BRICS

Basic Research in Computer Science

Further Results on Partial Order Equivalences on Infinite Systems

Kim Sunesen

BRICS Report Series RS-98-6

ISSN 0909-0878 March 1998

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/98/6/`

Further Results on Partial Order Equivalences on Infinite Systems

Kim Sunesen
BRICS*
Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Aarhus C.
ksunesen@daimi.aau.dk

Abstract

In [26], we investigated decidability issues for standard language equivalence for process description languages with two generalisations based on traditional approaches for capturing non-interleaving behaviour: *pomset equivalence* reflecting global causal dependency, and *location equivalence* reflecting spatial distribution of events.

In this paper, we continue by investigating the role played by TCSP-style renaming and hiding combinators with respect to decidability. One result of [26] was that in contrast to pomset equivalence, location equvialence remained decidable for a class of processes consisting of finite sets of BPP processes communicating in a TCSP manner. Here, we show that location equivalence becomes undecidable when either renaming or hiding is added to this class of processes.

Furthermore, we investigate the weak versions of location and pomset equivalences. We show that for BPP with τ prefixing, both weak pomset and weak location equivalence are decidable. Moreover, we show that weak location equivalence is undecidable for BPP semantically extended with CCS communication.

---

*Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

1 Introduction

In this paper, we investigate the decidability of non-interleaving linear-time behavioural equivalences on infinite-state systems described by process algebraic languages such as CCS [19] and TCSP [5].

Our results contribute to the ongoing and systematic investigation of decidability of problems about infinite-state systems. But, our results may also be seen as a contribution to the search for elucidating the sometimes delicate computational trade-offs involved in moving from the standard view of interleaving to more intentional non-interleaving views of behaviour.

Process algebraic languages, notably CCS [19], TCSP [5] and ACP [3], have proved a rich source of infinite-state systems, and moreover, an appropriate framework for a systematic study based on the choice of combinators. One of the most interesting suggestions is *Basic Parallel Processes*, BPP, introduced in [7]. BPPs are recursive expressions constructed from inaction, action, variables, and the standard operators prefixing, choice and parallel compositions. By removing the parallel operator one obtains a calculus with exactly the same expressive power as finite automata. BPPs can hence be seen as arising from a minimal concurrent extension of finite automata and therefore a natural starting point when exploring concurrent infinite-state systems.

The notion of behavioural equivalences is a cornerstone in the theory of process algebraic languages. It is common to classify behavioural equivalences into branching-time and linear-time equivalences depending on whether or not the branching structure of the behaviour is taken into account or not. Another central distinction is made between strong and weak equivalences. The distinction arises when actions are divided into visible and invisible actions. Often, there is just one invisible or silent action denoted τ. In the strong case, the invisible τ action is an action no different from the other actions whereas in the weak case equivalence is based on abstracting away from the invisible actions only requiring equivalent behaviour with respect to visible actions.

Many results about decidability are known for interleaving equivalences such as bisimulation and language equivalences on infinite-state systems, see [8, 11] for surveys. Also, for non-interleaving bisimulation equivalences results are known, see [6, 17].

In [26], we compared standard language equivalence with two generalisations based on traditional approaches capturing non-interleaving behaviour. The first known as *pomset equivalence* was based on *pomsets* representing global causal dependency [22], and the second known as *location equivalence* on *locality* [4] representing spatial distribution of events. The two notions of non-interleaving equivalences were shown to be decidable on BPP contrasting the result of Hirshfeld [10] that language equivalence is undecidable. Moreover, larger subclasses of CCS and TCSP obtained by adding different means for communication were studied. It was hence shown that when adding the parallel combinator of Milner's CCS to BPP, BPP_M, we keep the decidability of both location and pomset equivalence whereas when adding the parallel combinator of Hoare's TCSP both become undecidable. Also, for a non-trivial subclass of processes between BPP and TCSP, BPP_S, consisting of finite sets of communicating BPP processes, it was shown that location equivalence is decidable whereas pomset equivalence is not.

The work presented in this paper continues by investigating the role of the renaming and hiding combinators with respect to decidability, and by investigating the weak versions of pomset and location equivalence.

First, we look at BPP extended with renaming and hiding combinators, and show by a reduction to the same problem for BPP that both pomset and location equivalence remain decidable. Second, we turn to BPP_S. It follows from the undecidability for BPP_S that pomset equivalence for BPP_S extended with renaming or hiding combinators is undecidable. Here, we show that adding any of the combinators makes a significant difference for location equivalence which becomes undecidable. The result is shown by a reduction to the halting problem for two-counter machines base on weak encodings of counter machines. Our results are summarised in the table below where *yes* indicates decidability and *no* undecidability. The results of the first column are all direct consequences of Hirshfeld's result on BPP [10]. The second and third show our results:

	Language equiv.	Pomset equiv.	Location equiv.
BPP	no	yes	yes
BPP + renaming and/or hiding	no	yes	yes
BPP_S	no	no	yes
BPP_S + renaming and/or hiding	no	no	no

Furthermore, we turn to the weak case. We consider three extensions: BPP with τ prefixing BPP^τ, BPP with CCS-communication BPP_M, and BPP with both τ prefixing and CCS-communication BPP^τ_M. We show that for BPP^τ, both weak pomset and weak location equivalence are decidable. This points out a current contrast to the results in the interleaving world where there are currently no positive results on deciding weak equivalences for the full class of BPP^τ. In fact, one major open problem is the decidability of weak bisimulation on BPP^τ, see [9, 13]. In [17], a number of non-interleaving weak bisimulations were shown to be decidable for the class of so–called h-convergent BPP^τ_M processes which are processes that cannot evolve into a divergent process. Also, positive results are known for the asymmetric problem of deciding weak equivalences between a finite-state system and an infinite-state system such as BPP^τ, see [18, 14]. As a natural next step, we look at the class of processes BPP^τ_M obtained by semantically extending BPP^τ with the communication rule of Milner [19]. We show that for BPP^τ_M, (strong) location equivalence remains decidable whereas weak location equivalence becomes undecidable. The positive result is shown by a reduction to the same problem for BPP^τ and the negative result is shown by a reduction to the halting problem for two-counter machines. For the problem of deciding pomset equivalence on BPP^τ_M, we give and *effective* characterisation for the strong case in terms of a containment problem between finite tree automata and a family of finite tree automata. Our results are summarised in the table below where *yes* indicates decidability, *no* indicates undecidability, and ? means that the question is still open. The results of the first column are all direct consequences of Hirshfeld's result on BPP [10]. The second and third show our results:

	Weak		
	Language equiv.	Pomset equiv.	Location equiv.
BPP	no	yes	yes
BPP^τ	no	yes	yes
BPP_M	no	?	no
BPP_M^τ	no	?	no
CCS	no	no	no

The rest of the paper is organised as follows. In Section 2, we define fairly standard TCSP and CCS-style languages, and along the lines of [26] we augment the standard transitional semantics so that not only actions but also information of their locality is observed. Moreover, we define the subclasses studied in the following sections. Language, pomset and location equivalence as presented in [26] are defined in Section 3. Section 4 is devoted to the study of the renaming and hiding combinator. Weak versions of language, pomset and location equivalence are defined in Section 5. In Section 6, 7 and 8, we investigate BPP^τ, BPP_M, and BPP_M^τ, respectively. We conclude with discussions on some loose ends and suggestions for future work.

2 TCSP/CCS-style languages

We start by defining the abstract syntax and semantics of TCSP [12, 21] and CCS [19] – style a languages. The definitions are fairly standard. As usual, we fix a countably infinite set of *actions* $\Lambda = \{\alpha, \beta, \ldots\}$. Then, $\overline{\Lambda} = \{\bar{\alpha}, \bar{\beta}, \ldots\}$ is the set of complement actions such that $^-$ is a bijection between Λ and $\overline{\Lambda}$, mapping $\bar{\bar{\alpha}}$ to α. Let $\mathcal{A}ct = \Lambda \cup \overline{\Lambda}$ and let $\mathcal{A}ct_\tau = \mathcal{A}ct \cup \{\tau\}$ be the set of *actions*, where τ is a distinguished action not in $\mathcal{A}ct$. τ is known as the *invisible* action. Any other action is *visible*. Also, fix a countably infinite set of *variables* $\text{Var} = \{X, Y, Z, \ldots\}$. A renaming f is an endofunction on $\mathcal{A}ct_\tau$ such that τ is preserved and reflected, that is, $f^{-1}(\{\tau\}) = \{\tau\}$.

The set of process expressions of TCSP is defined by the abstract syntax

$$E \;::=\; 0 \mid X \mid \sigma.E \mid E + E \mid E \parallel_A E \mid E[f] \mid E\backslash\backslash L$$

where X is in Var, σ in $\mathcal{A}ct_\tau$, A and L are subsets of $\mathcal{A}ct$ and f is a renaming. All constructs are standard. 0 denotes inaction, X a process variable, $\sigma.$ prefixing, $+$ non-deterministic choice, $\parallel_A$ TCSP parallel composition of processes executing independently with forced synchronisation on actions in the *synchronisation set*, A, $[f]$ renaming of actions according to the renaming f, $\backslash\backslash L$ hiding of the actions in L. For convenience, we shall write $\parallel$ for $\parallel_\emptyset$.

The set of CCS process expressions is defined by the abstract syntax

$$E \;::=\; 0 \mid X \mid \sigma.E \mid E + E \mid E \parallel E \mid E\backslash L \mid E[f]$$

where X is in Var, σ in $\mathcal{A}ct_\tau$, L a subset of Λ and f is renaming. 0, X, $\sigma.$, $+$, and $[f]$ are as for TCSP. $\parallel$ is CCS parallel composition of processes executing independently with the possibility of pairwise CCS-synchronisation and $\backslash L$ is CCS-restriction. Note that we do not put the usual requirement of preservation of complement [19] on the renaming (relabelling) function because our results go through with or without.

A *process family* is a family of recursive equations $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, 2\ldots, n\}$, where $X_i \in \mathit{Var}$ are distinct variables and E_i are process expressions containing at most variables in $\mathit{Var}(\Delta) = \{X_1, \ldots, X_n\}$. A *process* E is a process expression of with a process family Δ such that all variables occurring in E, $\mathit{Var}(E)$, are contained in $\mathit{Var}(\Delta)$. We shall often assume the family of a process to be defined implicitly. Dually, a process family denotes the process defined by its *leading variable* X_1, if not mentioned explicitly. Let $\mathcal{A}ct(E)$ denote the set of actions occurring in process E and its associated family. A process expression E is *guarded* if each variable in E occurs within some subexpression $\sigma.F$ of E. Following [19], we also consider the more restricted kind of guarding where furthermore the "guard" cannot be a τ action, that is, $\sigma \neq \tau$, in this case we say that the process is Milner guarded. A process family is (Milner) guarded if for each equation the right side is (Milner) guarded. A process E with family Δ is (Milner) guarded if E and Δ are (Milner) guarded. Throughout the paper we shall only consider guarded processes and process families.

We enrich the standard operational semantics of TCSP [27] and CCS [19] by adding information to the transitions allowing us to observe an action together with its location. More precisely, the location of an action in a process P is the path from the root to the action in the concrete syntax tree represented by a string over $\{0, 1\}$ labelling left and right branches of $\|_A$-nodes with 0 and 1, respectively, and all other branches with the empty string ϵ.

Let $\mathcal{L} = \mathcal{P}(\{0, 1\}^*)$, i.e. finite subsets of strings over $\{0, 1\}^*$, and let l range over elements of $\mathcal{L}$. We interpret prefixing a symbol to $\mathcal{L}$ as prefixing elementwise, i.e. $0l = \{0s \mid s \in l\}$. With this convention, any process determines a $(\mathcal{A}ct_\tau \times \mathcal{L})$-labelled transition system with states the set of process expressions reachable from the leading variable and transitions given by the transitions rules of Table 1, 2, 3, and 4 for TCSP, and Table 1, 5, 6, and 3 for CCS. The set of *computations* of a process, E, is as usual defined as sequences of transitions, decorated by action and locality information:

$$c:\ E = E_0 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

We let $loc(c)$ denote the set of locations occurring in c, i.e. $loc(c) = \bigcup_{1 \leq i \leq n} l_i$.

$$\sigma.E \xrightarrow[\{\epsilon\}]{\sigma} E \qquad (\textit{prefix}) \qquad\qquad \frac{E \xrightarrow[l]{\sigma} E'}{X \xrightarrow[l]{\sigma} E'}, (X \stackrel{\mathrm{def}}{=} E) \in \Delta \quad (\textit{unfold})$$

$$\frac{E \xrightarrow[l]{\sigma} E'}{E + F \xrightarrow[l]{\sigma} E'} \quad (sum_l) \qquad\qquad \frac{F \xrightarrow[l]{\sigma} F'}{E + F \xrightarrow[l]{\sigma} F'} \quad (sum_r)$$

$$\frac{E \xrightarrow[l]{\sigma} E'}{E \|_A F \xrightarrow[0l]{\sigma} E' \|_A F}, \ \sigma \notin A \quad (par_l) \qquad\qquad \frac{F \xrightarrow[l]{\sigma} F'}{E \|_A F \xrightarrow[1l]{\sigma} E \|_A F'}, \sigma \notin A \quad (par_r)$$

Table 1: Transition rules for TCSP/CCS.

$$\frac{E \xrightarrow[l_0]{\sigma} E' \qquad F \xrightarrow[l_1]{\sigma} F'}{E \|_A F \xrightarrow[0l_0 \cup 1l_1]{\sigma} E' \|_A F'}, \ \sigma \in A \quad (\mathrm{com})$$

Table 2: Transition rule for TCSP communication.

Example 1 Consider the process

$$p_1 = a.b.c.0 \parallel_{\{b\}} b.0.$$

The following is an example of an associated computation (representing the unique maximal run)

$$p_1 \xrightarrow[\{0\}]{a} b.c.0 \parallel_{\{b\}} b.0 \xrightarrow[\{0,1\}]{b} c.0 \parallel_{\{b\}} 0 \xrightarrow[\{0\}]{c} 0 \parallel_{\{b\}} 0.$$

Consider alternatively the process

$$p_2 = a.b.0 \parallel_{\{b\}} b.c.0$$

with computation

$$p_2 \xrightarrow[\{0\}]{a} b.0 \parallel_{\{b\}} b.c.0 \xrightarrow[\{0,1\}]{b} 0 \parallel_{\{b\}} c.0 \xrightarrow[\{1\}]{c} 0 \parallel_{\{b\}} 0.$$

■

$$\frac{E \xrightarrow[l]{\sigma} F}{E[f] \xrightarrow[l]{f(\sigma)} F[f]}, \quad (ren)$$

Table 3: Transition rule for TCSP/CCS renaming.

$$\frac{E \xrightarrow[l]{\sigma} F}{E\backslash\backslash L \xrightarrow[l]{\sigma} F\backslash\backslash L}, \sigma \notin L \quad (hid_1) \qquad\qquad \frac{E \xrightarrow[l]{\sigma} F}{E\backslash\backslash L \xrightarrow[l]{\tau} F\backslash\backslash L}, \sigma \in L \quad (hid_2)$$

Table 4: Transition rules for TCSP hiding.

$$\frac{E \xrightarrow[l_0]{\sigma} E' \qquad F \xrightarrow[l_1]{\bar{\sigma}} F'}{E \parallel F \xrightarrow[0l_0 \cup 1l_1]{\tau} E' \parallel F'} \quad (\tau - com)$$

Table 5: Transition rule for CCS communication.

$$\frac{E \xrightarrow[l]{\sigma} F}{E\backslash L \xrightarrow[l]{\sigma} F\backslash L}, \sigma, \bar{\sigma} \notin L \quad (res)$$

Table 6: Transition rule for CCS restriction.

