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 [CHM93] contains a similar result for so-called Basic

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

Parallel Processes, *BPP* ; these correspond essentially to the processes generated by the standard syntax of *CCS* without the restriction operator. The decision procedures are not particularly efficient but at least they open up the possibility of the development of feasible proof techniques for classes of infinite state processes. The recent thesis, [Chr93], provides an excellent overview. The basic underlying idea is the use of proof tableaux to compare the behaviour of processes. Extending a tableau corresponds to "unwinding" the processes in order to compare their future possible behaviours. By restricting in various subtle ways the methods allowed for extending tableaux one can ensure that they are always finite. If in addition only a finite number of tableaux can be produced from a given pair of process descriptions then, provided the specific tableau method is sound for a particular equivalence, we have a decision procedure for that equivalence. This approach has been applied successfully to a number of infinite state languages but mostly for *strong bisimulation equivalence*. We wish to investigate the extent to which these techniques can also be used to develop decision procedures for *non-interleaving* semantic equivalences, [DD89, vGV87, DNM89].

We already know of one result of this kind. In [Chr92] a decision procedure, based on a tableaux method, is presented for *distributed bisimulation* $\sim_d$, [CH89], over the language *BPP* . However the soundness of the procedure depends crucially on a separation property which is part and parcel of the underlying transition system. The procedure could be adapted if we had a cancellation property

$$P \mid Q \sim P \mid R \text{ implies } Q \sim R$$

where $\mid$ represents the binary parallel composition of processes. But most (if not all other) standard non-interleaving equivalences such as *causal bisimulation*, [DD89], *ST-bisimulation*, [vGV87] or *location equivalence*, [BCHK93] do not have this property. So we show how to extend the more recent technique developed in [CHM93], to yield decision procedures for

- *location equivalence*
- *causal bisimulation*
- *ST-bisimulation*

over *BPP* . We also show how the method can be adapted to provide in addition decision procedures for weak versions of each of these equivalences over a large subclass of *BPP* , essentially those processes which can never evolve into states which are *divergent*. Moreover we show that this property is itself decidable for *BPP* processes.

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*, [BCHK93]. We then show how very simple modifications lead to a decision procedure for *global cause bisimulation*, which is known to be equivalent to *causal bisimulation* [DD89], and to our version of *ST-bisimulation*. Finally we show how further modifications can be made so as to deal with weak versions of these three equivalences.

We now briefly outline the remainder of the paper. In the next section we define the language, BPP_l , essentially the language of [CHM93] extended with sets of causes. We also define *weak local cause bisimulation* and state some of its properties which we require. In the next section we develop some orderings on processes which are used in the semantic tableaux which in turn are explained in Section 4. This section gives the decision procedure for *local cause bisimulation* over BPP_l . The last section shows how to adapt this procedure to *causal bisimulation* and *ST-bisimulation* and to the weak version of these equivalences for *h-convergent* processes.

2 Local Cause Bisimulation

As usual there is a countable set of atomic actions, $Act = \{a, b, c, \ldots, \bar{a}, \bar{b}, \bar{c}, \ldots\}$, ranged over by a, an invisible action τ not in Act and Act_τ, ranged over by μ, is used to denote $Act \cup \{\tau\}$. If X denotes a set of variables, ranged over by x, then the *standard* set of processes, $CCS(X)$, is obtained from the abstract syntax

$$t ::= nil \quad | \quad \mu.t \quad | \quad t + t \quad | \quad t \mid t \quad | \quad x \quad | \quad rec\, x.\, t$$

In order to describe the non-interleaving theories of processes we extend this syntax by introducing a countable set of *causes*, $\mathcal{C}$, which we assume have the form $\{l_1, l_2, \ldots\}$. These will be used in three different ways to describe the three different equivalences discussed in the paper. However for each of the equivalences we use exactly the same extended syntax:

$$T ::= \Gamma \rhd t \quad | \quad T \mid T$$

where $t \in CCS(X)$, and Γ is a *cause set*, i.e. a finite subset of $\mathcal{C}$. We use $\mathcal{CS}$ to denote the set of possible cause sets which can occur in a term, i.e. $\mathcal{CS}$ is the collection of

finite subsets of $\mathcal{C}$. Let $BPP_l(X)$ denote the set of extended processes. As usual the variable x in $recx.\ t$ 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 $recx.\ t$ 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 $\xrightarrow{a}{}^{(lc)}_{\Gamma,l} \subseteq (\ BPP_l \times BPP_l\ )$ be the least binary relations which satisfy the following axiom and rules.

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

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

(LG3) $\quad P \xrightarrow{a}{}^{(lc)}_{\Gamma,l} P',\ l \notin cau(Q) \quad$ implies $\quad P \mid Q \xrightarrow{a}{}^{(lc)}_{\Gamma,l} P' \mid Q$

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

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

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 (LG3) 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 $\{Q \mid P \xrightarrow{a}{}^{(lc)}_{\Gamma,l} Q\}$. 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) $P \xrightarrow{a}{}^{(lc)}_{A,l} P', l = new(cau(P) \cup cau(Q))$,
implies $Q \xrightarrow{a}{}^{(lc)}_{A,l} Q'$ for some $Q' \in BPP_l$ with $(P',Q') \in R$.

