming which is
order preserving and bijective on $cs(P_1) = cs(P_1')$ and there is a dominated
node labelled $P_1 = P_1'$ or $P_1' = P_1$ with $pf(P_1') \sqsubset_{lex} pf(P_1)$

(SUBR)

$$\frac{Q \quad = \quad \pi(P_1) \mid P_2}{Q \quad = \quad \pi(P_1') \mid P_2}$$

where π is a cause set renaming which is
order preserving and bijective on $cs(P_1) = cs(P_1')$ and there is a dominated
node labelled $P_1 = P_1'$ or $P_1' = P_1$ with $pf(P_1') \sqsubset_{lex} pf(P_1)$

Figure : The Tableau Rules

2. it must dominate a node **m** labelled by $P_1 = P_1'$ or $P_1' = P_1$,

. in the label on the dominated node it must be the case that $pf(P_1') \sqsubset_{lex} pf(P_1)$.

The result of applying the rule is the generation of a new node labelled by $\pi(P_1') \mid P_2 = Q$. Note that as a result of the application of the rule the lexicographical order of the process is decreased. This order, $\sqsubset_{lex}$, is where **A** is $a.P \mid \{1\} \rhd nil = a.P \mid \{1\} \rhd nil$ and empty cause sets have been omitted. Since all terminal nodes are successful the tableau is successful, hence $P \sim_{lc} Q$.

Theorem 4.2 *Let* $P, Q \in \mathrm{BPP}_l$ *. Every tableau for* $P = Q$ *is finite.*

Proof Let $X = Gen(P) \cup Gen(Q)$. If the tableau is not finite then —as it is finitely branching— it must contain an infinite path. By Proposition .5 every application of a **SUB** rule preserves the order $\sqsubset_{lex}$ and since this order is well-founded this infinite path can not eventually only consist of applications of **SUB**. So there are infinitely many nodes, $(\mathbf{n_i})_{i \in \mathbb{N}}$ along a path to which rule **UNWIND** is applied. Let $(U_i = V_i)_{i \in \mathbb{N}}$ denote the sequence of labels on these nodes.

We now consider the words over $\mathbb{N}^{|X|}$ generated by each pair U_i, V_i, $\omega(U_i), \omega(V_i)$ respectively. In fact it will be convenient to use words over the slightly larger set $\mathbb{N}^{|X|+1}$ and identify any vector $\alpha \in \mathbb{N}^{|X|}$ as the vector in $\mathbb{N}^{|X|+1}$ obtained by setting the last component to 0. If we then encode the equality symbol = as a vector $\beta \in \mathbb{N}^{|X|+1}$ with $\beta(i) = 0$ for each generator position (i.e. for $i \leq |X|$) and $\beta(|X| + 1) = 1$ for the new component then $U_i = V_i$ can be represented as the word $\omega(U_i)\beta\omega(V_i)$ over the alphabet $\mathbb{N}^{|X|+1}$. For convenience let ω_i denote this word which labels the node $\mathbf{n_i}$.

Now consider two nodes $\mathbf{n_j}, \mathbf{n_k}$, where $j < k$, which are labelled by the equations $U_j = V_j$ and $U_k = V_k$ respectively. We show, by contradiction, that $\omega_j \not\preceq \omega_k$. Suppose this is not the case, i.e. $\omega_j \preceq \omega_k$ and therefore both $U_j \preceq U_k$ and $V_j \preceq V_k$. Since $\sqsubset_{lex}$ is a total order we can assume without loss of generality that $pf(U_j) \sqsubset_{lex} pf(V_j)$. By Proposition .8 we know that $V_k \equiv \pi(V_j) \mid$

Proof If $P \sim_{lc} Q$ then we can build up a tableau such that $R \sim_{lc} S$ for each proper node labelled $R = S$. This is possible because the **UNWIND** rule directly reflects the operational semantics and when applying the **SUM** rules one can choose the functions f, g in the appropriate way. Lemma 2.2 guarantees that the nodes introduced via an application of the **SUB** rules also have this property. Hence the resulting tableau can not contain unsuccessful nodes. Finally by Theorem 4.2 the tableau is finite. $\square$