2.1 BPP, BPP^τ, BPP_M, and BPP_M^τ

We shall investigate a number of syntactic as well as semantic subsets of TCSP and CCS. The calculus known as Basic Parallel Processes [7] BPP is a syntactic subset of CCS and TCSP which can be seen as the largest common subset of these (except for the renaming combinator). The abstract syntax of BPP expressions is

$$E \quad ::= \quad 0 \mid X \mid \sigma.E \mid E+E \mid E \parallel E$$

where $\sigma \in \mathcal{A}ct$ (note that τ prefixing is not allowed), and the semantics is given by the rules in Table 1, in particular there is no rule for communication.

BPP^τ, is the subset of CCS obtained by adding τ prefixing, that is, syntactically the prefixing combinator $\sigma.$ is extended to all $\sigma \in \mathcal{A}ct_\tau$ the semantics is the same as for BPP

BPP_M, is the subset of CCS obtained by adding the transition rule τ-*com* of Table 5 to BPP and hence introducing CCS-synchronisation. Since there is no restriction operator in BPP_M communication cannot be forced. Whenever a communication occurs in a computation, also the computation with the communicating actions occurring separately is possible. Conversely, if there is a computation in which two complementing actions occur independently then the same computation except from the two actions now communicating exists.

BPP_M^τ, is the subset of CCS obtained by adding both τ prefixing and the transition rule τ-*com* of Table 5 to BPP.

In the following, we shall also consider the subsets of TCSP and CCS obtained by adding the renaming and the hiding combinator to the syntax, and the rules of Table 3 and 4 to the semantics of BPP, BPP^τ, $\mathrm{BPP}_M$and BPP_M^τ, we called these classes BPP, BPP^τ, $\mathrm{BPP}_M$and BPP_M^τ *with renaming and hiding*, respectively.

We shall make convenient use of the following structural congruence.

Definition 2 Let $\equiv$ be the least congruence on BPP_M^τ expressions with respect to all operators such that the following laws hold.

Abelian monoid laws for $+$:

$$\begin{aligned} E+F &\equiv F+E \\ E+(F+G) &\equiv (E+F)+G \\ E+0 &\equiv E \end{aligned}$$

Abelian monoid laws for $\parallel$:

$$\begin{aligned} E \parallel F &\equiv F \parallel E \\ E \parallel (F \parallel G) &\equiv (E \parallel F) \parallel G \\ E \parallel 0 &\equiv E \end{aligned}$$

Idempotence law for $+$:

$$E+E \equiv E$$

Linear-time laws:

$$\begin{aligned} (E+F) \parallel G &\equiv (E \parallel G) + (F \parallel G) \\ \sigma.(E+F) &\equiv \sigma.E + \sigma.F \end{aligned}$$

■

As parallel composition is commutative and associative, it is convenient to represent a parallel composition $X_0 \parallel \ldots \parallel X_k$ by the multiset $\{|X_0, \ldots, X_k|\}$. Inaction 0 is represented by the empty multiset. For a set *Var*, we denote by $Var^{\otimes}$ the set of all finite multisets over *Var*.

Definition 3 A BPP_M^τ family $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, 2 \ldots, n\}$ is in *(quasi) normal form* if and only if each expression E_i is of the form

$$E_i \equiv \sum_{j=1}^{n_i} \sigma_{ij} \alpha_{ij}$$

where $\sigma_{ij} \in \mathcal{A}ct_\tau$ and $\alpha_{ij} \in Var(\Delta)^{\otimes}$. ■

For a BPP_M^τ family $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, 2 \ldots, n\}$ in normal form the branching bound is the maximal cardinality of the multisets appearing in the sums, that is, $max\{|\alpha_{ij}| \mid i \in [n],\ j \in [n_i]\}$. We sometimes write $\sigma\alpha \in E_i$ to denote that there is a $j \in [n_i]$ such that $\sigma = \sigma_{ij}$ and $\alpha = \alpha_{ij}$.

2.2 BPP_S

A natural restriction when dealing with non-interleaving behaviours is to allow only parallel composition in a fixed static setup, see e.g. [2, 1]. This of course leads to finite-state systems. We generalise the idea to possibly infinite-state systems. Let BPP_S be the syntactic subset of TCSP obtained by allowing only synchronisation, i.e. the $\parallel_A$ operator with $A \neq \emptyset$, at the top level and restricting the synchronisation sets to be the set of all actions possible in either of the components.

A BPP_S process can hence be seen as a fixed set of BPP processes synchronising on every action. Formally, a BPP_S expression is given by the abstract syntax

$$E \quad ::= \quad X_1 \parallel_\Sigma \ldots \parallel_\Sigma X_l,$$

where $\Sigma \supseteq \mathcal{A}ct$. A BPP_S family is a process family $\Delta = \{X \stackrel{\text{def}}{=} X_1 \parallel_\Sigma \ldots \parallel_\Sigma X_l\} \cup \Delta'$ with leading variable X such that the leading variable X does not occur on any right-side, the variables $X_1, \ldots, X_l$ are contained in $Var(\Delta')$, the synchronisation set Σ is a superset of the actions $\mathcal{A}ct(\Delta')$ in Δ', and Δ' is a BPP family in normal form. A BPP_S process E is a BPP_S expression with a process family Δ' such that $\{X_0 \stackrel{\text{def}}{=} E\} \cup \Delta'$ is a BPP_S family with leading variable X_0. We call l the *arity* of Δ.

BPP_S *with renaming* is all TCSP processes of the form $\Delta = \{X \stackrel{\text{def}}{=} (X_1 \parallel_\Sigma \ldots \parallel_\Sigma X_l)[f]\} \cup \Delta'$ such that $\{X \stackrel{\text{def}}{=} X_1 \parallel_\Sigma \ldots \parallel_\Sigma X_l\} \cup \Delta'$ is a BPP_S process and f is a renaming.

BPP_S *with hiding* is all TCSP processes of the form $\Delta = \{X \stackrel{\text{def}}{=} (X_1 \parallel_\Sigma \ldots \parallel_\Sigma X_l)\backslash\backslash L\} \cup \Delta'$ such that $\{X \stackrel{\text{def}}{=} X_1 \parallel_\Sigma \ldots \parallel_\Sigma X_l\} \cup \Delta'$ is a BPP_S process.

3 Language, pomset, and location equivalence

Let $\sqsubseteq$ be the prefix ordering on $\{0,1\}^*$, extended to sets, i.e. for $l, l' \in \mathcal{L}$

$$l \sqsubseteq l' \iff \exists s \in l, s' \in l'.\ s \sqsubseteq s'.$$

As usual, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a given computation

$$c : E_0 \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n,$$

we define the location dependency ordering over $[n]$ as follows:

$$i \leq_c j \iff l_i \sqsubseteq l_j \wedge i \leq j.$$

As usual, let $\leq_c^*$ denote the transitive closure of $\leq_c$, let $<_c^*$ denote the strict version of $\leq_c^*$, that is, $i <_c^* j$ iff $i \leq_c^* j$ and $i \neq j$, and let $\lessdot_c^*$ denote the covering relation $i <_c^* j$, that is, $i \lessdot_c^* j$ iff $i <_c^* j$ and $\forall k \in [n].\, k <_c^* j \Rightarrow k \leq_c^* i$.

Definition 4 *Behavioural Equivalences.*
Processes E and E' are said to be *language equivalent*, $E \sim_{lan} E'$, iff for every computation of E

$$c : E \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

there exists a computation of E'

$$c' : E' \xrightarrow[l'_1]{\sigma_1} E'_1 \ldots \xrightarrow[l'_n]{\sigma_n} E'_n$$

and vice versa.
E and E' are said to be *pomset equivalent*, $E \sim_{pom} E'$, iff the above condition for language equivalence is satisfied, and c' is further required to satisfy $i \leq_c^* j \iff i \leq_{c'}^* j$.
E and E' are said to be *location equivalent*, $E \sim_{loc} E'$, iff the above condition for language equivalence is satisfied, and c' is further required to satisfy that there exists a relation $\mathcal{R} \subseteq loc(c) \times loc(c')$ satisfying that for each $1 \leq i \leq n$, $\mathcal{R}$ restricts to a bijection on $l_i \times l'_i$, and for each $i \leq j$, $s_0(\mathcal{R} \cap l_i \times l'_i)s'_0$ and $s_1(\mathcal{R} \cap l_j \times l'_j)s'_1$, $s_0 \sqsubseteq s_1 \iff s'_0 \sqsubseteq s'_1$.

In each case, we say that c' is a match of c with respect to $\sim_{lan}$, $\sim_{pom}$ and $\sim_{loc}$, respectively. $\square$

Notice that the condition in the definition of pomset equivalence requires identical global causal relationship between the events of c and c', whereas the condition in the definition of location equivalence requires the same set of local causal relationships (up to renaming of locations). Also, notice that our notion of pomset equivalence is consistent with formal definitions from e.g. [15], and that location equivalence is a natural application of the concepts from [4] to the setting of language equivalence.

Example 5 It follows immediate from the definition that for our process language considered so far, location equivalence is included in pomset equivalence, which in turn is included in language equivalence. The standard example of processes a.0 $\|$ b.0 and a.b.0 + b.a.0 shows that the inclusion in language equivalence is strict. The different intuitions behind our two non-interleaving equivalences may be illustrated by the two processes from Example 1. Formally, the reader may verify that p_1 and p_2 are pomset equivalent but not location equivalent. Intuitively, both processes may perform actions a,b, and c in sequence, i.e. same set pomsets, but in p_1 one location is responsible for both a and c, whereas in p_2 two different locations are responsible for these actions. $\square$

4 Renaming and hiding

In this section, we investigate the role of renaming and hiding with respect to the decidability. First we show that for BPP, the decidability of both pomset and location equivalence is preserved when adding renaming and hiding. Second, we show that adding renaming or hiding to BPP_S makes location equivalence undecidable.

Theorem 6 For BPP with renaming and hiding, $\sim_{loc} = \sim_{pom} \subset \sim_{lan}$.

Proof: That $\sim_{pom}$ and $\sim_{lan}$ coincide follows from the proof of Theorem 7 below. The inclusion into $\sim_{lan}$ follows from the definition and the strictness follows from Example 5. $\square$

Theorem 7 For BPP with renaming and hiding, $\sim_{loc}$ and $\sim_{pom}$ are decidable.

Proof: It is not hard to check the following equalities:

$$(\sigma.E)[f] \sim_{pom} f(\sigma).E[f],$$

$$(\sigma.E)\backslash\backslash L \sim_{pom} \sigma.(E\backslash\backslash L), \quad \mathit{if}\ \sigma \notin L,$$

$$(\sigma.E)\backslash\backslash L \sim_{pom} \tau.(E\backslash\backslash L), \quad \textit{if } \sigma \in L,$$

$$E[f]\backslash\backslash L \sim_{pom} (E\backslash\backslash f^{-1}(L))[f], \text{ and}$$

$$(E\backslash\backslash L)[f] \sim_{pom} (E[f'])\backslash\backslash\{\mu_E^{new}\}, \quad f'(\sigma) = \begin{cases} \mu_E^{new} & \textit{if } \sigma \in L, \\ f(\sigma) & \textit{otherwise} \end{cases}$$

where μ_E^{new} is a new action not present in the process E.

Because no communication is possible, it is straightforward using such equalities to check that renaming and hiding combinators can be eliminated by pushing them inwards while renaming and hiding actions in the prefixing combinator explicitly such that the obtained process is an ordinary BPP process which is pomset and location equivalent to the orignal process. $\square$

Theorem 8 For BPP_S with renaming and hiding, $\sim_{loc} \subset \sim_{pom} \subset \sim_{lan}$.

Proof: The inclusions follow by the definition, and the strictness from simple modifications of Example 1 and 5. $\square$

We spend the rest of the section showing that location equivalence is undecidable for BPP_S with renaming or hiding. The proof is by a reduction from the halting problem for two-counter machines. A (Minsky) two-counter machine [20] consists of a finite program

$$\begin{array}{lcl} l_1 & : & com_1 \\ & \vdots & \\ l_{n-1} & : & com_{n-1} \\ l_n & : & \text{HALT} \end{array}$$

and unbounded counters c_0 and c_1. The l_is and the com_is are called labels and commands, respectively. Commands are of one of two different types: commands of type I are of the form $c_j := c_j + 1;\mathsf{goto}\ l$ (*unconditional increment*) and commands of type II are of the form $\mathsf{if}\ c_j = 0\ \mathsf{then\ goto}\ l\ \mathsf{else}\ c_j := c_j - 1;\mathsf{goto}\ l'$ (*conditional decrement*), where j is either 0 or 1, and l and l' are labels.

A two-counter machine $\mathcal{M}$ executes on a given input (contents of the counters (c_0, c_1)) (m_0, m_1) by first executing com_1, then com_2, and so forth. Stopping if and only if the HALT command is reached. $\mathcal{M}$ *halts* on input (m_0, m_1) if it reaches label l_n and hence the HALT command in finitely many steps. It is well-known that the halting problem for two-counter machines is undecidable.

Theorem 9 [20] It is undecidable whether a two-counter machine $\mathcal{M}$ halts on input $(0, 0)$. $\square$

Given a two-counter machine $\mathcal{M}$ the idea is to encode the state of $\mathcal{M}$ by a BPP_S process of the form

$$X \parallel_\Sigma (C_0 \parallel d_0^{m_0} \parallel C_1 \parallel d_1^{m_1})$$

where the variable X encodes the state of the finite-state program of $\mathcal{M}$, m_0 and m_1 are the values of the counters, and C_0 and C_1 controls in interaction with X the incrementing, decrementing and zero-testing of the counters. We exhibit two different encodings which are both weak in the sense that they allow computations which do not correspond to any execution of the encoded machine. For convenience, we work with TCSP processes and not BPP_S with renaming for starters. Later, we transform the setting appropriately.

Definition 10 *First weak encoding*
Given a two-counter machine $\mathcal{M}$ let $\Delta_{\mathcal{M}}$ be the TCSP family with leading variable X_0 given by the following definitions where k ranges over $1, \ldots, n-1$ and j ranges over $0, 1$:

$$X_0 \stackrel{\text{def}}{=} X_1 \parallel_A ((GC_0 \parallel GC_1) \parallel_B S),$$

where $A = \{i_j, z_j, d_j \mid j = 0, 1\}$ and $B = \{z_j, d_j \mid j = 0, 1\}$. If com_k is $c_j := c_j + 1;\mathsf{goto}\ l_p$ then

$$X_k \stackrel{\text{def}}{=} i_j.X_p,$$

and if com_k is $\mathsf{if}\ c_j = 0\ \mathsf{then\ goto}\ l_p\ \mathsf{else}\ c_j := c_j - 1;\ \mathsf{goto}\ l_q$ then

$$X_k \stackrel{\text{def}}{=} z_j.X_p + d_j.X_q,$$

$$X_n \stackrel{\text{def}}{=} h.0,$$

$$\begin{aligned} GC_j &\stackrel{\text{def}}{=} i_j.(GC_j \parallel C_j) + z_j.GC_j, \\ C_j &\stackrel{\text{def}}{=} d_j.0, \\ S &\stackrel{\text{def}}{=} \textstyle\sum_j z_j.S + d_j.S. \end{aligned}$$

■

Let $\mathcal{M}$ be a two-counter machine $\mathcal{M}$ and let $\Delta_{\mathcal{M}}$ be the TCSP family given by Definition 10. It is clear from the definition that for any computation

$$c : X_1 \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

of $\Delta_{\mathcal{M}}$ and for each $i \in [n]$ there is k_i, j_i, m_0^i and m_1^i such that either

$$E_i \equiv X_{k_i} \parallel_A (GC_0 \parallel C_0^{m_0} \parallel GC_1 \parallel C_1^{m_1}) \parallel_B S$$

or

$$E_i \equiv 0 \parallel_A (GC_0 \parallel C_0^{m_0} \parallel GC_1 \parallel C_1^{m_1}) \parallel_B S$$

in the latter case E_i is called a *halting state*. The computation c is a halting computation if it reaches a *halting state*. For each $i \in [n]$ and $j = 0, 1$, let $count_j(E_i) = m_j$. The computations of the encoding above always increment and decrement counters properly but they may take the zero branch eventhough the corresponding counter is not zero.