Let $\stackrel{\tau}{\longrightarrow}^{(lc)} \subseteq (\ BPP_l \times BPP_l\ )$ be the least binary relation defined by the following axiom and rules.

(L1) $\quad \Gamma \rhd \tau.p \stackrel{\tau}{\longrightarrow}^{(lc)} \Gamma \rhd p$

(L2) $\quad P \stackrel{\tau}{\longrightarrow}^{(lc)} P'$ implies $P + Q \stackrel{\tau}{\longrightarrow}^{(lc)} P'$, $Q + P \stackrel{\tau}{\longrightarrow}^{(lc)} P'$

(L3) $\quad P \stackrel{\tau}{\longrightarrow}^{(lc)} P'$ implies $P \mid Q \stackrel{\tau}{\longrightarrow}^{(lc)} P' \mid Q$, $Q \mid P \stackrel{\tau}{\longrightarrow}^{(lc)} Q \mid P'$

(L4) $\quad P \stackrel{a}{\underset{\Gamma,l}{\longrightarrow}}^{(lc)} P',\ Q \stackrel{\bar{a}}{\underset{\Delta,k}{\longrightarrow}}^{(lc)} Q'$ implies $P \mid Q \stackrel{\tau}{\longrightarrow}^{(lc)} P'[\emptyset/l] \mid Q'[\emptyset/k]$

(L5) $\quad \Gamma \rhd p[rec\, x.\ p/x] \stackrel{\tau}{\longrightarrow}^{(lc)} P'$ implies $\Gamma \rhd rec\, x.\ p \stackrel{\tau}{\longrightarrow}^{(lc)} P'$

(L6) $\quad P \equiv P',\ \ P \stackrel{\tau}{\longrightarrow}^{(lc)} Q,$ implies $P' \stackrel{\tau}{\longrightarrow}^{(lc)} Q$

Figure 2: Invisible Local Cause Transitions

Two processes P and Q are *local cause equivalent*, $P \sim_{lc} Q$, iff there is a local cause bisimulation R such that $(P, Q) \in R$. □

Note that this definition uses the function new which when applied to a finite set of causes Γ returns a cause l not in Γ. In order to obtain a finitely branching transition system for our decision algorithm we assume that $new(\Gamma)$ is the least cause in the ordering $l_0, l_1, \ldots$ not in Γ. This assumption does not affect the equivalence as it is closed under bijective cause renaming. We refer the reader to [Kie94] for a proof that $\sim_{lc}$ coincides with *location equivalence* over *CCS* as defined in [BCHK94].

Our decision procedure depends on certain properties of the transition semantics which we now discuss. The first states that the equivalence is largely independent of the location sets of a process; it is preserved by uniformed renaming of cause sets. A (uniform) cause set renaming is a any function $\pi{:}\mathcal{CS} \longrightarrow \mathcal{CS}$.

Lemma 2.2 *Let $P, Q, R \in$ BPP$_l$ and let π be a cause set renaming which is bijective over $cs(P) \cup cs(Q)$. Then*

1. *$P \sim_{lc} Q$ if and only if $\pi(P) \sim_{lc} \pi(Q)$,*
2. *$P \sim_{lc} Q$ implies $P \mid R \sim_{lc} Q \mid R$.*

□

In general processes in BPP_l are not finite-state in this transition system but the state space of any process has a finite basis, i.e. every element of a state space can be viewed as a polynomial over a finite set of generators. For any process $t \in CCS(X)$ we associate the set of generators, $Gen(t)$, defined as follows.

$$Gen(t) = \begin{cases} \{x\} & \text{if } t := x \\ \{nil\} & \text{if } t := nil \\ \{t\} \cup Gen(t_1) \cup Gen(t_2) & \text{if } t := t_1 + t_2 \\ \{t\} \cup Gen(t_1) & \text{if } t := \mu.t_1 \\ Gen(t_1) \cup Gen(t_2) & \text{if } t := t_1 \mid t_2 \\ \{t\} \cup Gen(t_1)[rec\, x.\ t_1/x] & \text{if } t := rec\, x.\ t_1 \end{cases}$$

In this definition the two operators + and | are treated unequally. The reason for this is that we will show how to represent any process as the "parallel product" of generators as opposed to a "choice product". We use $\prod_{i \in I} p_i$ to denote the "parallel product" $p_{i_1} \mid p_{i_2} \mid \ldots \mid p_{i_k}$ where I is the finite index set $\{i_1, \ldots, i_k\}$. Since we are working with respect to structural congrunce $\equiv$, which includes the associativity and commutativity of |, this notation is consistent. But first an important syntactic property of the function *Gen*:

Lemma 2.3 *For every* CCS *term u and closed* CCS *term p, $Gen(u[p/x]) \subseteq Gen(p) \cup Gen(u)[p/x]$.*

Proof By structural induction on u. We only give the case of $u = rec\, y.w$ as the others are straightforward. By α-conversion we may assume $x \neq y$.