Proposition 4.4 (Soundness) *If a tableau $T(P = Q)$ is successful then $P \sim_{lc} Q$.*

Proof

Let $\mathcal{R}^*$ denote the least set which contains $\equiv$ and

$$\mathcal{R} = \{(P, Q) \mid P = Q \text{ is the label of a proper node in the tableau}\}$$

and which is closed under the following operations:

1. $(P, Q) \in \mathcal{R}^*$ implies $(P \mid R, Q \mid R) \in \mathcal{R}^*$ for every R,
2. $(P, Q) \in \mathcal{R}^*$ implies $(\pi(P), \pi(Q)) \in \mathcal{R}^*$ for every cause renaming π which is bijective on $cs(P)(= cs(Q))$,

. $(P, Q) \in \mathcal{R}^*$, $(Q, R) \in \mathcal{R}^*$ implies $(P, R) \in \mathcal{R}^*$,

4. $(P, Q) \in \mathcal{R}^*$ implies $(P \mid \Gamma \rhd nil, Q) \in \mathcal{R}^*$ and $(P, Q \mid \Gamma \rhd nil) \in \mathcal{R}^*$ for any $\Gamma \in \mathcal{CS}$.

Then $\mathcal{R}^*$ is a local cause bisimulation. To prove this it is sufficient to check that for each node label $(P, Q) \in \mathcal{R}$ the moves from P can be matched by corresponding moves from its partner, that is,

- $P \xrightarrow{\tau}^{(lc)} P'$ implies $Q \xrightarrow{\tau}^{(lc)} Q'$ for some $Q' \in BPP_l$ with $(P', Q') \in \mathcal{R}^*$
- $P \xrightarrow[A,l]{a}^{(lc)} P', l = new(cau(P) \cup cau(Q))$, implies $Q \xrightarrow[A,l]{a}^{(lc)} Q'$ for some $Q' \in BPP_l$ with $(P', Q') \in \mathcal{R}^*$.

This can be established by induction on the length of the sequence of uninterrupted applications of **SUB** rules to a node.

case $i = 0$: In this case the node is either terminal —so we have $P \equiv Q$ and therefore $(P, Q) \in \mathcal{R}^*$— or rule **UNWIND** is applied to it in the next step. An application of **UNWIND** is always followed by an application of rule **SUM**. The labels of the nodes obtained from this application give the required matching moves (up to adding $\Gamma \rhd nil$ components).

case $i \rightarrow i + 1$: Without loss of generality the label under consideration is of the form (

Let $\xrightarrow{\mu}^{(st)} \subseteq (ST \times ST)$ be the least binary relation defined by the following axiom and rules.

$$\begin{array}{llll}
\text{(L1)} & \Gamma \rhd \mu.p \xrightarrow{\mu}^{(st)} \Gamma \rhd p & & \\[2ex]
\text{(L2)} & P \xrightarrow{\mu}^{(st)} P' & \text{implies} & P + Q \xrightarrow{\mu}^{(st)} P' \\
 & & & Q + P \xrightarrow{\mu}^{(st)} P' \\[2ex]
\text{(L)} & P \xrightarrow{\mu}^{(st)} P' & \text{implies} & P \mid Q \xrightarrow{\mu}^{(st)} P' \mid Q \\
 & & & Q \mid P \xrightarrow{\mu}^{(st)} Q \mid P' \\[2ex]
\text{(L4)} & P \xrightarrow{a}^{(st)} P',\ Q \xrightarrow{\bar{a}}^{(st)} Q' & \text{implies} & P \mid Q \xrightarrow{\tau}^{(st)} P' \mid Q' \\[2ex]
\text{(L5)} & \Gamma \rhd p[rec\,x.\ p/x] \xrightarrow{\mu}^{(st)} P' & \text{implies} & \Gamma \rhd rec\,x.\ p \xrightarrow{\mu}^{(st)} P' \\[2ex]
\text{(L6)} & P \equiv P',\ \ P \xrightarrow{\mu}^{(st)} Q, & \text{implies} & P' \xrightarrow{\mu}^{(st)} Q
\end{array}$$