Definition 11 *Proper transitions and computations*
Let

$$c : X_1 = E_0 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

be a computation of $\Delta_{\mathcal{M}}$. For each $i \in [n]$ and $j = 0, 1$, the ith transition of c is a *proper transition* if and only if *the zero-branch is only chosen on a zero-counter*, that is, if $\sigma_i = z_j$ then $count_j(E_{i-1}) = 0$. The computation c is a *proper computation* if and only if for each $i \in [n]$ the ith transition is a proper transition. Dually, an *improper transition (computation)* is a transition (computation) that is not proper. If c is an improper computation, the ith transition is the first improper transition in c and $\sigma_i = z_j$, c is said to *cheat* on counter j at the ith transition, and furthermore, k is said to be the *witness* iff the kth transition in c is the first transition to enable a d_j still enabled in E_i. ■

The following lemma shows that the encoding is correct in the sense that the execution of $\mathcal{M}$ can be uniquely simulated by $\Delta_{\mathcal{M}}$. It is however only weakly correct in the sense that there may be computations of $\Delta_{\mathcal{M}}$ which do not correspond to executions of $\mathcal{M}$.

Lemma 12 *"Weak" correctness of simulation*

- If $\mathcal{M}$ halts on input $(0, 0)$ then $\Delta_{\mathcal{M}}$ has a unique maximal proper computation reaching a halting state.
- If $\mathcal{M}$ does not halt on input $(0, 0)$ then all proper computations of $\Delta_{\mathcal{M}}$ are prefixes of a single infinite proper computation which never reaches a halting state

Proof: Both properties are not hard to verify: each execution step of $\mathcal{M}$ is matched by a unique transitions of $\Delta_{\mathcal{M}}$. Matching for each $j = 0, 1$, a *test for zero*, an *increment* and a *decrement* of the jth counter by z_j, i_j and d_j, respectively. □

Definition 13 *Second weak encoding*
Given a two-counter machine $\mathcal{M}$ let $\Delta'_{\mathcal{M}}$ be the TCSP family with leading variable Y_0 given by the following definitions where k ranges over $1, \ldots, n-1$ and j ranges over $0, 1$:

$$Y_0 \stackrel{\text{def}}{=} Y_1 \parallel_A (GD_0 \parallel GD_1) \parallel_B T,$$

where $A = \{i_j, z_j, d_j \mid j = 0,1\}$ and $B = \{z_j, z'_j, d_j, d'_j \mid j = 0,1\}$. If com_k is $c_j := c_j+1;\textsf{goto}\ l_p$ then

$$Y_k \stackrel{\text{def}}{=} i_j.Y_p,$$

and if com_k is $\textsf{if}\ c_j = 0\ \textsf{then goto}\ l_p\ \textsf{else}\ c_j := c_j - 1;\textsf{goto}\ l_q$ then

$$Y_k \stackrel{\text{def}}{=} z_j.Y_p + d_j.Y_q + z'_j.Y^{\mathcal{H}},$$

$$Y_n \stackrel{\text{def}}{=} 0$$

and

$$\begin{array}{lcl}
GD_j & \stackrel{\text{def}}{=} & i_j.(GD_j \parallel D_j) + i_j.(GD'_j \parallel D'_j) + z_j.GD_j, \\
GD'_j & \stackrel{\text{def}}{=} & i_j.(GD'_j \parallel D_j) + z_j.GD'_j + z'_j.GD'_j. \\
 & & \\
D_j & \stackrel{\text{def}}{=} & d_j.0, \\
D'_j & \stackrel{\text{def}}{=} & d'_j.0, \\
 & & \\
T & \stackrel{\text{def}}{=} & \sum_j z_j.T + d_j.T + z'_j.T^{\mathcal{H}} \\
T^{\mathcal{H}} & \stackrel{\text{def}}{=} & \sum_j z_j.T^{\mathcal{H}} + d_j.T^{\mathcal{H}} + d'_j.T^{\mathcal{H}}.
\end{array}$$

Furthermore, for each k over $1, \ldots, n-1$ and each j over $0,1$. If com_k is $c_j := c_j + 1;\textsf{goto}\ l_p$ then

$$Y_k^{\mathcal{H}} \stackrel{\text{def}}{=} i_j.Y_p^{\mathcal{H}}$$

and if com_k is $\textsf{if}\ c_j = 0\ \textsf{then goto}\ l_p\ \textsf{else}\ c_j := c_j - 1;\textsf{goto}\ l_q$ then

$$Y_k^{\mathcal{H}} \stackrel{\text{def}}{=} z_j.Y_p^{\mathcal{H}} + d_j.Y_q^{\mathcal{H}} + d'_j.Y_q^{\mathcal{H}},$$

and

$$Y_n^{\mathcal{H}} \stackrel{\text{def}}{=} h.0.$$

■

Let f be the renaming function given by

$$f(\sigma) = \begin{cases} z_j & \textit{if } \sigma = z'_j \\ d_j & \textit{if } \sigma = d'_j \\ \sigma & \textit{otherwise} \end{cases}$$

We call a state of a computation of $\Delta'_{\mathcal{M}}$ containing a variable labeled by superscript $\mathcal{H}$ an *$\mathcal{H}$-labeled state*. The idea is that any proper computation of $\Delta_{\mathcal{M}}$ can and can only be matched by a computation of $\Delta'_{\mathcal{M}}$ using only states which are not $\mathcal{H}$-labeled. Whereas an improper computation of $\Delta_{\mathcal{M}}$ can and can only be matched by a computation of $\Delta'_{\mathcal{M}}$ using only states which are not $\mathcal{H}$-labeled up to the first improper transition and from then on using only states which are $\mathcal{H}$-labeled.

Lemma 14 $\mathcal{M}$ does not *halt* on input $(0,0)$ $\iff$ $X_0 \sim_{loc} Y_0[f]$

Proof: To see the only if direction, assume that $\mathcal{M}$ does not halt. We show that $X_0 \lesssim_{loc} Y_0[f]$ and that $Y_0[f] \lesssim_{loc} X_0$. The second case is easy. Let h be the variable-relabeling homomorphism on syntactic trees of *Proc* induced by letting $h(Y_k) = X_k$, $h(GD_j) = h(GD'_j) = GC_j$, $h(D_j) = h(D'_j) = C_j$ and $h(T) = h(T^{\mathcal{H}}) = S$. Then, it is routine to check that for each computation

$$d : F_0 \underset{l_1}{\overset{f(\sigma_1)}{\longrightarrow}} F_1 \ldots \underset{l_n}{\overset{f(\sigma_n)}{\longrightarrow}} F_n$$

of $Y_0[f]$,

$$c : h(F_0) \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} h(F_1) \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} h(F_n)$$

is a computation of X_0.

To show the first case, it suffices by Lemma 12 to consider only every non-halting proper computation and every improper computation of X_0. We split the proof into two. Let

$$c : X_1 = E_0 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

be a proper and non-halting computation of X_0, and let g be the variable-relabeling homomorphism on syntactic trees of *Proc* induced by letting $g(X_k) = Y_k$, $g(GC_j) = GD_j$, $g(C_j) = D_j$ and $g(S) = T$. Again, it is routine to check that

$$d : g(F_0) \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} g(F_1) \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} g(F_n)$$

is a computation of $Y_0[f]$. Next, let

$$c : X_1 = E_0 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

be an improper computation of X_0. Observe that it is straightforward to show by induction in the length of the computation that for any computation

$$d : F_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} F_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} F_n$$

of $Y_0[f]$ with matching transition labelling, also the counters match, that is, for each $i \in [n]$ and $j = 0, 1$, $count_j(E_i) = count_j(F_i)$. Now, assume that c is improper because it cheats on J at I with witness K. By the above the (proper) computation up to the Ith transition can be matched by $Y[f]$. Clearly, this match may be assumed to generate $CD'_J \parallel D'_J$ by the K transition, and hence the improper Ith transition of c may be match by a z'_J transition. Since a z'_J transition leads to an $\mathcal{H}$-labelled state it is easy to see that any continuation can be matched.

Conversely to see the if direction, assume that $\mathcal{M}$ does halt. We show that $X_0 \not\lesssim_{loc} Y_0[f]$. By Lemma 12, X_0 has a unique maximal proper computation reaching a halting state. We show that this computation cannot be matched by $Y_0[f]$. Assume that there were a match. It is easy to check that $Y_0[f]$ can only perform h in a $\mathcal{H}$-labelled state and that the only way to enter such a state is by jperforming a z'_j which again can only be performed after a D'_j has been generated. Moreover, since also a d'_j can only be performed in an $\mathcal{H}$-labelled state, we get that D'_j must be present in the state performing the first z'_j. Now, this is a contradiction by the fact that $f(z'_j) = z_j$ and the above observation on counters in matching computations. □

Let

$$U_L \stackrel{\mathrm{def}}{=} \sum_{\sigma \in L} \sigma . U_L .$$

The follwing lemma makes it straightforward to make the reduction to BPP$_S$ processes with renaming.

Lemma 15 Let L_1 and L_2 be the sets $\{h\}$ and $\{i_0, i_1, h\}$, respectively, s an action in $\mathcal{A}ct$,

$$X'_0 \stackrel{\mathrm{def}}{=} s.X_1 \parallel_\Sigma (s.(GC_0 \parallel GC_1 \parallel U_{L_1}) \parallel_\Sigma s.(S \parallel U_{L_2})), \text{ and}$$

$$Y'_0 \stackrel{\mathrm{def}}{=} s.Y_1 \parallel_\Sigma (s.(GD_0 \parallel GD_1 \parallel U_{L_1}) \parallel_\Sigma s.(T \parallel U_{L_2})).$$

Then,

$$X_0 \sim_{loc} Y_0[f] \iff X'_0 \sim_{loc} Y'_0[f].$$

Proof: Straightforward because the computations are the same except from the involvement of every component in every transition and because the new locations observed are exactly the same in both processes. □

Theorem 16 For BPP$_S$ with renaming, $\sim_{loc}$ and $\sim_{pom}$ are undecidable.

Proof: Immediate consequence of Lemma 15 and 14. □

Lemma 17 Let L be the set of actions $\{d_j, d'_j, z_j, z'_j \mid j = 0, 1\}$.

$$\mathcal{M} \text{ does not } halt \text{ on input } (0,0) \iff X_0 \backslash\backslash L \sim_{loc} Y_0 \backslash\backslash L$$

Proof: The proof is a routine adaption of the Lemma 14. □

Lemma 18 Let L, L_1 and L_2 be the sets $\{d_j, d'_j, z_j, z'_j \mid j = 0, 1\}$, $\{h\}$ and $\{i_0, i_1, h\}$, respectively, and let

$$X'_0 \stackrel{\text{def}}{=} s.X_1 \parallel_\Sigma (s.(GC_0 \parallel GC_1 \parallel U_{L_1}) \parallel_\Sigma s.(S \parallel U_{L_2})), \text{ and}$$

$$Y'_0 \stackrel{\text{def}}{=} s.Y_1 \parallel_\Sigma (s.(GD_0 \parallel GD_1 \parallel U_{L_1}) \parallel_\Sigma s.(T \parallel U_{L_2})).$$

Then,

$$X_0 \backslash\backslash L \sim_{loc} Y_0 \backslash\backslash L \iff X'_0 \backslash\backslash L \sim_{loc} Y'_0 \backslash\backslash L$$

Proof: Straightforward. □

Theorem 19 For BPP_S with hiding, $\sim_{loc}$ and $\sim_{pom}$ are undecidable.

Proof: Immediate consequence of Lemma 17 and 18. □

5 Weak language, pomset, and location equivalence

All the undecidability results for the strong case extend immediately to the weak case. In the following we show that for BPP^τ the decidability results for pomset and location equivalence extent to the weak case.

With each computation c we associate a partial function $v^c : \mathbb{N} \hookrightarrow \mathbb{N}$ yielding on i the index of the ith visible action in c if it exists and undefined otherwise, and the function $\parallel c \parallel$ yielding the number of occurrences of visible actions in c.

Definition 20 Processes E and E' are said to be *weak language preordered*, $E \lesssim_{lan} E'$, iff for every computation of E

$$c : E \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

there exists a computation of E'

$$c' : E' \overset{\sigma'_1}{\underset{l'_1}{\longrightarrow}} E'_1 \ldots \overset{\sigma'_m}{\underset{l'_m}{\longrightarrow}} E'_m$$

such that there $\| c \| = \| c' \|$, for each $i \in [\| c \|]$, $\sigma_{v^c(i)} = \sigma'_{v^{c'}(i)}$.
E and E' are said to be *weak pomset preordered*, $E \lesssim_{pom} E'$, iff c' is further required to satisfy that for each $i, j \in [\| c \|]$, $v^c(i) \leq_c^* v^c(j) \iff v^{c'}(i) \leq_{c'}^* v^{c'}(j)$.
E and E' are said to be *weak location preordered*, $E \lesssim_{loc} E'$, iff c' is further required to satisfy that there exists a relation $\mathcal{R} \subseteq loc(c) \times loc(c')$ satisfying that for each $1 \leq i \leq \| c \|$, $\mathcal{R}$ restricts to a bijection on $l_i \times l'_i$, and for each $i, j \in [\| c \|]$ such that $i \leq j$, $s_0(\mathcal{R} \cap l_i \times l'_i)s'_0$ and $s_1(\mathcal{R} \cap l_j \times l'_j)s'_1$, $s_0 \sqsubseteq s_1 \iff s'_0 \sqsubseteq s'_1$. In each case, we say that c' is a match of c with respect to $\approx_{lan}$, $\approx_{pom}$ and $\approx_{loc}$, respectively.