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

By induction $Gen(w[p/x]) \subseteq Gen(p) \cup Gen(w)[p/x]$ so the left hand side is contained in

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

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* $P \stackrel{\tau}{\longrightarrow}^{(lc)} Q$ *or* $P \stackrel{a}{\underset{\Gamma,l}{\longrightarrow}}^{(lc)} Q$ *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* $P \stackrel{a}{\underset{\Gamma,l}{\longrightarrow}}^{(lc)} Q$ *or* $P \stackrel{\tau}{\longrightarrow}^{(lc)} Q$ *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 of length k of natural numbers. We use $\alpha, \beta, \ldots$ to range over these vectors with α_i denoting the i^{th}-component. The vectors can be lexicographically ordered in the standard way by

$\alpha \sqsubset_{lex} \beta$ if there is a j such that $\alpha_j < \beta_j$ and $\alpha_k = \beta_k$ for all $k < j$.

This is a total and well-founded ordering on $\mathbb{N}^k$ and it can be used to induce a similar ordering on the set of *CCS* processes obtained from a finite set of generators.

For the moment let G be a finite set of *CCS* generators and let us assume that they are totally ordered as $g_1 < g_2 < \ldots < g_k$. This may be any set but in the next section where we are trying to decide whether or not P and Q are equivalent we will choose as G the set $Gen(P) \cup Gen(Q)$; from Lemma 2.5 we know that any process derived from P or Q will also be a parallel product of elements from G, or polynomial over G. Therefore we are interested in terms which are polynomials over G. For such a polynomial p let $\alpha_i(p)$ be the number of occurrences of g_i in p and let $\alpha(p) \in \mathbb{N}^{|G|}$ be the vector $\langle \alpha_1(p), \alpha_2(p), \ldots \rangle$. This enables us to define a total well-founded ordering on polynomials over G by

$p \sqsubset_{lex} q$ if $\alpha(p) \sqsubset_{lex} \alpha(q)$.

This ordering can be in turn lifted to a subclass of extended terms, called *G-parforms* provided we first order cause sets.

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 3.3 Let P, Q be parforms. $P \sqsubset_{lex} Q$ if and only if there is a Γ such that $\Gamma(P) \sqsubset_{lex} \Gamma(Q)$ and for every $\Delta < \Gamma$, $\Delta(P) = \Delta(Q)$. □

Proposition 3.4 *The relation $\sqsubset_{lex}$ is a total well-founded ordering on the set of* G-parforms. □

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 $\pi(P) \sqsubset_{lex} \pi(Q)$.* □

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 $f : \{1, \ldots, |v|\} \to \{1, \ldots, |w|\}$ 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, [Lot83]. 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 first vector chopped off. Then

$$x_1, x_2, \ldots, x_{i_1 - 1}, x_{i_1}[2 \ldots], x_{i_2}[2 \ldots], x_{i_3}[2 \ldots], \ldots$$

is earlier than $(x_i)_{i \in \mathbb{N}}$ contradicting the choice of $(x_i)_{i \in \mathbb{N}}$. □

We now lift this ordering on words over $\mathbb{N}^k$ to *G-parforms* by associating in a systematic manner a word over $\mathbb{N}^k$ with each *G-parform*. If P is the G-parform $\prod_{i \in I} \Gamma_i \rhd p_i$ then let $\omega(P)$ denote the word $\alpha(p_1)\alpha(p_2) \ldots \alpha(p_{|I|})$.

Definition 3.7 For any pair of extended processes $P, Q \in BPP_l$ with G-parforms P', Q' let $P \preceq Q$ if $\omega(P') \preceq \omega(Q')$. □

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 $\pi: cs(P) \longrightarrow cs(Q_1)$ 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 $\Gamma \rhd nil$ 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 is *successful*
- $P = Q$ where $S(P) = S(Q) = \emptyset$; in which case the node is *successful*
- $P = Q$ where either

  1. there is some action a and some l such that $Der_{\Gamma,l}(P, a) = \emptyset$ and $Der_{\Gamma,l}(Q, a) \neq \emptyset$ or $Der_{\Gamma,l}(P, a) \neq \emptyset$ and $Der_{\Gamma,l}(Q, a) = \emptyset$
  2. or
     $Der(P, \tau) = \emptyset$ and $Der(Q, \tau) \neq \emptyset$ or $Der(P, \tau) \neq \emptyset$ and $Der(Q, \tau) = \emptyset$.

  In this case the terminal node is called *unsuccessful.*

The rules are similar in nature to those used in [CHM93] but a little more complicated. **UNWIND** replaces a label P = Q with a collection of intermediate nodes each labelled by expressions of the form $S = T$ where S, T are *finite sets* of extended processes. In turn the rule **SUM** can be applied to each of these nodes labelled by $S = T$ to obtain once more nodes properly labelled by expressions $P' = Q'$ where P', Q' are extended processes. Note that in the development of a tableau each application of **UNWIND** is necessarily followed by an application of **SUM**.

The two **SUB** rules allow us to replace one subprocess by another provided suitably relabelled versions of them appear higher up in the proof tree. However the replacement can only be made under special circumstances referred to in the side conditions of the rules. A node **n** *dominates* another node **m** if

- **m** appears higher up in the proof tree than **n**
- the rule **UNWIND** has been applied to the node **m**.

The side condition dictates that in order to apply a **SUBL** rule, for example, to a node **n**

1. it must be labelled by an equation of the form $\pi(P_1) \mid P_2 = Q$ where π is some order preserving cause set renaming which is bijective when restricted to $cs(P_1)$,

(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 3: The Tableau Rules

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

3. 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 defined as in the last section using as the finite set G the set $Gen(P) \cup Gen(Q)$, where $P = Q$ labels the root of the tree, and is based on some fixed ordering on G.