Figure 4: Invisible ST Transitions

In the ST operational semantics of [AH9] communication consists of the simultaneous occurrence of *complete* actions and therefore we can not define the required τ transitions using the rules in Figure 2; instead we need to define it separately, along the standard lines as in [Mil89]. As usual in order to define the invisible transition,

τ

These are obtained by abstracting from internal moves. We only outline the development for *local cause equivalence* but it can be easily adapted for the other two. The weak local cause transitions are defined as follows:

For $a \in Act$ let $\overset{a}{\underset{\Gamma,l}{\Longrightarrow}}^{(lc)}$ be the least relation which satisfies

- $P \overset{a}{\underset{\Gamma,l}{\longrightarrow}}^{(lc)} Q$ implies $P \overset{a}{\underset{\Gamma,l}{\Longrightarrow}}^{(lc)} Q$
- $P \overset{a}{\underset{\Gamma,l}{\Longrightarrow}}^{(lc)} Q'$ and $Q' \xrightarrow{\tau}^{(lc)} Q$ implies $P \overset{a}{\underset{\Gamma,l}{\Longrightarrow}}^{(lc)} Q$
- $P \xrightarrow{\tau}^{(lc)} P'$ and $P' \overset{a}{\underset{\Gamma,l}{\Longrightarrow}}^{(lc)} Q$ implies $P \overset{a}{\underset{\Gamma,l}{\Longrightarrow}}^{(lc)} Q$

We also use

never evolve to a process which can diverge internally. Our decision procedure for the weak equivalences will only apply to these terms. However at least this is a decidable class:

Theorem 5.4 *The predicate* h-convergent *is decidable over* BPP_l .

Proof

It is not too difficult to see that P is *h-convergent* if and only if $pure(P)$ is *h-convergent* under the standard operational semantics where labels and causes are not mentioned. Therefore it is sufficient to consider CCS processes in *BPP*. Any such process p can be represented by the Petri net $PN(p)$ constructed as follows:

- take $Gen(p)$ as the set of places
- for each transition $g \xrightarrow{\mu} q$ introduce a Petri net transition labelled μ with g as the only input place and the generators $g_1, \ldots, g_n$ as output places, where q is equivalent to a polynomial over $\{g$

6 0 in Lecture Notes in Computer Science, pages 1 8–147. Springer–Verlag, 1992.

[CPS89]

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[SV89] R. De Simone and D. Vergamimi. Aboard auto. Report RT111, INRIA, 1989.

[Tau89] D.A. Taubner. *Finite Representations of CCS and TCSP Programs by Automata and Petri nets.* Number 69 in Lecture Notes in Computer Science. Springer–Verlag, 1989.

[vGV87] R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Prooceedings PARLE conference*, number 259 in Lecture Notes in Computer Science, pages 224–242. Springer–Verlag, 1987.

[VJ85] R. Valk and M. Jantzen. The residue of vector sets wit applications to decidability problems in Petri nets. *Acta Informatica*, (21):64 –674, 1985.

[Vog91a] W. Vogler. Deciding history preserving bisimilarity. In *Proceedings of ICALP 91*, number 510 in Lecture Notes in Computer Science, pages 495–505. Springer–Verlag, 1991.

[Vog91b] W. Vogler. Generalized OM-bisimulation. Report TUM-I911 , Technische Universität München, 1991.

[Vog92] W. Vogler. *Modular Construction and Partial Order Semantics of Petri nets.* Number 625 in Lecture Notes in Computer Science. Springer–Verlag, 1992.