Moreover, E and E' are said to be *weak language equivalent*, $E \approx_{lan} E'$, iff $E \lesssim_{lan} E'$ and $E' \lesssim_{lan} E$. E and E' are said to be *weak pomset equivalent*, $E \approx_{pom} E'$, if and only if $E \lesssim_{pom} E'$ and $E' \lesssim_{pom} E$. E and E' are said to be *weak location equivalent*, $E \approx_{loc} E'$, if and only if $E \lesssim_{loc} E'$ and $E' \lesssim_{loc} E$. ■

We write $E \overset{\epsilon}{\Longrightarrow} E'$, if $E = E'$, or if there exists a computation

$$E \overset{\tau}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\tau}{\underset{l_n}{\longrightarrow}} E_n = E',$$

from E to E' with only τ transition, $E \overset{\sigma}{\underset{l}{\Longrightarrow}} E'$, if there exists processes E_1 and E_2 such that

$$E \overset{\epsilon}{\Longrightarrow} E_1 \overset{\sigma}{\underset{l}{\longrightarrow}} E_2 \overset{\epsilon}{\Longrightarrow} E',$$

and $E \overset{\sigma}{\longrightarrow}\overset{\epsilon}{\Longrightarrow} E'$, if there exists a process E'' such that

$$E \overset{\sigma}{\longrightarrow} E'' \overset{\epsilon}{\Longrightarrow} E'.$$

Example 21 Consider the process

$$p_3 = (a.b.0 \parallel \bar{b}.c.0)\backslash\{b\}$$

The following is an example of an associated computation (representing the unique maximal run)

$$c : p_3 \underset{\{0\}}{\overset{a}{\longrightarrow}} (b.0 \parallel \bar{b}.c.0)\backslash\{b\} \underset{\{0,1\}}{\overset{\tau}{\longrightarrow}} (0 \parallel c.0)\backslash\{b\} \underset{\{1\}}{\overset{c}{\longrightarrow}} (0 \parallel 0)\backslash\{b\}.$$

Consider alternatively the process

$$p_4 = \tau.a.\tau.c.0$$

with computation

$$d : p_4 \underset{\{\epsilon\}}{\overset{\tau}{\longrightarrow}} a.\tau.c.0 \underset{\{\epsilon\}}{\overset{a}{\longrightarrow}} \tau.c.0 \underset{\{\epsilon\}}{\overset{\tau}{\longrightarrow}} c.0 \underset{\{\epsilon\}}{\overset{c}{\longrightarrow}} 0.$$

The computation d is a match of c with respect to $\approx_{lan}$ and $\approx_{pom}$ but not with respect to $\approx_{loc}$. ■

6 BPP$^\tau$

In the strong case, τ is just another action no different than the others. Hence, all results of decidablility on BPP transfers to BPP$^\tau$.

Theorem 22 For BPP$^\tau$, $\approx_{loc} = \approx_{pom} \subset \approx_{lan}$.

Proof: That $\approx_{pom}$ and $\approx_{lan}$ coincide is an easy consequence of the definition because no communication is possible. The inclusion into $\approx_{lan}$ follows immediate from the definition and the strictness follows from Example 5. □

It is easy to see that a Milner guarded BPP$^\tau$ family Δ can effectively be transformed into a guarded BPP family Δ' by systematically getting rid of $\tau.E$ subexpressions by appropriate substitutions of E in such a way that $\Delta \approx_{pom} \Delta'$ ($\Delta \approx_{loc} \Delta'$).

For the general case, this elimination procedure cannot be applied due to the possibility of τ cycles. Instead, we introduce a closure operation performing combined elimination and bounded saturation of τs as explained below.

To appriciate the difference in the underlying "τ-structure" of computations which correspond to matching computations with respect to $\approx_{pom}$, consider the pomsets in Figure 1, as pomsets they clearly do not match but as weak pomset they match. The pomset to the left (right) is a pomset of Δ_1 (Δ_2) in Example 23 below. In fact, $\Delta_1 \approx_{pom} \Delta_2$ but clearly $\Delta_1 \not\sim_{pom} \Delta_2$. Moreover, Δ_1 and Δ_2 can match any BPP$^\tau$ computation over a and b with respect to $\approx_{pom}$.

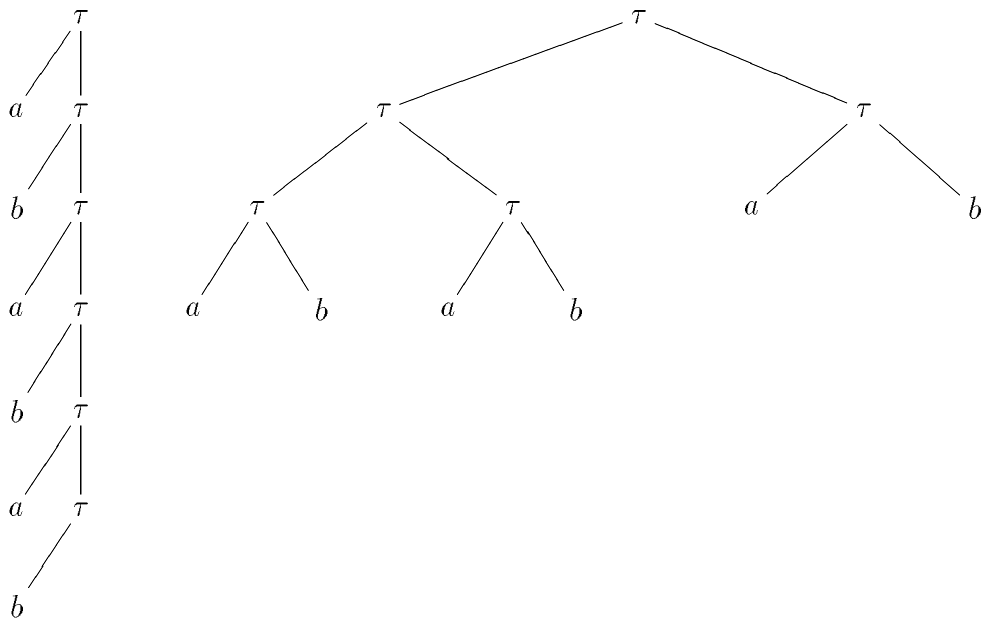

Figure 1: Distinct strong pomsets whose weak versions match.

Example 23

$$\Delta_1 = \{ \quad X_1 \stackrel{\text{def}}{=} \tau.\{X_2, X_3\}, \quad X_2 \stackrel{\text{def}}{=} a.\{X_1\}, \quad X_3 \stackrel{\text{def}}{=} \tau.\{X_1, X_4\}, \quad X_4 \stackrel{\text{def}}{=} b.\{X_1\} \quad \}$$

$$\Delta_2 = \{ \quad Y_1 \stackrel{\text{def}}{=} \tau.\{Y_1, Y_1\} + \tau.\{Y_2, Y_3\}, \quad Y_2 \stackrel{\text{def}}{=} a.\{Y_1\}, \quad Y_3 \stackrel{\text{def}}{=} b.\{Y_1\} \}$$

■

In the following, we show how to reduce the decidability of $\lessapprox_{pom}$ to the decidability of $\lesssim_{pom}$. The main observation used is that whereas a BPP$^{\tau}$ family may allow computations of arbitrary large branching degrees when restricting to visible transitions, the underlying dependency ordering is a tree of a branching degree uniformly bounded over all computations of the family. Hence, if Δ and Δ' are BPP$^{\tau}$ families then based on the branching bound B of Δ, we seek to effectively compute a family $\mathcal{S}_B(\Delta')$ such that $\Delta \lessapprox_{pom} \Delta'$ iff $\Delta \lesssim_{pom} \mathcal{S}_B(\Delta')$. Intuitively, $\mathcal{S}_B(\Delta')$ will do all computations whose underlying dependency ordering is a tree with branching bounded by B and which can be match by a computation of Δ' with respect to $\approx_{pom}$. The first step is to compute for each variable X a finite representation of the set of all states reachable from X by doing any number of τ actions, that is, the set

$$\{\beta \mid X \stackrel{\epsilon}{\Longrightarrow} \beta\}$$

and for each action $\sigma \in \mathcal{A}ct(\Delta) \cup \{\tau\}$ a finite representation of the set of all states reachable from X by doing a σ action and then doing any number of τ actions, that is,

$$\{\beta \mid X \stackrel{\sigma}{\longrightarrow} \stackrel{\epsilon}{\Longrightarrow} \beta\}.$$

For this purpose, it is convenient to assume that the families are of the following form.

Definition 24 A BPP$^\tau$ family $\Delta = \{X_i \stackrel{\mathrm{def}}{=} E_i \mid i = 1, 2 \ldots, n\}$ is in *subset closed normal form* if and only if it is in normal form and furthermore for each expression

$$E_i \equiv \sum_{j=1}^{n_i} \sigma_{ij} \alpha_{ij}$$

the set $\Gamma_{ij} = \{\alpha_{ij} \mid j \in [n_i]\}$ is subset closed, that is, for every $\alpha_1, \alpha_2 \in \mathit{Var}(\Delta)^{\otimes}$ such that $\alpha_1 \subseteq \alpha_2$, if $\alpha_2 \in \Gamma_{ij}$ then $\alpha_1 \in \Gamma_{ij}$. ■

As shown next, we can safely restrict ourselves to families in subset closed normal form in the following.

Proposition 25 Let Δ be a BPP$^\tau$ family with leading variable X_1. Then a BPP$^\tau$ family in subset closed normal form Δ' can be *effectively* constructed such that $\Delta'' \sim_{pom} \Delta'$, where Δ'' is Δ extended with a new leading variable $X_1' = s.X_1$, for some $s \in \mathcal{A}ct$ and $X_1' \notin \mathit{Var}(\Delta)$.

Proof: Straightforward extension of the normal form result in [26]. □

Note that for example the process $(a.0 \parallel b.0) + c.0$ can not be brought on normal form while preserving pomset equivalence whereas the process $s.((a.0 \parallel b.0) + c.0)$ can. Hence, the point of the slightly technical normal form result is that prefixing the leading equation of two BPP processes by the same action respects and reflects pomset equivalence.

We base the computation of the set reachable states discussed above on a "weak" version of the standard Karp-Miller tree, see *e.g.* [23, 24]. For this purpose, we need a bunch of fairly standard definitions.

Definition 26 The set $I\!N \cup \{\omega\}$ of the natural numbers $I\!N$ extended with a special (limit) symbol ω is denoted by $I\!N_\omega$. As usual, the operations $+$ and $-$, and the relation $\leq$ over $I\!N$ are extended to $I\!N_\omega$ by stipulating that for all $n \in I\!N$, $\omega + \omega = \omega + n = n + \omega = \omega$ and $n \leq \omega$. The set of all n-tuples over $I\!N$ ($I\!N_\omega$) is denoted $I\!N^n$ ($I\!N^n_\omega$), elements of $I\!N$ ($I\!N^n_\omega$) are denoted $\bar{m}$, and the components are for each $i \in [n]$ denoted by $\bar{m}_i$. For each $i \in [n]$, we denote by $\bar{e}_i$ the ith *unit n-tuple*, that is, the n-tuple with 1 in the ith entry and 0 in all other entries. For any $k \in I\!N$, the n-tuple with k in all entries is denoted by $\bar{k}$. Operations and relations on $I\!N_\omega$ extend componentwise to $I\!N^n_\omega$. The *downwards closure* of a subset $M \subseteq I\!N^n_\omega$ is the set $\widehat{M} = \{\bar{m} \in I\!N^n \mid \exists \bar{m}' \in M.\, \bar{m} \leq \bar{m}'\}$. Let α a finite multiset over a finite set of variables $\{X_1, \ldots, X_n\}$, then we denote by $\bar{m}(\alpha)$ the n-tuple over $I\!N$ defined by taking the ith entry to be the number of copies of the variable X_i in α. For any BPP$^\tau$ family Δ, let $|\rangle_\Delta \subseteq I\!N^n_\omega \times (\mathcal{A}ct(\Delta) \cup \{\tau\}) \times I\!N^n_\omega$ be the relation defined by for each $\bar{m}, \bar{m}' \in I\!N^n_\omega$ and $\sigma \in \mathcal{A}ct(\Delta) \cup \{\tau\}$, $(\bar{m}, \sigma, \bar{m}') \in |\rangle_\Delta$ iff there exists $X_i \in Var(\Delta)$ and $\alpha \in Var(\Delta)^\otimes$ such that $\bar{m} \geq \bar{e}_i$, $X \xrightarrow{\sigma} \alpha$ and $(\bar{m} - \bar{e}_i) + \bar{m}(\alpha) = \bar{m}'$. Often, we use the more convenient infix notionation $\bar{m}|\sigma\rangle_\Delta \bar{m}'$ instead of $(\bar{m}, \sigma, \bar{m}') \in |\rangle_\Delta$ and whenever Δ is clear from the context we drop the Δ subscript. ■

Keeping in mind that BPP$^\tau$ processes in normal form may be viewed as communication-free nets, the *weak* Karp-Miller tree defined next are essentially the Karp-Miller trees on nets but restricted to τ actions.

Definition 27 Let Δ be a BPP$^\tau$ family in normal form. The *weak Karp-Miller tree* associated with the process $\alpha \subseteq Var(\Delta)^\otimes$ is a node labelled tree T_α in which each node is labelled with an n-tuple over $I\!N_\omega$. The tree T_α is inductively defined as follows

(i) the root of T_α is labelled by $\bar{m}(\alpha)$, and

whenever v is a node in T_α labelled $\bar{m}$, then

(ii) if $\bar{m}$ is also the label of an ancestor of v then v has no sons, and

(iii) otherwise, for each $\bar{m}' \in I\!N^n_\omega$ such that $t = \bar{m}|\tau\rangle\bar{m}'$, v has a son v^t with label $\bar{m}^t$ where

(a) if there is an ancestor v'' of v with label $\bar{m}''$ such that $\bar{m}'' \leq \bar{m}'$ then for each $i \in [n]$,

$$\bar{m}^t_i = \begin{cases} \omega, & \textit{if } \bar{m}''_i < \bar{m}'_i, \\ \bar{m}'_i, & \textit{else} \end{cases}$$

(b) otherwise, $\bar{m}^t = \bar{m}'$.

For each $\sigma \in \mathcal{A}ct(\Delta) \cup \{\tau\}$, the *$\sigma$-weak Karp-Miller tree* associated with the process $\alpha \subseteq Var(\Delta)^\otimes$ is a node labelled tree T^σ_α in which each node is labelled with an n-tuple over $I\!N_\omega$ such that *(i)* the root of T^σ_α is labelled by $\bar{m}(\alpha)$, and

(ii) for each $\beta \subseteq \mathit{Var}(\Delta)^{\otimes}$ such that $\alpha \xrightarrow{\sigma} \beta_t$, the root has the weak Karp-Miller tree T_{β_t} as a subtree.