There is a further constraint on use of the **SUB** rules or rather on the development of a proof tree: the two **SUB** rules have precedence over **UNWIND** and **SUM** in that the latter two may only be applied if **SUBL** and **SUBR** are not applicable. This extra condition does not apply to the node **n** to which a **SUB** rule is being applied but to the entire proof tree above **n**.

A proof tree is developed for $P = Q$ by starting with a root labelled by $P = Q$ and systematically applying **UNWIND, SUM** and the **SUB** rules to non-terminal nodes, subject to this constraint. A proof tree is said to be *sucessful* if every leaf is a successful terminal node. An example of a complete tableau is as follows:

Example 4.1 Let $P = rec\, x.\ (a.x \mid b)$ and $Q = rec\, x.\ (b \mid a.x) + a.P \mid b$ with $P \sqsubset_{lex} Q$. To decide $P \sim_{lc} Q$ we develop a tableau $T(P = Q)$. Initially, rule **UNWIND** is applied yielding the two nodes labelled

$$\begin{array}{rcll} \{\{1\} \rhd P \mid b\} & = & \{b \mid \{1\} \rhd Q, \{1\} \rhd P \mid b\} & \text{for the derivatives of } a \\ \{a.P \mid \{1\} \rhd nil\} & = & \{\{1\} \rhd nil \mid a.Q, a.P \mid \{1\} \rhd nil\} & \text{for the derivatives of } b \end{array}$$

For these two nodes the tableau is extended for the first node by

$$\dfrac{\textbf{(SUM)}\quad \{\{1\} \rhd P \mid b\} = \{b \mid \{1\} \rhd Q, \{1\} \rhd P \mid b\}}{\dfrac{\textbf{(SUBR)}\quad \{1\} \rhd P \mid b \;=\; b \mid \{1\} \rhd Q}{\{1\} \rhd P \mid b \;=\; b \mid \{1\} \rhd P} \qquad \{1\} \rhd P \mid b \;=\; \{1\} \rhd P \mid b}$$

and for the second by

$$\dfrac{\textbf{(SUM)}\quad \{a.P \mid \{1\} \rhd nil\} = \{\{1\} \rhd nil \mid a.Q, a.P \mid \{1\} \rhd nil\}}{\dfrac{\textbf{(UNWIND)}\quad a.P \mid \{1\} \rhd nil \;=\; \{1\} \rhd nil \mid a.Q}{\dfrac{\textbf{(SUM)}\quad \{\{2\} \rhd P \mid \{1\} \rhd nil\} \;=\; \{\{1\} \rhd nil \mid \{2\} \rhd Q\}}{\dfrac{\textbf{(SUBR)}\quad \{2\} \rhd P \mid \{1\} \rhd nil \;=\; \{1\} \rhd nil \mid \{2\} \rhd Q}{\{2\} \rhd P \mid \{1\} \rhd nil \;=\; \{1\} \rhd nil \mid \{2\} \rhd P}}}} \qquad \textbf{A}$$

where $\mathbf{A}$ 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 3.5 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 3.8 we know that $V_k \equiv \pi(V_j) \mid R$ for some cause set renaming π which is bijective when restricted to $cs(V_j)$. But we now have all of the conditions necessary to apply the rule **SUBR** to the node $\mathbf{n_k}$. But by construction we know that **UNWIND** has been applied, which contradicts the condition that **UNWIND** can only be applied when a **SUB** rule is not applicable.

We have shown that $\omega_j \not\preceq \omega_k$ for any pair of nodes $\mathbf{n_j}, \mathbf{n_k}$ on the infinite subpath. We therefore have an infinite sequence of words over $\mathbb{N}^{|X|+1}$, namely $(\omega_i)_{i \in \mathbb{N}}$ with the property that $i < j$ implies $\omega_i \not\preceq \omega_j$. This contradicts Theorem 3.6.

Hence there is no infinite path in the tableau. □

Proposition 4.3 (Completeness)
Let $P, Q \in \mathrm{BPP}_l$. If $P \sim_{lc} Q$ then there is a successful tableau $T(P = Q)$.

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. □

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 $cs(P)(= cs(Q))$,
3. $(P, Q) \in \mathcal{R}^*$, $(Q, R) \in \mathcal{R}^*$ 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 $i \to i + 1$: Without loss of generality the label under consideration is of the form $(\pi(P_1) \mid P_2 = Q)$ and there is a dominated node labelled $P_1 = P_1'$ (or $P_1' = P_1$). By the definition of domination rule **UNWIND** is applied to $P_1 = P_1'$ in the next step, so by induction we know that both $(\pi(P_1') \mid P_2, Q)$ and (P_1, P_1') satisfy the requirement on matching moves above. By a simple but tedious case analysis one can show that also $(\pi(P_1) \mid P_2, Q)$ does so also.