7 Appendix

We here show that ST-equivalence which has been introduced in [vGV87] for Petri nets and in [AH9] for process algebras indeed coincides with the formalization we have given in Section 5. The proof is only for the set of processes BPP considered in this paper but it can be extended to whole of CCS in a straightforward way. We first outline the definition of ST-equivalence as given in [AH9] and then compare it to our formalisation given in Section 5.

ST processes are generated by the abstract syntax

$$P := t \quad | \quad F(a_l).t \quad | \quad P \mid P$$

where $t \in CCS$, $a \in Act$ and each index l occurs at most once in a term. Let ST denote the set of these processes. It is often called the set of *configurations* or *states* as it contains processes which have started some of their actions without having finished them. Each $F(a_l)$ stands for such an unfinished action. The index l serves to uniquely identify particular executions of an action. For our comparison we simply choose l from the set of causes $\mathcal{C}$.

The transition systems for visible and invisible moves are given in Figure 7. Axiom (S1) is only needed to be able to derive invisible action due to a communication. Transitions of this kind are not directly considered when comparing processes. It is important to note that these transition relations are only defined as relations over ST processes where each index l occurs at most once. So, for example, the rule (S5) for parallel can only be applied to an ST process whenever we are sure that the resulting term is also an ST process.

The use of unique indices gives a slightly different formulation than that in [AH9] where it was only necessary to ensure that for each action a and each index l the prefix $F(a_l)$ occurs at most once. However our formulation enables us to give a much simpler definition of

For each $\mu \in Act_\tau$ and $\lambda \in \{S(a_l), F(a_l) \mid a \in Act, l \in \mathcal{C}\}$ let $\xrightarrow{\mu}, \xrightarrow{\lambda} \subseteq (ST \times ST)$ be the least binary relations satisfying the following axioms and rules.

(S1) $\qquad \mu.p \xrightarrow{\mu} p$

(S2) $\qquad a.p \xrightarrow{S(a_l)} F(a_l).p \qquad l \in \mathcal{C}$

Lemma 7.2 *1. If $P \in ST$ then*

(a) $P \overset{S(a_l)}{\rightarrow} P'$ *implies* $lg(P) \xrightarrow[\emptyset,l]{a}{}^{(st)} lg(P')$,

(b) $P \overset{F(a_l)}{\rightarrow} P'$ *implies* $lg(P) \xrightarrow[\{l\},k]{a}{}^{(st)} lg(P')$ *for any* $k \in \mathcal{C} \setminus cau(P)$,

(c) $P \xrightarrow{\mu} P'$ *implies* $lg(P) \xrightarrow{\mu}{}^{(st)} lg(P')$.

2. If $P \in \mathrm{BPP}_{ST}$ *then*

(a) $P \xrightarrow[\emptyset,l]{a}{}^{(st)} P'$ *implies* $st(P) \overset{S(a_l)}{\rightarrow} st(P')$,

(b) $P \xrightarrow[\{l\},k]{a}{}^{(st)} P'$ *implies* $st(P) \overset{F(a_l)}{\rightarrow} st(P')$,

(c) $P \xrightarrow{\mu}{}^{(st)} P'$ *implies* $st(P) \xrightarrow{\mu} st(P')$.

Proof By induction on the length of the proof of a transition. □

The invariance of the two formalizations of ST-equivalence is now easily established. It is an immediate corollary of the final proposition since a *CCS* process p is represented in *BPP* by $\emptyset \rhd p$.

Proposition 7.3 *1. If* $P \sim P'$, $P, P' \in ST$, *then* $lg(P) \sim_{st} lg(P')$,

2. if $P \sim_{st} P'$, $P, P' \in \mathrm{BPP}_{ST}$, *then* $st(P) \sim st(P')$.

Proof Using the mappings st and lg a bisimulation of one set up can be translated into the other. The preceding lemma guarantees that the resulting relations are indeed bisimulations. □