The set of all labels in T_α and T^σ_α is denoted by $\mathcal{E}(\alpha)$ and $\mathcal{E}^\sigma(\alpha)$, respectively. Moreover,

$$\mathcal{E}(\Delta) = \bigcup_{X \in \mathit{Var}(\Delta)} \mathcal{E}(X) \cup \bigcup_{\sigma \in \mathcal{A}ct(\Delta) \cup \{\tau\}} \mathcal{E}^\sigma(X)$$

■

Lemma 28 Let Δ be a BPP$^\tau$ family in normal form. Then for any process $\alpha \subseteq \mathit{Var}(\Delta)^{\otimes}$, the *weak Karp-Miller tree* T_α *is finite and effectively constructible, and moreover, for any* $\sigma \in \mathcal{A}ct(\Delta) \cup \{\tau\}$*, the* σ*-weak Karp-Miller tree* T^σ_α *is finite and effectively constructible. In particular, the set* $\mathcal{E}(\alpha)$ *of all labels in* T_α*, the set* $\mathcal{E}^\sigma(\alpha)$*, of all labels in* T^σ_α*, and the set* $\mathcal{E}(\Delta)$ *are finite and effectively computable.*

Proof: Standard Karp-Miller trees for vector addition systems and Petri nets are well-known to be finite and effectively constructible, see *e.g.* [23, 24], and it is routine to transfer the proof to (σ-) weak Karp-Miller trees. The rest follows easily. □

Lemma 29 Let Δ be a BPP$^\tau$ family in subset closed normal form. Then,

$$\widehat{\mathcal{E}(X)} = \{\bar{m}(\beta) \mid X \stackrel{\epsilon}{\Longrightarrow} \beta\}, \text{ and}$$

$$\widehat{\mathcal{E}^\sigma(X)} = \{\bar{m}(\beta) \mid X \xrightarrow{\sigma} \stackrel{\epsilon}{\Longrightarrow} \beta\}.$$

Proof: Straightforward from the construction of the (σ-) weak Karp-Miller tree and downwards closure ensure by the subset closedness. □

Definition 30 Let BPP$^\tau$ be a family $\Delta = \{X_i \stackrel{\mathrm{def}}{=} E_i \mid i = 1,2\ldots,n\}$ in subset closed normal form such that for each $i \in [n]$, $E_i \equiv \sum_{j=1}^{n_i} \sigma_{ij}\alpha_{ij}$ and let $B \in \mathbb{N}$. Define the *τ-saturation of Δ up-to branching bound B*,

$$\mathcal{S}_B(\Delta) = \{Y_i \stackrel{\mathrm{def}}{=} F_i \mid i \in [n]\} \cup \{Z^\sigma_{\bar{m}} \stackrel{\mathrm{def}}{=} G^\sigma_{\bar{m}} \mid \bar{m} \in \mathcal{E}(\Delta) \wedge \sigma \in \mathcal{A}ct(\Delta) \cup \{\tau\}\},$$

where for each $\bar{m} \in \mathcal{E}(\Delta)$, the set $\mathcal{C}_{\bar{m}}$ consists of all submultisets γ for which there exist subsets $\bar{m}_1,\ldots,\bar{m}_k \in \mathcal{E}(\Delta)$, $\alpha \subseteq Var(\Delta)^\otimes$, such that

$$\gamma = \{|Z^\tau_{\bar{m}_1},\ldots,Z^\tau_{\bar{m}_k}|\} \cup \alpha \ \wedge\ |\gamma| \leq B \ \wedge\ \sum_{1\leq i\leq k} \bar{m}_i + \bar{m}(\alpha) \leq \bar{m},$$

and

$$G^\sigma_{\bar{m}} \equiv \sum_{\beta\in\mathcal{C}_{\bar{m}}} \sigma.\beta,$$

and for each $i \in [n]$,

$$F'_i \equiv \sum_{j=1}^{n_i} \sum_{\bar{m}\in\mathcal{E}^{\sigma_{ij}}(X_i)} Z^{\sigma_{ij}}_{\bar{m}} + \sum_{\bar{m}\in\mathcal{E}(X_i)} Z^\tau_{\bar{m}}, \text{ and} \tag{1}$$

$$F_i \equiv F'_i + \sum_{k\in[n],\, k\neq i,\, \bar{e}_k\in\mathcal{E}(X_i)} F'_k. \tag{2}$$

■

Lemma 31 For any BPP$^\tau$ family Δ in subset closed normal form and any natural number B

$$\Delta \approx_{pom} \mathcal{S}_B(\Delta), \text{ in fact, } \Delta \lesssim_{pom} \mathcal{S}_B(\Delta) \text{ and } \mathcal{S}_B(\Delta) \lessapprox_{pom} \Delta.$$

Proof: Straightforward from Definition 30. □

Lemma 32 Let Δ_1 and Δ_2 be BPP$^\tau$ families in subset closed normal form such that B is the branching bound of Δ_1. Then,

$$\Delta_1 \lessapprox_{pom} \Delta_2 \iff \Delta_1 \lesssim_{pom} \mathcal{S}_B(\Delta_2).$$

Proof: The if direction follows from Lemma 31, because

$$\Delta_1 \lesssim_{pom} \mathcal{S}_B(\Delta_2) \approx_{pom} \Delta_2.$$

To see the only if direction assume that $\Delta_1 \lessapprox_{pom} \Delta_2$. We show a slightly stronger result. For any $X \in Var(\Delta)$ and any computation

$$c : X \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \dots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

of Δ_1, if there is a computation

$$d' : Y \underset{l'_1}{\overset{\sigma'_1}{\longrightarrow}} F_1 \dots \underset{l'_m}{\overset{\sigma'_m}{\longrightarrow}} F_m$$

such that d' is a match of c with respect to $\approx_{pom}$ then there is a computation

$$d'' : Y \underset{l''_1}{\overset{\sigma_1}{\longrightarrow}} F''_1 \dots \underset{l''_n}{\overset{\sigma''_n}{\longrightarrow}} F''_n$$

such that d'' is a match of c with respect to $\sim_{pom}$. We proceed by induction in the length n of the computation c, The base case ($n = 1$) is obvious. In the step ($n > 1$) there are three cases $\sigma_1 = \sigma'_1$, $\sigma_1 = \tau \wedge \sigma'_1 \neq \tau$, and $\sigma_1 \neq \tau \wedge \sigma'_1 = \tau$ each of which follows by induction and use of, respectively, the first and second clause in the sum (1) and the second clause in the sum (2) of Definition 30. □

Theorem 33 For BPP$^\tau$, $\approx_{pom}$ and $\approx_{loc}$ are decidable.

Proof: The decidability of weak pomset equivalence follows from Lemma 32 and the decidability of BPP$^\tau$ in the strong case. The decidability of weak location equivalence follows since by Theorem 22 $\approx_{pom}$ and $\approx_{loc}$ coincide on BPP$^\tau$. □

7 BPP$_M$

We settled the strong case in [26] by showing that pomset and location equivalence conicide and remain decidable for BPP$_M$. In the weak setting, $\approx_{pom}$ and $\approx_{loc}$ still conicide for BPP$^\tau$ but when moving to BPP$_M$ this changes. In fact, they become incomparable which contrast the strong case where location equivalence is always finer then pomset. We have borrowed the following example from Kiehn [16] to show this.

Example 34 For the BPP$_M$ processes

$$r_1 = a.b.0 \parallel \bar{b}.c.0 + a.c.0 \text{ and } r_2 = a.b.0 \parallel \bar{b}.c.0,$$

we have that $r_1 \approx_{pom} r_2$ and $r_1 \not\approx_{loc} r_2$, and for BPP$_M$ processes

$$s_1 = a.b.0 \parallel \bar{b}.c.0 + c.b.0 \parallel \bar{b}.a.0 + a.0 \| c.0 \text{ and } s_2 = a.b.0 \parallel \bar{b}.c.0 + c.b.0 \parallel \bar{b}.a.0,$$

we have that $s_1 \approx_{loc} s_2$ and $s_1 \not\approx_{pom} s_2$. ■

Theorem 35 For BPP_M, $\approx_{loc}$ and $\approx_{pom}$ incomparable and both strictly finer than $\approx_{lan}$.

Proof: Immediate from the definition and Example 34. $\square$

7.1 Weak location equivalence

In this section, we settle the weak case by showing that $\lessapprox_{loc}$ and $\approx_{loc}$ are undecidable for BPP_M. Again, the proof is by a reduction from the halting problem for two-counter machines. The encodings used are considerably weaker than those used for BPP_S with renaming in the sense that those encodings could only cheat on zero testing whereas these can additionally cheat on decrement leaving only increment behaving properly.

Clearly, the decidability of $\lessapprox_{loc}$ would imply decidability of $\approx_{loc}$. The following lemma observes that in fact $\lessapprox_{loc}$ and $\approx_{loc}$ are equivalent with respect to decidability.

Lemma 36 Let E and F be BPP_M processes.

$$E \lessapprox_{loc} F \Leftrightarrow E + F \approx_{loc} F$$

Proof: Immediate from the definition. $\square$

We spend the rest of this section showing that $\lessapprox_{loc}$ and hence $\approx_{loc}$ are undecidable for BPP_M. The proof is by a reduction from the Halting problem for Minsky two-counter machines to the $\lessapprox_{loc}$ problem using weak encodings of two-counter machines into BPP_M processes.

Given a two-counter machine $\mathcal{M}$ the idea is to encode the state of $\mathcal{M}$ by a BPP process of the form

$$X \parallel C_0^{m_0} \parallel C_1^{m_1}$$

where the variable X encodes the state of the finite-state program of $\mathcal{M}$ and m_0 and m_1 are the values of the counters.

Definition 37 *First weak encoding*
Given a two-counter machine $\mathcal{M}$ let $\Delta_{\mathcal{M}}$ be the BPP_M family with leading variable X_1 given by the following definitions where k ranges over $1, \ldots, n-1$ and j ranges over $0, 1$. If com_k is $c_j := c_j + 1;\mathsf{goto}\ l_p$ then

$$X_k \stackrel{\text{def}}{=} i_j.(X_p \parallel C_j),$$

and if com_k is $\mathsf{if}\ c_j = 0\ \mathsf{then}\ \mathsf{goto}\ l_p\ \mathsf{else}\ c_j := c_j - 1;\mathsf{goto}\ l_q$ then

$$X_k \stackrel{\text{def}}{=} z_j.X_p + g_j.P_{kj}$$

and

$$P_{kj} \stackrel{\text{def}}{=} g'_j.X_q,$$

$$X_n \stackrel{\mathrm{def}}{=} h.0$$

and

$$C_j \stackrel{\mathrm{def}}{=} d_j.0$$

■

Let $\mathcal{M}$ be a two-counter machine $\mathcal{M}$ and let $\Delta_{\mathcal{M}}$ be the BPP^{τ}_{M} family given by Definition 37. It is clear from the definition that for any computation

$$c : X_1 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of $\Delta_{\mathcal{M}}$, for each $i \in [n]$ there is $k \in [n], j \in \{0,1\}$, and $m_0, m_1 \in I\!N$ such that either

$$E_i \equiv X_k \parallel C_0^{m_0} \parallel C_1^{m_1}, \qquad E_i \equiv P_{kj} \parallel C_0^{m_0} \parallel C_1^{m_1}, \qquad \textit{or } E_i \equiv C_0^{m_0} \parallel C_1^{m_1}$$

in the latter case E_i is called a *halting state*. The computation c is a *halting computation* if it reaches a halting state. For each $i \in [n]$ and $j = 0, 1$, let $count_j(E_i) = m_j$. The computations of the encoding above always increment counters properly but they may take the zero branch eventhough the corresponding counter is not zero and they may decrement a non-zero counter at any point.

Definition 38 *Proper transitions and computations*
Let

$$c : X_1 = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

be a computation of $\Delta_{\mathcal{M}}$. For each $i \in [n]$ and $j = 0, 1$, the ith transition of c is a *proper transition* if and only if

1. (*The zero-branch is only chosen on a zero-counter*)
   If $\sigma_i = z_j$ then $count_j(E_{i-1}) = 0$.
2. (*A decrement is performed: $g_j d_j g'_j$*)
   (a) If $\sigma_i = d_j$ then $i > 1$ and $\sigma_{i-1} = g_j$, and
   (b) if $\sigma_i = g'_j$ then $i > 1$ and $\sigma_{i-1} = d_j$.

The computation c is a *proper computation* if and only if for each $i \in [n]$ the ith transition is a proper transition. ■

Lemma 39 *"Weak" correctness of simulation*

- If $\mathcal{M}$ halts on input $(0,0)$ then $\Delta_{\mathcal{M}}$ has a unique maximal proper computation reaching for some $m_0, m_1 \in \mathbb{N}$ a halting state.
- If $\mathcal{M}$ does not halt on input $(0,0)$ then all proper computations of $\Delta_{\mathcal{M}}$ are prefixes of a single infinite proper computation which never reaches a halting state

Proof: Both properties are not hard to verify: each execution step of $\mathcal{M}$ is matched by a unique sequence of one or more transitions of $\Delta_{\mathcal{M}}$. Matching for each $j = 0, 1$, a *test for zero*, an *increment* and a *decrement* of the jth counter by z_j, i_j and $g_j d_j g'_j$, respectively. □

The second encoding is more complicated than the first. Its task is to match any computation of Δ_M except for a possible proper halting computation.

Definition 40 *Second weak encoding*
Given a two-counter machine $\mathcal{M}$ let $\Delta'_{\mathcal{M}}$ be the BPP$_M$ family with leading variable Y_1 given by the following definitions where k ranges over $1, \ldots, n-1$ and j over $0, 1$. If com_k is $c_j := c_j + 1;\mathsf{goto}\ l_p$ then

$$\begin{aligned} Y_k &\stackrel{\text{def}}{=} i_j.(Y_p \parallel D_j) + \\ &\quad t_j.t'_j.Y_k^{\mathcal{H}} + t_{j\oplus1}.t'_{j\oplus1}.Y_k^{\mathcal{H}} \ \textit{(wrong decrement of a counter)} \end{aligned}$$

and if com_k is $\mathsf{if}\ c_j = 0\ \mathsf{then\ goto}\ l_p\ \mathsf{else}\ c_j := c_j - 1;\mathsf{goto}\ l_q$ then

$$\begin{aligned} Y_k &\stackrel{\text{def}}{=} z_j.Y_p + g_j.Q_{kj} + \\ &\quad s_j.z_j.Y_p^{\mathcal{H}} + \ \textit{(choosing zero-branch on non-zero counter)} \\ &\quad t_j.t'_j.Y_k^{\mathcal{H}} + t_{j\oplus1}.t'_{j\oplus1}.Y_k^{\mathcal{H}}, \ \textit{(wrong decrement of a counter)} \\ Q_{kj} &\stackrel{\text{def}}{=} t_j.t'_j.R_{kj} + \\ &\quad g'_j.Y_q^{\mathcal{H}} + \ \textit{(leaving out decrement)} \\ &\quad t_{j\oplus1}.t'_{j\oplus1}.Q_{kj}^{\mathcal{H}}, \ \textit{(decrement of the wrong counter)} \\ R_{kj} &\stackrel{\text{def}}{=} g'_j.Y_q + \\ &\quad t_j.t'_j.R_{kj}^{\mathcal{H}} + t_{j\oplus1}.t'_{j\oplus1}.Q_{kj}^{\mathcal{H}}, \ \textit{(wrong decrement of a counter)} \end{aligned}$$

$$Y_n \stackrel{\text{def}}{=} 0$$

and

$$D_j \stackrel{\text{def}}{=} \bar{t}_j.d_j.\bar{t}'_j.0 + \bar{s}_j.D_j.$$

Furthermore, If com_k is $c_j := c_j + 1;\mathsf{goto}\ l_p$ then

$$Y_k^{\mathcal{H}} \stackrel{\text{def}}{=} i_j.(Y_p^{\mathcal{H}} \parallel D_j) + t_j.t'_j.Y_k^{\mathcal{H}} + t_{j\oplus1}.t'_{j\oplus1}.Y_k^{\mathcal{H}}$$

and if com_k is $\mathsf{if}\ c_j = 0\ \mathsf{then\ goto}\ l_p\ \mathsf{else}\ c_j := c_j - 1;\mathsf{goto}\ l_q$ then

$$\begin{array}{rcl} Y_k^{\mathcal{H}} & \stackrel{\text{def}}{=} & z_j.Y_p^{\mathcal{H}} + g_j.Q_{kj}^{\mathcal{H}} + s_j.z_j.Y_p^{\mathcal{H}} + t_j.t'_j.Y_k^{\mathcal{H}} + t_{j\oplus 1}.t'_{j\oplus 1}.Y_k^{\mathcal{H}}, \\ Q_{kj}^{\mathcal{H}} & \stackrel{\text{def}}{=} & t_j.t'_j.R_{kj}^{\mathcal{H}} + g'_j.Y_q^{\mathcal{H}} + t_{j\oplus 1}.t'_{j\oplus 1}.Q_{kj}^{\mathcal{H}}, \\ R_{kj}^{\mathcal{H}} & \stackrel{\text{def}}{=} & g'_j.Y_q^{\mathcal{H}} + t_j.t'_j.R_{kj}^{\mathcal{H}} + t_{j\oplus 1}.t'_{j\oplus 1}.Q_{kj}^{\mathcal{H}}, \end{array}$$

$$Y_n^{\mathcal{H}} \stackrel{\text{def}}{=} h.0$$

■

We call a state of a computation of $\Delta'_{\mathcal{M}}$ containing a variable labelled by superscript $\mathcal{H}$ an *$\mathcal{H}$-labelled state*. The idea is that any proper computation of $\Delta_{\mathcal{M}}$ can and can only be matched by a computation of $\Delta'_{\mathcal{M}}$ using only states which are not $\mathcal{H}$-labelled. Whereas an improper computation of $\Delta_{\mathcal{M}}$ can and can only be matched by a computation of $\Delta'_{\mathcal{M}}$ using only states which are not $\mathcal{H}$-labelled up to the first improper transition and from then on using only states which are $\mathcal{H}$-labelled.