□

Theorem 4.5 *Local cause equivalence is decidable on all* BPP_l *processes.*

Proof There are only finitely many different tableaux $T(R = S)$. □

5 Other Equivalences

Reflecting on the decidability algorithm of the last section we observe that it depends on very few properties of the *local cause* transition system or the semantic equivalence. The only direct use of the *local cause* transition system is to determine the sets of derivatives $Der_{\Gamma,l}(P, a)$ and $Der(P, \tau)$ which are required to be finite. Thus the algorithm can be applied to other equivalences based on causes and cause sets provided they have the properties described in Lemma 2.2 and they have finite derivative sets. In the following we consider two such candidates namely global cause equivalence and ST equivalence. We then go on to outline how the algorithm can also be adapted to decide the corresponding weak equivalences for a large subclass of BPP_l .

Global cause equivalence $\sim_{gc}$ is an alternative formulation of causal bisimulations, [DD89], [DD90] based on cause sets; see [Kie94] for a detailed exposition and a proof that it does indeed coincide with causal bisimulation. Global cause equivalence is defined in

the standard way, as a maximal bisimulation, but the underlying transition is a variation on that given in Section 2. It is obtained from the transition rules given in Figure 1 and Figure 2 replacing the rule for communication (L4) with

$$(\mathrm{L4}') \quad P \xrightarrow[\Gamma,l]{a}{}^{(gc)} P', \quad Q \xrightarrow[\Delta,k]{\bar{a}}{}^{(gc)} Q' \quad \text{implies} \quad P \mid Q \xrightarrow{\tau}{}^{(gc)} P'[\Delta/l] \mid Q'[\Gamma/k].$$

The intuition behind this rule is that a communication establishes causal links between the communicating partners. This is formalised by the injection of a copy of the partner's causes relevant to the communication. The equivalence is then defined in the same way as local cause equivalence, by replacing all lc indices by gc in Definition 2.1. Like local cause equivalence $\sim_{gc}$ is preserved by bijective cause set renamings (Lemma 2.2). Moreover $\sim_{gc}$ satisfies sufficient equations to ensure that extended processes can be transformed to *parforms* and the derivative sets in the underlying transition system remain finite. Therefore the proofs of completeness and soundness of the tableau method remain valid if $\sim_{lc}$ is replaced by $\sim_{gc}$ and we have:

Theorem 5.1 *Global cause equivalence, and therefore causal bisimulation, is decidable over* BPP_l . □

We now turn our attention to ST-equivalence, introduced for Petri Nets in [vGV87] and for process algebras in [AH93]. Here we follow the development in [AH93] and show how it can be simulated by a variant of cause-based transition systems. ST-equivalence is not motivated by localities or causes. Instead it takes the view that actions are non-atomic and uses an underlying transition system based on sub-actions consisting of the *starting* and the *finishing* of actions. However this type of operational semantics can easily be formulated as another variation on the *local cause* transition system. The transition system for visible moves consists of the rules in Figure 1 where the axiom for prefixing is replaced by the two new axioms

$$(\mathrm{ST1a}) \qquad \emptyset \rhd a.p \quad \xrightarrow[\emptyset,l]{a}{}^{(st)} \quad \{l\} \rhd a.p,$$

$$(\mathrm{ST1b}) \qquad \{k\} \rhd a.p \quad \xrightarrow[\{k\},l]{a}{}^{(st)} \quad \emptyset \rhd p.$$

Here the start of action a is simulated by performing action a with cause set $\emptyset$ while the end of a is simulated by a with the cause set $\{l\}$. Note that in (ST1b) the l cause is superfluous as it is not inserted into the term. It is simply there to ensure that transitions and terms of the ST transition system fit the general schema of our cause based operational semantics. If we restrict our attention to processes from CCS these transitions will only ever generate terms whose cause sets are either empty, $\emptyset$, or a singleton, $\{l\}$. Moreover, because of the side-condition on the rule (LG3), for any particular label l there will only be at most one occurrence of the cause set $\{l\}$ in a generated term.

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

(L1) $\quad \Gamma \rhd \mu.p \stackrel{\mu}{\longrightarrow}^{(st)} \Gamma \rhd p$

(L2) $\quad P \stackrel{\mu}{\longrightarrow}^{(st)} P'$ implies $P + Q \stackrel{\mu}{\longrightarrow}^{(st)} P'$, $Q + P \stackrel{\mu}{\longrightarrow}^{(st)} P'$

(L3) $\quad P \stackrel{\mu}{\longrightarrow}^{(st)} P'$ implies $P \mid Q \stackrel{\mu}{\longrightarrow}^{(st)} P' \mid Q$, $Q \mid P \stackrel{\mu}{\longrightarrow}^{(st)} Q \mid P'$

(L4) $\quad P \stackrel{a}{\longrightarrow}^{(st)} P',\ Q \stackrel{\bar{a}}{\longrightarrow}^{(st)} Q'$ implies $P \mid Q \stackrel{\tau}{\longrightarrow}^{(st)} P' \mid Q'$

(L5) $\quad \Gamma \rhd p[rec\, x.\ p/x] \stackrel{\mu}{\longrightarrow}^{(st)} P'$ implies $\Gamma \rhd rec\, x.\ p \stackrel{\mu}{\longrightarrow}^{(st)} P'$

(L6) $\quad P \equiv P',\ P \stackrel{\mu}{\longrightarrow}^{(st)} Q,$ implies $P' \stackrel{\mu}{\longrightarrow}^{(st)} Q$

Figure 4: Invisible ST Transitions

In the ST operational semantics of [AH93] 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, $\stackrel{\tau}{\longrightarrow}^{(st)}$, it is necessary to define the auxiliary relations $\stackrel{a}{\longrightarrow}^{(st)}$ for each action a. For completeness these are given in Figure 4.