Also for the second encoding, it is clear from the definition that for any computation

$$d : Y_1 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} F_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} F_n$$

of $\Delta'_{\mathcal{M}}$, for each $i \in [n]$ there is $k \in [n], j \in \{0,1\}$, and $m_0, m_1 \in I\!N$ such that F_i is of the form

$$F_i \equiv \mathcal{Z} \parallel D_0^{m_0} \parallel D_1^{m_1},\ or\ F_i \equiv D_0^{m_0} \parallel D_1^{m_1}$$

where $\mathcal{Z}$ is either Y_k, Q_{kj}, R_{kj}, $Y_k^{\mathcal{H}}$, $Q_{kj}^{\mathcal{H}}$, or $R_{kj}^{\mathcal{H}}$. For each $i \in [n]$ and $j = 0, 1$, let $count_j(F_i) = m_j$ and in the first case, let $control(F_i) = \mathcal{Z}$. In fact, there is a close relationship between the states of the encodings. The following definition gives a way of mapping states of Δ_M to states of Δ'_M which will be useful in the next lemma.

Definition 41 For each computation

$$c : X_1 = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of $\Delta_{\mathcal{M}}$ for each $i \in [n]$, let f_i^c be the variable-relabelling homomorphism on syntactic trees of *Proc* induced by letting $f_i^c(C_j) = D_j$, and whenever the computation from E_0 to E_i is a proper computation, $f_i^c(X_k) = Y_k$ and

$$f_i^c(P_{kj}) = \begin{cases} R_{kj} & if\ i > 0\ and\ \sigma_i = d_j \\ Q_{kj} & otherwise \end{cases}$$

and whenever the computation from E_0 to E_i is not a proper computation, $f_i^c(X_k^{\mathcal{H}}) = Y_k^{\mathcal{H}}$ and

$$f_i^c(P_{kj}^{\mathcal{H}}) = \begin{cases} R_{kj}^{\mathcal{H}} & if\ i > 0\ and\ \sigma_i = d_j \\ Q_{kj}^{\mathcal{H}} & otherwise \end{cases}$$

■

Lemma 42 Any proper non-halting computation and any improper computation of $\Delta_{\mathcal{M}}$ can be matched with respect to $\lessapprox_{loc}$ by a computation of $\Delta'_{\mathcal{M}}$, *i.e.* for each proper and non-halting or improper (possibly halting) computation

$$c : X_1 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of $\Delta_{\mathcal{M}}$ there exists a computation

$$d : Y_1 \overset{\sigma_1}{\underset{l_1}{\Longrightarrow}} F_1 \ldots \overset{\sigma_n}{\underset{l_n}{\Longrightarrow}} F_n$$

of $\Delta'_{\mathcal{M}}$ such that for each $i \in [n]$, $F_i = f_i^c(E_i)$. Moreover, the case where c proper non-halting computation the match d is unique.

Proof: Let $f_0^c(E_0) = f_0^c(X_1) = Y_1 = F_0$. Given a computation

$$c : X_1 = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of $\Delta_{\mathcal{M}}$, we construct a unique computation

$$d : Y_1 = F_0 \overset{\sigma_1}{\underset{l_1}{\Longrightarrow}} F_1 \ldots \overset{\sigma_n}{\underset{l_n}{\Longrightarrow}} F_n$$

of $\Delta'_{\mathcal{M}}$ such that for each $i \in \{0, \ldots, n\}$, $F_i = f_i^c(E_i)$. We show that for each $i \in [n-1]$ and $j = 0, 1$,

$$E_i \overset{\sigma_{i+1}}{\underset{l_{i+1}}{\longrightarrow}} E_{i+1} \textit{ implies } F_i \overset{\sigma_{i+1}}{\underset{l_{i+1}}{\Longrightarrow}} F_{i+1}$$

from which the result follows by induction in the length n of c. Let j range over $0, 1$. Given $i = 0, \ldots, n-1$, we divide the proof into three cases:

1. The computation E_0 to E_i a proper non-halting computation and the $(i+1)$th transition is proper.

   (a) for $\sigma_{i+1} = i_j, z_j, g_j, g'_j$,

   $$E_i \overset{\sigma_{i+1}}{\underset{l_{i+1}}{\longrightarrow}} E_{i+1} \text{ is matched by } F_i \overset{\sigma_{i+1}}{\underset{l_{i+1}}{\longrightarrow}} F_{i+1}, \text{ and}$$

   (b) for $\sigma_{i+1} = d_j$,

   $$E_i \overset{d_j}{\underset{l_{i+1}}{\longrightarrow}} E_{i+1} \text{ is matched by } F_i \overset{\tau}{\underset{l'_{i+1}}{\longrightarrow}} F'_i \overset{d_j}{\underset{l_{i+1}}{\longrightarrow}} F''_i \overset{\tau}{\underset{l'_{i+1}}{\longrightarrow}} F_{i+1}$$

   where for some $m_0, m_1 \in I\!N$ $(m_j > 0)$,

   $$\begin{array}{rcl} F'_i & \equiv & t'_j.R_{k_ij} \parallel d_j.\bar{t}'_j.0 \parallel D_0^{m_j-1} \parallel D_1^{m_{j\oplus 1}}, \\ F''_i & \equiv & t'_j.R_{k_ij} \parallel \bar{t}'_j.0 \parallel D_0^{m_j-1} \parallel D_1^{m_{j\oplus 1}}, \end{array}$$

   and $l'_{i+1} = l_{i+1} \cup \{l\}$ where l is the location of *control*(F_i) in F_i (and in particular, of R_{k_ij} in F'_i and F''_i.)

2. The computation E_0 to E_i a proper non-halting computation and the $(i+1)$th transition is improper.

   (a) *(leaving out decrement)*

   $$E_i \xrightarrow[l_{i+1}]{g'_j} E_{i+1} \text{ is matched by } F_i \xrightarrow[l_{i+1}]{g'_j} F_{i+1}$$

   (b) *(choosing zero-branch on non-zero counter)*

   $$E_i \xrightarrow[l_{i+1}]{z_j} E_{i+1} \text{ is matched by } F_i \xrightarrow[l'_{i+1}]{\tau} F'_i \xrightarrow[l_{i+1}]{z_j} F_{i+1},$$

   where for some label p and $m_0, m_1 \in I\!N$,

   $$F'_i \equiv z_j.Y_p^{\mathcal{H}} \parallel D_0^{m_0} \parallel D_1^{m_1},$$

   and $l'_{i+1} = l_{i+1} \cup \{l\}$ where l is the location of some C_j in E_i (and in particular, of some D_j in F_i.)

   (c) *(decrement of the wrong counter or wrong decrement of a counter)*
   For some label k and $m_0, m_1 \in I\!N$, let $E_i \equiv P_{kj} \parallel C_0^{m_0} \parallel C_1^{m_1}$, then note that necessarily $i > 0$ and that the state of F_i dependent on whether or not $\sigma_i = d_j$ in any case though

   $$E_i \xrightarrow[l_{i+1}]{d_{j\oplus 1}} E_{i+1} \text{ is matched by } F_i \xrightarrow[l'_{i+1}]{\tau} F'_i \xrightarrow[l_{i+1}]{d_{j\oplus 1}} F''_i \xrightarrow[l'_{i+1}]{\tau} F_{i+1}$$

   where for some label k and $m_0, m_1 \in I\!N$,

   $$\begin{array}{rcl} F'_i & \equiv & t'_{j\oplus 1}.Q_{kj\oplus 1} \parallel d_{j\oplus 1}.\bar{t}'_{j\oplus 1}.0 \parallel D_0^{m_{j\oplus 1}-1} \parallel D_1^{m_j}, \\ F''_i & \equiv & t'_{j\oplus 1}.Q_{kj\oplus 1} \parallel \bar{t}'_{j\oplus 1}.0 \parallel D_0^{m_{j\oplus 1}-1} \parallel D_1^{m_j}, \end{array}$$

   and $l'_{i+1} = l_{i+1} \cup \{l\}$ where l is the location of P_{kj} in E_i (and in particular, of Q_{kj} in F_i.)

   (d) *(wrong decrement of a counter)*
   For some label k and $m_0, m_1 \in I\!N$, let $E_i \equiv P_{kj} \parallel C_0^{m_0} \parallel C_1^{m_1}$, and let $\sigma_i = d_j$ (note that necessarily $i > 0$)

   $$E_i \xrightarrow[l_{i+1}]{d_j} E_{i+1} \text{ is matched by } F_i \xrightarrow[l'_{i+1}]{\tau} F'_i \xrightarrow[l_{i+1}]{d_j} F''_i \xrightarrow[l'_{i+1}]{\tau} F_{i+1}$$

   where for some label k and $m_0, m_1 \in I\!N$,

   $$\begin{array}{rcl} F'_i & \equiv & t'_{j\oplus 1}.R_{kj\oplus 1} \parallel d_{j\oplus 1}.\bar{t}'_{j\oplus 1}.0 \parallel D_0^{m_{j\oplus 1}-1} \parallel D_1^{m_j}, \\ F''_i & \equiv & t'_{j\oplus 1}.R_{kj\oplus 1} \parallel \bar{t}'_{j\oplus 1}.0 \parallel D_0^{m_{j\oplus 1}-1} \parallel D_1^{m_j}, \end{array}$$

   and $l'_{i+1} = l_{i+1} \cup \{l\}$ where l is the location of P_{kj} in E_i (and in particular, of R_{kj} in F_i.)

(e) *(wrong decrement of a counter)*
For some label k and $m_0, m_1 \in I\!N$, $E_i \equiv X_k \parallel C_0^{m_0} \parallel C_1^{m_1}$,

$$E_i \underset{l_{i+1}}{\overset{d_j}{\longrightarrow}} E_{i+1} \text{ is matched by } F_i \underset{l'_{i+1}}{\overset{\tau}{\longrightarrow}} F'_i \underset{l_{i+1}}{\overset{d_j}{\longrightarrow}} F''_i \underset{l'_{i+1}}{\overset{\tau}{\longrightarrow}} F_{i+1}$$

where

$$\begin{array}{rcl} F'_i & \equiv & t'_j.Y_k^{\mathcal{H}} \parallel d_j.\bar{t}'_j.0 \parallel D_0^{m_j-1} \parallel D_1^{m_j}, \\ F''_i & \equiv & t'_j.Y_k^{\mathcal{H}} \parallel \bar{t}'_j.0 \parallel D_0^{m_j-1} \parallel D_1^{m_j}, \end{array}$$

and $l'_{i+1} = l_{i+1} \cup \{l\}$ where l is the location of X_k in E_i (and in particular, of Y_k in F_i.)

3. The computation from E_0 to E_i an improper computation. In this case, it easy to see that once in an $\mathcal{H}$-labelled state the matching is straightforward.

$\square$

Lemma 43 $\Delta_{\mathcal{M}} \lessapprox_{loc} \Delta'_{\mathcal{M}}$ *if and only if* $\mathcal{M}$ *does not halt*

Proof: Assume that $\mathcal{M}$ does not halt $(0,0)$. Then by Lemma 39, $\Delta_{\mathcal{M}}$ has no proper halting computation. Hence, $\Delta_{\mathcal{M}} \lessapprox_{loc} \Delta'_{\mathcal{M}}$ by Lemma 42.

Conversely, assume that $\mathcal{M}$ does halt on input $(0,0)$. Then by Lemma 39, there is a unique proper halting computation of $\Delta_{\mathcal{M}}$

$$c : X_1 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n \underset{l_{n+1}}{\overset{h}{\longrightarrow}} E_{n+1}.$$

The by Lemma 42, unique matching computation of

$$c' : X_1 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

of $\Delta'_{\mathcal{M}}$ is

$$d : Y_1 \underset{l_1}{\overset{\sigma_1}{\Longrightarrow}} F_1 \ldots \underset{l_n}{\overset{\sigma_n}{\Longrightarrow}} F_n.$$

Since c is proper the final state of d is $f_n(E_n) = F_n \equiv Y_m \parallel D_0^{m_0} \parallel D_1^{m_1}$ which does not enable h. Hence by the uniqueness of the match, we conclude that

$$\Delta_{\mathcal{M}} \not\lessapprox_{loc} \Delta'_{\mathcal{M}}.$$

$\square$

Theorem 44 For BPP$_M$, $\lessapprox_{loc}$ and $\approx_{loc}$ are undecidable.

Proof: Immediate consequence of Lemma 43, Theorem 9 and Lemma 36. $\square$

The result is in fact slightly stronger since the first encoding is only a BPP process.

7.2 Weak pomset equivalence

Next, we turn to weak pomset equivalence for BPP_M. We leave the decidability of unsettled. Instead, we give two characterisations which might be helpful in settling the question.

Definition 45 Processes E and E' are said to be *weak tree-pomset preordered*, $E \lessapprox^{tree}_{pom} E'$, iff for every computation of E

$$c : E \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

without communication there exists a computation of E'

$$c' : E' \underset{l'_1}{\overset{\sigma'_1}{\longrightarrow}} E'_1 \ldots \underset{l'_m}{\overset{\sigma'_m}{\longrightarrow}} E'_m$$

(possibly with communication) such that $\| c \| \;=\; \| c' \|$, for each $i \in [\| c \|]$, $\sigma_{v^c(i)} = \sigma'_{v^{c'}(i)}$ and furthermore for each $i, j \in [\| c \|]$, $v^c(i) \leq^*_c v^c(j) \iff v^{c'}(i) \leq^*_{c'} v^{c'}(j)$. ■

Proposition 46 Let E and F be BPP_M processes.

$$E \lessapprox_{pom} F \text{ if and only if } E + F \approx_{pom} F.$$

Proof: Straightforward. □

Proposition 47 Let E and F be BPP_M processes.

$$E \lessapprox_{pom} F \text{ if and only if } E \lessapprox^{tree}_{pom} F.$$

Proof: The only if direction is obvious. To see the other direction, observe that a computation with communication can be split into one without communication - a tree - which can be matched by assumption, and clearly the match composes to a match for the original computation with communication. Following this argument it is easy to do an induction proof in the number of communications occuring in a computation.

Assume that $E \lessapprox^{tree}_{pom} F$. We show by induction in the number of communications that for every computation

$$c : E = E_0 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

of E there exists a computation

$$d : F = F_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} F_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} F_n$$

of F such that d is a match of c with respect to $\approx_{pom}$.

In the base case, no communications occur in c and hence the existence of d follows from the assumption.

In the induction step, assume that $\sigma_k = \tau$ $(1 \leq k \leq n)$. By Lemma 59,

$$c' : E = E_0 \overset{\sigma_1'}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_{k-1}'}{\underset{l_{k-1}}{\longrightarrow}} E_{k-1} \overset{\mu}{\underset{u_1}{\longrightarrow}} E_k' \overset{\bar{\mu}}{\underset{u_2}{\longrightarrow}} E_k \overset{\sigma_{k+1}'}{\underset{l_{k+1}}{\longrightarrow}} E_{k+1} \ldots \overset{\sigma_m'}{\underset{l_m}{\longrightarrow}} E_m$$

is a computation of E such that $u_1 \not\sqsubseteq u_2$ and $\mu \neq \tau$. Then by induction, there exists a computation

$$d' : F = F_0 \overset{\sigma_1'}{\underset{l_1'}{\longrightarrow}} F_1 \ldots \overset{\sigma_{k-1}'}{\underset{l_{k-1}'}{\longrightarrow}} F_{k-1} \overset{\mu}{\underset{v_1}{\longrightarrow}} F_k' \overset{\bar{\mu}}{\underset{v_2}{\longrightarrow}} F_k \overset{\sigma_{k+1}'}{\underset{l_{k+1}'}{\longrightarrow}} F_{k+1} \ldots \overset{\sigma_m'}{\underset{l_m'}{\longrightarrow}} F_m$$

such that $\| c' \| = \| d' \|$ and $\sigma_{v^{c'}(i)} = \sigma'_{v^{d'}(i)}$ for $i \in [\| c' \|]$, and c' furthermore satisfies that $v^{c'}(i) \leq^*_{c'} v^{c'}(j) \iff v^{d'}(i) \leq^*_{d'} v^{d'}(j)$. Since $u_1 \not\sqsubseteq u_2$ and

$$u_1 \not\sqsubseteq u_2 \Longrightarrow k \not\leq^*_{c'} k+1 \Longrightarrow k \not\leq^*_{d'} k+1 \Longrightarrow v_1 \not\sqsubseteq v_2,$$

it is not hard using Lemma 60 to verify that

$$d : F = F_0 \overset{\sigma_1}{\underset{l_1'}{\longrightarrow}} F_1 \ldots \overset{\sigma_n}{\underset{l_n'}{\longrightarrow}} F_n$$

is a computation of F with $l_m' = v_1 \cup v_2$ and such that d is a match of c with respect to $\approx_{pom}$.

By induction and a symmetric argument, we conclude that $E \lessapprox_{pom} F$. $\square$

Let Δ_1 and Δ_2 be BPP_M processes, we can summarise the results of Proposition 46 and 47 as follows. The following problems are equivalent with respect to decidability:

(i) $\Delta_1 \approx_{pom} \Delta_2$,

(ii) $\Delta_1 \lessapprox_{pom} \Delta_2$, and

(iii) $\Delta_1 \lessapprox^{tree}_{pom} \Delta_2$.

8 BPP^τ_M

In this section, we are able to complete the picture for location equivalence.

The following example shows that with τ prefixing the characterization no longer holds for pomset equivalence.

Example 48 Let

$$s_1 = a.b.0 \parallel \bar{b}.c.0 + a.\tau.c.0 \text{ and } s_2 = a.b.0 \parallel \bar{b}.c.0.$$

Then, s_1 and s_2 are pomset equivalent when communication is allowed, that is, as BPP_M^τ processes, and not when it is disallowed, that is, as BPP^τ processes. Moreover as BPP_M^τ processes, $s_1 \sim_{pom} s_2$ but $s_1 \not\sim_{loc} s_2$. ■

Theorem 49 For BPP_M^τ, $\sim_{loc} \subset \sim_{pom} \subset \sim_{lan}$.

Proof: The inclusions follow by definition and the properness from Example 48 and 5. □

8.1 Location equivalence

For location equivalence, the decidability proof of BPP_M straightforwardly extends to the case with τ prefixing, since τ-actions stemming from prefixing and those stemming from communications cannot be confused.

Lemma 50 The BPP_M^τ processes E and F are location equivalent if and only if the BPP^τ processes E and F are location equivalent.

Proof: For the only if direction, assume that E and F are location equivalent as BPP_M^τ processes. Let

$$c : E = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

be a BPP^τ computation of E, since any BPP^τ computation is also a BPP_M^τ computation there exists a BPP^τ computation

$$d : F = F_0 \overset{\sigma_1}{\underset{l'_1}{\longrightarrow}} F_1 \ldots \overset{\sigma_n}{\underset{l'_n}{\longrightarrow}} F_n$$

of F such that there exists a relation $\mathcal{R} \subseteq loc(c) \times loc(d)$ satisfying that for each $1 \leq i \leq n$, $\mathcal{R}$ restricts to a bijection on $l_i \times l'_i$, and for each $i \leq j$, $s_0(\mathcal{R} \cap l_i \times l'_i)s'_0$ and $s_1(\mathcal{R} \cap l_j \times l'_j)s'_1$, $s_0 \sqsubseteq s_1 \iff s'_0 \sqsubseteq s'_1$

Since $\mathcal{R}$ restricts to a bijection on $l_i \times l'_i$, l_i and l'_i have the same cardinality. But, c is a BPP^τ computation and thus each l_i is a singleton and thus there cannot be communications in d. Therefore, d is a BPP^τ computation which matches c with respect to $\sim_{loc}$.

For the if direction, assume that $E \sim_{loc} F$ when E and F are considered as BPP^τ processes. We show by induction in the number of communications that for every computation

$$c : E = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of E there exists a computation

$$d : F = F_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} F_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} F_n$$

of F such that there exists a relation $\mathcal{R} \subseteq loc(c) \times loc(d)$ satisfying that for each $1 \leq i \leq n$, $\mathcal{R}$ restricts to a bijection on $l_i \times l'_i$, and for each $i \leq j$, $s_0(\mathcal{R} \cap l_i \times l'_i)s'_0$ and $s_1(\mathcal{R} \cap l_j \times l'_j)s'_1$, $s_0 \sqsubseteq s_1 \iff s'_0 \sqsubseteq s'_1$.

In the base case, no communications occur in c and hence the existence of d follows from the assumption.

In the induction step, assume that $\sigma_m = \tau$ ($m \in [n]$) such that σ_n stems from a communication By Lemma 59,

$$c' : E = E_0 \underset{l^1_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l^1_{m-1}}{\overset{\sigma_{m-1}}{\longrightarrow}} E_{m-1} \underset{l^1_m}{\overset{\mu}{\longrightarrow}} E'_m \underset{l^1_{m+1}}{\overset{\bar{\mu}}{\longrightarrow}} E_m \underset{l^1_{m+2}}{\overset{\sigma_{m+1}}{\longrightarrow}} E_{m+1} \ldots \underset{l^1_{n+1}}{\overset{\sigma_n}{\longrightarrow}} E_n$$

is a computation of E such that $u_1 \not\sqsubseteq u_2$, $\mu \neq \tau$ and for each $i \in [n+1]$,

$$l^1_i = \begin{cases} l_i & \textit{if } i < m \\ \{u_1\} & \textit{if } i = m \\ \{u_2\} & \textit{if } i = m+1 \\ l_{i-1} & \textit{if } i > m+1. \end{cases}$$

Then by induction, there exists a computation

$$d' : F = F_0 \underset{l^2_1}{\overset{\sigma_1}{\longrightarrow}} F_1 \ldots \underset{l^2_{m-1}}{\overset{\sigma_{m-1}}{\longrightarrow}} F_{m-1} \underset{l^2_m}{\overset{\mu}{\longrightarrow}} F'_m \underset{l_{m+1}}{\overset{\bar{\mu}}{\longrightarrow}} F_m \underset{l^2_{m+2}}{\overset{\sigma_{m+1}}{\longrightarrow}} F_{m+1} \ldots \underset{l^2_{n+1}}{\overset{\sigma_n}{\longrightarrow}} F_n$$

of F such that there exists a relation $\mathcal{R} \subseteq loc(c') \times loc(d')$ satisfying that for each $i \in [n+1]$, $\mathcal{R}$ restricts to a bijection on $l^1_i \times l^2_i$, and for each $i \leq j$, $s_0(\mathcal{R} \cap l^1_i \times l^2_i)s'_0$ and $s_1(\mathcal{R} \cap l^1_j \times l^2_j)s'_1$, $s_0 \sqsubseteq s_1 \iff s'_0 \sqsubseteq s'_1$.

By a cardinality argument, there exist v_1 and v_2 such that $l^2_m = \{v_1\}$ and $l^2_{m+1} = \{v_2\}$. Since $u_1 \not\sqsubseteq u_2$, we hence get that $v_1 \not\sqsubseteq v_2$. By Lemma 60, it follows that

$$d : F = F_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} F_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} F_n$$

is a computation of F, where for each $i \in [n]$,

$$l'_i = \begin{cases} l^2_i & \textit{if } i < m \\ \{v_1, v_2\} & \textit{if } i = m \\ l^2_{i+1} & \textit{if } i \geq m+1 \end{cases}$$

Moreover, $loc(c) = loc(c')$, $loc(d) = loc(d')$ and for each $i \in [n]$, $\mathcal{R}$ restricts to a bijection on $l_i \times l'_i$, and for each $i \leq j$, $s_0(\mathcal{R} \cap l_i \times l'_i)s'_0$ and $s_1(\mathcal{R} \cap l_j \times l'_j)s'_1$, $s_0 \sqsubseteq s_1 \iff s'_0 \sqsubseteq s'_1$.

By induction and a symmetric argument, we conclude that $E \sim_{loc} F$. $\square$

Theorem 51 For BPP^{τ}_{M}, $\sim_{loc}$ is decidable whereas $\approx_{loc}$ is undecidable.

Proof: In the strong case, the results follows from Lemma 50 and the decidability of location equivalence on BPP [26]. The weak case is a straightforward consequence of Lemma 44. □

8.2 Pomset equivalence

The positive decidability results in [26] rely on reductions to problems about automata on trees. In this section, we follow the same strategy in investigate the decidability of pomset equivalence of BPP^{τ}_{M} and give a characterisation in terms a containment problem between finite tree automata and a family of finite tree automata. The characterisation does not settle the question of decidability but we hope that the rephrasing might be a step towards establishing decidability.

8.2.1 Finite Tree Automata

In [26], we showed how to effectively construct a finite tree automaton $\mathcal{A}_{\Delta}$ from a BPP family Δ in normal form. Building on this result, we exhibit a similar construction for BPP^{τ}_{M} families. The construction is however more complex and involves tree languages which are not recognisable.

Let $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_n$ be a ranked finite alphabet. The set of all trees over Σ, T_{Σ} is the free term algebra over Σ, that is, T_{Σ}, is the least set such that $\Sigma_0 \subseteq T_{\Sigma}$ and such that if $a \in \Sigma_k$ and for $i = 1, \ldots, k$, $t_i \in T_{\Sigma}$, then $a[t_1, \ldots, t_k] \in T_{\Sigma}$. For convenience, we use a and $a[]$ interchangeably to denote members of Σ_0.

Definition 52 A non-deterministic top-down finite tree automaton, *NTA*, is a four-tuple $\mathcal{A} = (\Sigma, Q, S, \delta)$, where Σ is a ranked finite alphabet, Q a finite set of states, $S \subseteq Q$ is a set of initial states, and δ is a ranked family of labelled transition relations associating with each $k \geq 0$, a relation $\delta_k \subseteq Q \times \Sigma_k \times Q^k$ such that δ_k is non-empty for only finitely many k. ■

Definition 53 Let $\mathcal{A} = (\Sigma, Q, S, \delta)$ be a *NTA* and let $t \in T_{\Sigma}$. A *configuration* of $\mathcal{A}$, is a multiset of pairs from $Q \times T_{\Sigma}$. Denote by $conf_{\mathcal{A}}$ the set of all configurations of $\mathcal{A}$. For $\sigma \in \Sigma$, let $\xrightarrow{\sigma} \subseteq conf_{\mathcal{A}} \times conf_{\mathcal{A}}$ be the labelled transition relation between configurations defined by

$$\{|(q,t)|\} \cup c \xrightarrow{\sigma} \{|(q_1,t_1), \ldots, (q_k,t_k)|\} \cup c,$$

if and only if $\sigma \in \Sigma_k$, $t = \sigma[t_1, \ldots, t_k]$, $(q, \sigma, q_1, \ldots, q_k) \in \delta_k$ and c $\in conf_{\mathcal{A}}$. We write $\rightarrow$ for the union over all $\sigma \in \Sigma$ of $\xrightarrow{\sigma}$, and $\rightarrow^*$ for the reflexive and transitive closure of $\rightarrow$. A (*successful*) *run* of $\mathcal{A}$ on input t is a derivation $\{|(q_0,t)|\} \rightarrow^* \emptyset$, where $q_0 \in S$. The tree language, $L(\mathcal{A})$, *recognised* by $\mathcal{A}$ consists of all trees t, for which there is a successful run of $\mathcal{A}$ on t. ■

Definition 54 Given a BPP family Δ in normal form with leading variable X_1, define the *NTA* $\mathcal{A}_\Delta = (\mathcal{A}ct(\Delta), Var(\Delta), \{X_1\}, \delta)$ such that for every $(X \stackrel{\text{def}}{=} \sum_{i=1}^{n} \sigma_i \alpha_i) \in \Delta$, every index $1 \leq j \leq n$ and for every $\{|Y_1, \ldots, Y_k|\} \subseteq \alpha_j$,

$$(X, \sigma_j, Y_1, \ldots, Y_k) \in \delta_k.$$

The ranking of the alphabet $\mathcal{A}ct(\Delta)$ is induced by the definition of δ. ■

In [26], the following characterisation was shown.

Proposition 55 [26] Given BPP families Δ_1 and Δ_2 in normal form. Then

$$\Delta_1 \sim_{pom} \Delta_2 \iff \mathcal{L}(\mathcal{A}_{\Delta_1}) = \mathcal{L}(\mathcal{A}_{\Delta_2})$$

□

The proposition above does not hold for BPP_M families as shown by Example 48.

Here, the first characterisation is given in terms of the obvious pomset preorder.

Definition 56 Let E and E' be BPP_M^τ processes. $E \lesssim_{pom} E'$ iff for every computation of E

$$c : E \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

there exists a computation of E'

$$c' : E' \xrightarrow[l'_1]{\sigma_1} E'_1 \ldots \xrightarrow[l'_n]{\sigma_n} E'_n$$

such that $i \leq_c^* j \iff i \leq_{c'}^* j$. ■

Proposition 57 Let E and F be BPP_M^τ processes.

$$E \lesssim_{pom} F \text{ if and only if } E + F \sim_{pom} F.$$

Proof: Straightforward. □

Definition 58 Let E and E' be BPP_M^τ processes. $E \lesssim_{pom}^{tree} E'$ iff for every computation of E

$$c : E \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

without communication there exists a computation of E'

$$c' : E' \xrightarrow[l'_1]{\sigma_1} E'_1 \ldots \xrightarrow[l'_n]{\sigma_n} E'_n$$

(possibly with communication) such that $i \leq_c^* j \iff i \leq_{c'}^* j$. We say that E and E' are *pomset tree equivalent*, $E \sim_{pom}^{tree} E'$, iff $E \lesssim_{pom}^{tree} E'$ and $E' \lesssim_{pom}^{tree} E$. ■

It is an easy exercise to show the following lemmas.