Let $\sim_{st}$ be the bisimulation equivalence obtained obtained in the standard way from this transition system; note that only the transitions $\stackrel{a}{\underset{\Gamma,l}{\longrightarrow}}^{(st)}$ and $\stackrel{\tau}{\longrightarrow}^{(st)}$ are used in the definition of $\sim_{st}$. We show in the appendix that it coincides with ST-bisimulation as introduced in [AH93]. However this new formulation ensures that the requirements for the applicability of the decision procedure are satisfied and we obtain:

Theorem 5.2 *ST-equivalence is decidable on all* BPP *processes.* □

Let us now consider the weak versions of local cause, global cause and ST-equivalence.

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 $\stackrel{a}{\Longrightarrow}{}^{(lc)}_{\Gamma,l}$ be the least relation which satisfies

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

We also use $\stackrel{\varepsilon}{\Longrightarrow}{}^{(lc)}$ to denote the reflexive transitive closure of $\stackrel{\tau}{\longrightarrow}{}^{(lc)}$.

Based on these transitions weak local cause equivalence is defined in the standard way:

Definition 5.3 [Weak Local Cause Equivalence]
A symmetric relation $R \subseteq BPP \times BPP$ is called a *weak local cause bisimulation* iff $R \subseteq G(R)$ where $(p,q) \in G(R)$ iff

(i) $p \stackrel{\varepsilon}{\Longrightarrow}{}^{(lc)} p'$ implies $q \stackrel{\varepsilon}{\Longrightarrow}{}^{(lc)} q'$ for some $q' \in BPP_l$ with $(p',q') \in R$

(ii) $p \stackrel{a}{\Longrightarrow}{}^{(lc)}_{\Gamma,l} p', l = new(cau(p) \cup cau(q))$, implies $q \stackrel{a}{\Longrightarrow}{}^{(lc)}_{\Gamma,l} q'$ for some $q' \in BPP_l$ with $(p',q') \in R$.

Two processes p and q are *weak local cause equivalent*, $p \approx_{lc} q$, iff there is a local cause bisimulation R such that $(p,q) \in R$. □

It is fairly easy to see how in principle the tableau algorithm has to be adapted in order to handle this new equivalence. The modification concerns the unwinding of processes and the sets of derivatives to be compared. These sets of derivatives are now defined by:

$$\begin{aligned} WDer_{\Gamma,l}(P,a) &= \{Q \mid P \stackrel{a}{\Longrightarrow}{}^{(lc)}_{\Gamma,l} Q\}, \\ WDer(P,\varepsilon) &= \{Q \mid P \stackrel{\varepsilon}{\Longrightarrow}{}^{(lc)} Q\}. \end{aligned}$$

Unfortunately although we assume that processes are *guarded* these sets may still be infinite. So if we are to adapt the basic decision procedure we must restrict our attention to a subclass where these are guaranteed to be finite.

A process $P \in BPP_l$ is said to be *convergent* if there is no infinite sequence

$$P = P_1 \stackrel{\tau}{\longrightarrow}{}^{(lc)} P_2 \stackrel{\tau}{\longrightarrow}{}^{(lc)} P_3 \stackrel{\tau}{\longrightarrow}{}^{(lc)} \ldots$$

and *h-convergent* if Q is convergent whenever $P \rightarrow P_1 \rightarrow P_2 \ldots P_n \rightarrow Q$ where $P_i \rightarrow P_{i+1}$ stands for $P_i \stackrel{\tau}{\longrightarrow}{}^{(lc)} P_{i+1}$ or $P_i \stackrel{a}{\longrightarrow}{}^{(lc)}_{\Gamma,l} P_{i+1}$ for some a, Γ. Intuitively this means P will

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 $\{g_1, \ldots, g_n\}$. The arcs to the output place g_i are weighted according to the number of occurrences of g_i in the polynomial presentation of g
- for two Petri net transitions introduced by the previous item labelled a and $\bar{a}$ introduce a new transition labelled τ with input and output places determined by those of the two transitions being considered
- take as initial marking M_0 the multiset determined by the polynomial presentation of g.

It is easily seen that the *CCS* process p is h-convergent if and only if the net $PN(p)$ has a reachable marking which enables an infinite firing sequence consisting of τ-labelled transitions only. The latter property has been shown to be decidable by Vogler in [Vog92] (Theorem 3.2.9) using results of [VJ85]. □

Proposition 5.5 *If* $P \in \mathrm{BPP}_l$ *and* P *is h–convergent then* $Der_{\Gamma,l}(P,a)$, $Der(P,\tau)$ *and* $Der(P,\varepsilon)$ *are finite for all* $l \in \mathcal{C}$, $\Gamma \in \mathcal{CS}$ *and* $a \in Act$.

Proof A simple application of König's Lemma. □

(WeakUNWIND)

$$\frac{P \;=\; Q}{\{WDer_{\Gamma,l}(P,a) = WDer_{\Gamma,l}(Q,a)\} \qquad WDer(P,\varepsilon) = WDer(Q,\varepsilon)}$$

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

Figure 5: Modified Tableau Rule for Weak Equivalences

So we can adapt our decision procedure to decide $\approx_{lc}$ for the decidable sub-class of *h-convergent* processes; we need only replace the **UNWIND** rule with the **WeakUNWIND** rule given in Figure 5. As in the strong case one can show that tableaux generated by the new rules will be sound and that the algorithm is complete. With the modified rule the proofs in Section 4 can be easily modified to obtain:

Theorem 5.6 *Weak local cause equivalence is decidable on h–convergent* BPP_l *processes.* □

Moreover by changing in an appropriate manner the definition of the sets of derivatives the decision also works for weak versions of the other two equivalences and therefore we also obtain:

Theorem 5.7 *Weak global cause equivalence and weak ST-equivalence are decidable on h–convergent* BPP_l *processes.* □

6 Conclusions

In this paper we have generalised the method of deciding bisimulation equivalence for basic parallel processes of [CHM93] for three well-known non-interleaving equivalences. We have followed closely the general strategy of that paper but because of the complex nature of the equivalences, considerable modifications were required. The main difficulty was in finding a condition for the replacement of one process by a simpler one which at the same time guaranteed termination of the tableaux. The chosen solution, which uses cause set bijections, enabled us to guarantee termination employing the standard theorem by Higman, [Lot83], from formal language theory.

We also considered decidability of the corresponding weak equivalences but our results only apply to a subclass of processes, namely those which can never diverge. For arbitrary BPP processes these problems are still open, as it is for weak bisimulation equivalence. However at least we have shown that the subclass of processes we can handle is decidable.

The results presented here also apply to Petri nets as basic parallel processes correspond exactly to a certain class of Petri nets (cf. [Hir94] and [Esp94]). In fact in the case of global cause equivalence the way we treat histories can be seen as a generalisation of Vogler's work for safe nets and history preserving bisimulation ([Vog91b, Vog91a]). Note that in general bisimulation equivalence is not decidable for arbitrary Petri nets ([Jan94]).

References

[AH93] L. Aceto and M. Hennessy. Towards action refinement in process algebra. *Information and Computation*, 103(2):204–269, 1993.

[BCHK93] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, (114):31–61, 1993.

[BCHK94] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. *Formal Aspects of Computing*, (6):165–200, 1994.

[CH89] I. Castellani and M. Hennessy. Distributed bisimulations. *Journal of the Association for Computing Machinery*, 10(4):887–911, 1989.

[CHM93] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation equivalence is decidable for all basic parallel processes. In E. Best, editor, *Proceedings of CONCUR 93*, number 715 in Lecture Notes in Computer Science, pages 143–157. Springer–Verlag, 1993.

[Chr92] S. Christensen. Distributed bisimilarity is decidable for a class of infinite–state systems. In *Proceedings of CONCUR 92*, number 630 in Lecture Notes in Computer Science. Springer–Verlag, 1992.

[Chr93] S. Christensen. *Decidability and Decomposition in Process Algebras.* Ph.d. thesis, University of Edinburgh, 1993.

[CHS92] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context–free processes. In *Proceedings of CONCUR 92*, number

630 in Lecture Notes in Computer Science, pages 138–147. Springer–Verlag, 1992.

[CPS89] R. Cleaveland, J. Parrow, and B. Steffen. A semantics based verification tool for finite state systems. In *Proceedings of the* 9^{th} *International Symposium on Protocol Specification, Testing and Verification*, North Holland, 1989.

[DD89] P. Darondeau and P. Degano. Causal trees. In *Proceedings of ICALP 89*, number 372 in Lecture Notes in Computer Science, pages 234–248, 1989.

[DD90] P. Darondeau and P. Degano. Causal trees, interleaving + causality. In I. Guessarian, editor, *Semantics of Systems of Concurrent Processes*, number 469 in Lecture Notes in Computer Science, pages 239–255, 1990.

[DNM89] P. Degano, R. De Nicola, and U. Montanari. Partial ordering descriptions and observations of nonderterministic concurrent processes. In *Proceedings REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, number 354 in Lecture Notes in Computer Science, pages 438–466, 1989.

[Esp94] J. Esparza. On the uniform word problem for commutative context-free grammars. unpublished manuscript, University of Edinburgh, 1994.

[Gol88] U. Goltz. On representing CCS programs by finite Petri nets. In M. P. Chytil et al., editor, *Prooceedings of* MFCS 88, number 324 in Lecture Notes in Computer Science, pages 339–350. Springer–Verlag, 1988.

[Hir94] Y. Hirshfeld. Petri nets and the equivalence problem. In K. Meinke, editor, *Proceedings of* CSL, Lecture Notes in Computer Science, 1994. to appear.

[Jan94] Y. Jancăr. Decidability questions for bisimilarity of Petri nets and some related problems. In P. Enjalbert et al., editor, *Proceedings of* STACS 94, number 775 in Lecture Notes in Computer Science, pages 581–592. Springer–Verlag, 1994.

[Kie94] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 1994. to appear, Revision of *Local and global causes*, report TUM–I9132.

[Lot83] M. Lothaire. *Combinatorics on Words.* Addison–Wesley Publishing Company, 1983.

[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 369 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):643–674, 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-I9113, 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 [AH93] 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 [AH93] 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 [AH93] 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 ST-equivalence:

Definition 7.1 [ST-quivalence]
A symmetric relation $R \subseteq ST \times ST$ is called a *ST bisimulation* iff $R \subseteq S(R)$ where $(P,Q) \in S(R)$ iff

(i) $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ for some $Q' \in ST$ such that $(P',Q') \in R$

(ii) $P \xrightarrow{\lambda} P'$, $\lambda \in \{S(a_l), F(a_l) \mid a \in Act, l \in \mathcal{C}\}$, implies $Q \xrightarrow{\lambda} Q'$
for some $Q' \in ST$ such that $(P',Q') \in R$.

Two processes P and Q are *ST equivalent*, $P \sim Q$, if and only if there is a ST bisimulation R such that $(P,Q) \in R$.

□

In [AH93] in order to define ST-equivalence it was necessary to parameterise the relations R on partial bijections over the cause sets. This is unnecessary here because of each cause has at most a unique occurrence in a process.

We now compare this equivalence with our version, $\sim_{st}$, defined using cause sets. Let BPP_{ST} be the subset of BPP_l consisting of all processes which (up to $\equiv$)

1. have at most one occurrence of any particular cause
2. and only contains cause sets of the form $\emptyset$ or $\{l\}$.

Note that this is the set which is reachable from BPP using $\longrightarrow^{(st)}$ transitions. To simplify the comparison we strengthen the congruence $\equiv$ to the congruence obtained

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)	$\mu.p \xrightarrow{\mu} p$		
(S2)	$a.p \xrightarrow{S(a_l)} F(a_l).p$	$l \in \mathcal{C}$	
(S3)	$F(a_l).p \xrightarrow{F(a_l)} p$		
(S4)	$P \xrightarrow{\lambda} P'$	implies	$P + Q \xrightarrow{\lambda} P'$ $Q + P \xrightarrow{\lambda} P'$
(S5)	$P \xrightarrow{\lambda} P'$	implies	$P \mid Q \xrightarrow{\lambda} P' \mid Q$ $Q \mid P \xrightarrow{\lambda} Q \mid P'$
(S6)	$P \xrightarrow{a} P', \quad Q \xrightarrow{\bar{a}} Q'$	implies	$P \mid Q \xrightarrow{\tau} P' \mid Q'$
(S7)	$p[rec\, x.\ p/x] \xrightarrow{\lambda} P'$	implies	$p \xrightarrow{\lambda} P'$

Figure 6: ST Transitions

from the equations generating $\equiv$ without the equation for commutativity of $|$. In order not to lose any transition due to this change we add the rule (ST3′)

$$\text{(ST3}') \qquad P \xrightarrow[\Gamma,l]{a}{}^{(st)} P',\ l \notin cau(Q), \qquad \text{implies} \qquad Q \mid P \xrightarrow[\Gamma,l]{a}{}^{(st)} Q \mid P'$$

and modify the rule for $\equiv$ respectively. Similar changes are needed for τ transitions.

Each term in BPP_{ST} represents uniquely a term in ST and vice versa. To see this let $st : BPP_{ST} \to ST$ and $lg : ST \to BPP_{ST}$ be the mappings defined by

$$st(P) = \begin{cases} p & \text{if } P = \emptyset \rhd p \\ F(a_l).p & \text{if } P = \{l\} \rhd a.p \\ st(P_1) \mid st(P_2) & \text{if } P = P_1 \mid P_2 \end{cases} \qquad lg(P) = \begin{cases} \emptyset \rhd p & \text{if } P = p \in CCS \\ \{l\} \rhd a.p & \text{if } P = F(a_l).p \\ lg(P_1) \mid lg(P_2) & \text{if } P = P_1 \mid P_2 \end{cases}$$

We have $st \circ lg = id$ and $lg \circ st = id$ where id is the identity mapping. Moreover it is straightforward to verify that transitions are also preserved by these translations:

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

(a) $P \stackrel{S(a_l)}{\rightarrow} P'$ implies $lg(P) \stackrel{a}{\underset{\emptyset,l}{\longrightarrow}}{}^{(st)} lg(P')$,

(b) $P \stackrel{F(a_l)}{\rightarrow} P'$ implies $lg(P) \stackrel{a}{\underset{\{l\},k}{\longrightarrow}}{}^{(st)} lg(P')$ for any $k \in \mathcal{C} \setminus cau(P)$,

(c) $P \stackrel{\mu}{\rightarrow} P'$ implies $lg(P) \stackrel{\mu}{\longrightarrow}{}^{(st)} lg(P')$.

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

(a) $P \stackrel{a}{\underset{\emptyset,l}{\longrightarrow}}{}^{(st)} P'$ implies $st(P) \stackrel{S(a_l)}{\rightarrow} st(P')$,

(b) $P \stackrel{a}{\underset{\{l\},k}{\longrightarrow}}{}^{(st)} P'$ implies $st(P) \stackrel{F(a_l)}{\rightarrow} st(P')$,

(c) $P \stackrel{\mu}{\longrightarrow}{}^{(st)} P'$ implies $st(P) \stackrel{\mu}{\rightarrow} st(P')$.

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. □