Lemma 59 If $E \xrightarrow[l]{\tau} G$ and τ stems from a communication then there exist an expression $F \in Proc$, an action $\sigma \in \mathcal{A}ct$ and locations l_1 and l_2 such that $l = l_1 \cup l_2$, $\neg(l_1 \sqsubseteq l_2)$ and $E \xrightarrow[l_1]{\sigma} F \xrightarrow[l_2]{\bar{\sigma}} G$. $\square$

Lemma 60 If $E \xrightarrow[l_1]{\sigma} F \xrightarrow[l_2]{\bar{\sigma}} G$ and $\neg(l_1 \sqsubseteq l_2)$ then $E \xrightarrow[l_1 \cup l_2]{\tau} G$. $\square$

Proposition 61 Let E and F be BPP_M^τ processes.

$$E \lesssim_{pom} F \text{ if and only if } E \lesssim_{pom}^{tree} F.$$

Proof: The only if direction is obvious. To see the other direction, observe that a computation with communication can be split into one without communication - a tree - which can be matched by assumption, and clearly the match composes to a match for the original computation with communication. Following this argument it is easy to do an induction proof in the number of communications occurring in a computation.

Assume that $E \lesssim_{pom}^{tree} F$. We show by induction in the number of communications that for every computation

$$c : E = E_0 \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

of E there exists a computation

$$d : F = F_0 \xrightarrow[l'_1]{\sigma_1} F_1 \ldots \xrightarrow[l'_n]{\sigma_n} F_n$$

of F such that $i \leq_c^* j \iff i \leq_d^* j$.

In the base case, no communications occur in c and hence the existence of d follows from the assumption.

In the induction step, assume that $\sigma_m = \tau$ ($m \in [n]$) and σ_m stems from a communication. By Lemma 59,

$$c' : E = E_0 \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_{m-1}]{\sigma_{m-1}} E_{m-1} \xrightarrow[u_1]{\mu} E'_m \xrightarrow[u_2]{\bar{\mu}} E_m \xrightarrow[l_{m+1}]{\sigma_{m+1}} E_{m+1} \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

is a computation of E such that $u_1 \not\sqsubseteq u_2$ and $\mu \neq \tau$. Then by induction, there exists a computation

$$d' : F = F_0 \xrightarrow[l'_1]{\sigma_1} F_1 \ldots \xrightarrow[l'_{m-1}]{\sigma_{m-1}} F_{m-1} \xrightarrow[v_1]{\mu} F'_m \xrightarrow[v_2]{\bar{\mu}} F_m \xrightarrow[l'_{m+1}]{\sigma_{m+1}} F_{m+1} \ldots \xrightarrow[l'_n]{\sigma_n} F_n$$

such that $i \leq_{c'}^* j \iff i \leq_{d'}^* j$. Since $u_1 \not\sqsubseteq u_2$ and

$$u_1 \not\sqsubseteq u_2 \Longrightarrow m \not\leq_{c'}^* m+1 \Longrightarrow m \not\leq_{d'}^* m+1 \Longrightarrow v_1 \not\sqsubseteq v_2,$$

it follows from Lemma 60, that

$$d : F = F_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} F_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} F_n$$

is a computation of F, where $l'_m = v_1 \cup v_2$. Moreover, it is not hard to check that $i \leq_c^* j \iff i \leq_d^* j$. By induction and a symmetric argument, we conclude that $E \lesssim_{pom} F$. □

By Proposition 61, it suffices to match tree-ordered pomsets. Because tree-ordered pomsets are obviously only matched by tree-ordered pomsets, we next move to explicitly compute the tree-ordered pomsets arising through communication and then to forget about communication. The idea is to approximate a BPP_M^τ family Δ by a family of BPP^τ families $\{\Delta^{\tau K} \mid K \in \mathbb{N}\}$ in such a way that any tree-ordered pomset of Δ is also a pomset of $\Delta^{\tau K}$ for some $K \in \mathbb{N}$. The construction is based on augmenting variables with a memory used to remember processes available for communication. The memory is decreased when communicating and increased by non-deterministically picking up "brothers". The main observation is that when only tree-ordered pomsets are considered, the "brothers" are in fact the only possible candidates for communications "later on".

Example 62 Consider the BPP_M^τ family

$$\begin{aligned}
\Delta = \{ \quad & X_1 \stackrel{\text{def}}{=} a.\{X_1, X_2\} + a.\{X_2, X_3\}, \\
& X_2 \stackrel{\text{def}}{=} \bar{c}.\emptyset, \\
& X_3 \stackrel{\text{def}}{=} c.\{X_4\}, \\
& X_4 \stackrel{\text{def}}{=} b.\{X_3\} \qquad \qquad \qquad \}.
\end{aligned}$$

For each natural number K, define the BPP^τ family

$$\begin{aligned}
\Delta^{\tau K} = \{ \quad X_1(\bar{m}) \stackrel{\text{def}}{=} \; & a.\emptyset + \\
& a.\textstyle\sum\{\{X_1(\bar{m}')\} \mid \bar{0} \leq \bar{m}' - \bar{m} \leq \bar{e}_2\} + \\
& a.\textstyle\sum\{\{X_2(\bar{m}')\} \mid \bar{0} \leq \bar{m}' - \bar{m} \leq \bar{e}_1\} + \\
& a.\textstyle\sum\{\{X_2(\bar{m}')\} \mid \bar{0} \leq \bar{m}' - \bar{m} \leq \bar{e}_3\} + \\
& a.\textstyle\sum\{\{X_3(\bar{m}')\} \mid \bar{0} \leq \bar{m}' - \bar{m} \leq \bar{e}_2\} + \\
& a.\textstyle\sum\{\{X_1(\bar{m}_1), X_2(\bar{m}_2)\} \mid \bar{m}_1 + \bar{m}_2 \leq \bar{m}\} + \\
& a.\textstyle\sum\{\{X_2(\bar{m}_1), X_3(\bar{m}_2)\} \mid \bar{m}_1 + \bar{m}_2 \leq \bar{m}\}, \\
X_2(\bar{m}) \stackrel{\text{def}}{=} \; & \bar{c}.\emptyset + \tau.\emptyset, && \text{if } 0 \leq \bar{m} - \bar{e}_3, \\
X_2(\bar{m}) \stackrel{\text{def}}{=} \; & \bar{c}.\emptyset, && \text{if not } 0 \leq \bar{m} - \bar{e}_3, \\
X_3(\bar{m}) \stackrel{\text{def}}{=} \; & c.\emptyset + c.\{X_4(\bar{m})\} + \\
& \tau.\emptyset + \tau.\{X_4(\bar{m} - \bar{e}_2)\}, && \text{if } 0 \leq \bar{m} - \bar{e}_2, \\
X_3(\bar{m}) \stackrel{\text{def}}{=} \; & c.\emptyset + c.\{X_4(\bar{m})\}, && \text{if not } 0 \leq \bar{m} - \bar{e}_2, \\
X_4(\bar{m}) \stackrel{\text{def}}{=} \; & b.\emptyset + b.\{X_3(\bar{m})\} \mid \bar{0} \leq \bar{m} \leq \bar{K}\}.
\end{aligned}$$

■

Note that the constructed families approximate Δ from below in the sense that for each $K \in I\!N$, $\Delta^{\tau K} \lesssim_{pom} \Delta$.

Next, we formally define the approximations but first a convenient technical definition.

Definition 63 Let $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i \in [n]\}$ be a BPP_M^τ family, Let $\bar{m}_i$ range over $\mathbb{N}^n$ For convenience, we denote by $\{|X_{i_1}, \ldots, X_{i_k}|\}\langle \bar{m}_1, \ldots, \bar{m}_k\rangle$ $(i_1 \leq i_2 \leq \ldots \leq i_k)$ the set $\{|X_{i_1}(\bar{m}_1), \ldots, X_{i_k}(\bar{m}_k)|\}$. For each $\bar{m} \in \mathbb{N}^n$, and subset α of $Var(\Delta)$, let

$$\alpha\langle\langle \bar{m}\rangle\rangle = \sum\{\beta\langle \bar{m}_1, \ldots, \bar{m}_{|\beta|}\rangle \mid \beta \subseteq \alpha,\ \bar{0} \leq \sum_i \bar{m}_i - \bar{m} \leq \bar{m}(\alpha - \beta)\}.$$

■

Definition 64 Let $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i \in [n]\}$ be a BPP_M^τ family in normal form with $E_i \equiv \sum_{j=1}^{n_i} \sigma_{ij}\alpha_{ij}$ Define for each $K \in \mathbb{N}$ the K *–approximations* BPP^τ family,

$$\Delta^{\tau K} = \{X_i(\bar{m}) \stackrel{\text{def}}{=} F_i(\bar{m}) \mid i \in [n]\ \wedge\ \bar{0} \leq \bar{m} \leq \bar{K}\},$$

with leading variable $X_1(\bar{0})$ where for each $i \in [n]$ and $\bar{m} \leq \bar{K}$,

$$F_i(\bar{m}) \equiv \sum_{j=1}^{n_i} \sigma_{ij}\alpha_{ij}\langle\langle \bar{m}\rangle\rangle + \sum_{j=1}^{n_i} \sum\{\tau(\alpha_{ij} \cup \beta)\langle\langle \bar{m} - \bar{e}_k\rangle\rangle \mid \bar{e}_k \leq \bar{m}\ \wedge\ \bar{\sigma}_{ij}\gamma \in E_k\ \wedge\ \beta \subseteq \gamma\}.$$

■

The next lemma captures the use of approximation families in making communication dispensable when checking for $\lesssim_{pom}^{tree}$-containment.

Lemma 65 Let Δ a BPP_M^τ family in normal form with leading variable X_1. Then

(i) for every computation

$$c : X_1 = E_0 \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

of Δ such that $\leq_c^*$ is a tree ordering there exists a $K \in \mathbb{N}$ and a computation

$$c' : X_1(\bar{0}) = E'_0 \xrightarrow[l'_1]{\sigma_1} E'_1 \ldots \xrightarrow[l'_n]{\sigma_n} E'_n$$

without communication of $\Delta^{\tau K}$ such that $i \leq_c^* j \iff i \leq_{c'}^* j$.

(ii) for every $K \in \mathbb{N}$ and every computation

$$c : X_1(\bar{0}) = E_0 \xrightarrow[l_1]{\sigma_1} E_1 \ldots \xrightarrow[l_n]{\sigma_n} E_n$$

of $\Delta^{\tau K}$ without communication there exists a computation

$$c' : X_1 = E'_0 \xrightarrow[l'_1]{\sigma_1} E'_1 \ldots \xrightarrow[l'_n]{\sigma_n} E'_n$$

(possibly with communication) of Δ such that $i \leq_c^* j \iff i \leq_{c'}^* j$.

Proof:

(i) Given a computation

$$c : X_1 = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of Δ such that $\leq_c^*$ is a tree ordering. We proceed by induction in the number of communications K occuring in c.

In the base case, no communications occur in c and it is easy to check that

$$c' : X_1(\bar{0}) = E_0' \overset{\sigma_1}{\underset{l_1'}{\longrightarrow}} E_1' \ldots \overset{\sigma_n}{\underset{l_n'}{\longrightarrow}} E_n'$$

is a computation of $\Delta^{\tau K}$ such that $i \leq_c^* j \iff i \leq_{c'}^* j$, and $E_i' = \eta(E_i)$ where η is the relabeling homomorphism induced by taking for each $X \in Var(\Delta)$, $\eta(X) = X(\bar{0})$.

In the induction step, $K > 0$, assume that $\sigma_m = \tau$ ($m \in [n]$) and that σ_m stems from a communication. By Lemma 59, there is a computation

$$d : X_1 = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_{m-1}}{\underset{l_{m-1}}{\longrightarrow}} E_{m-1} \overset{\mu}{\underset{u_1}{\longrightarrow}} E_m' \overset{\bar{\mu}}{\underset{u_2}{\longrightarrow}} E_m \overset{\sigma_{m+1}}{\underset{l_{m+1}}{\longrightarrow}} E_{m+1} \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of Δ such that $u_1 \not\sqsubseteq u_2$ and $\mu \neq \tau$. Then by induction, since $\leq_d^*$ is a tree ordering, there exists a computation

$$d' : X_1(\bar{0}) = F_0 \overset{\sigma_1}{\underset{l_1'}{\longrightarrow}} F_1 \ldots \overset{\sigma_{m-1}}{\underset{l_{m-1}'}{\longrightarrow}} F_{m-1} \overset{\mu}{\underset{v_1}{\longrightarrow}} F_m' \overset{\bar{\mu}}{\underset{v_2}{\longrightarrow}} F_m \overset{\sigma_{m+1}}{\underset{l_{m+1}'}{\longrightarrow}} F_{m+1} \ldots \overset{\sigma_n}{\underset{l_n'}{\longrightarrow}} F_n$$

of $\Delta^{\tau K}$ such that $i \leq_d^* j \iff i \leq_{d'}^* j$.

Let k be the greatest common predecessor of m and $m+1$ with respect to $\leq_d^*$. Such a k exists since Δ is in normal form and $\leq_d^*$ is a tree ordering.

The important observation is now that either m or $m+1$ is a son (immediate successor) of k since otherwise there exist k_1 and k_2 such that $k_1 \not\leq_d^* k_2$, $k_2 \not\leq_d^* k_1$, $k <_d^* k_1 <_d^* m$ and $k <_d^* k_2 <_d^* m+1$ which contradicts the assumption that $\leq_c^*$ is a tree ordering.

Assume without loss of generality that m is a son of k. From this it is not hard to verify that d' can be modified so that the kth transition picks up the appropriate son into memory sends it along the appriate path while performing the transitions as in d', and when reaching the mth transition performs the communication between μ and $\bar{\mu}$ reaching F_m modulo $\equiv$. Also, it is clear that the obtained computation is a match of c with respect to $\sim_{pom}$.

(ii) Let ι be the homomorphism on induced by letting $\iota(X_i((m_1, \ldots, m_n))) = X_i \parallel X_1^{m_1} \parallel \ldots \parallel X_n^{m_n}$. Given a $K \in \mathbb{N}$ and a computation

$$c : X_1(\bar{0}) = E_0 \overset{\sigma_1}{\underset{l_1}{\longrightarrow}} E_1 \ldots \overset{\sigma_n}{\underset{l_n}{\longrightarrow}} E_n$$

of $\Delta^{\tau K}$ without communication it is not hard to verify that there exists a computation

$$c' : X_1 = E'_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} E'_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} E'_n$$

(possibly with communication) of Δ such that for each $i \in [n]$, $E'_i \equiv \iota(E_i)$ and for each $i, j \in [n]$, $i \leq^*_c j \iff i \leq^*_{c'} j$.

□

Lemma 66 Given BPP^τ_M families Δ_1 and Δ_2 in normal form with leading variables X_1 and Y_1, respectively. Then $\Delta_1 \lesssim^{tree}_{pom} \Delta_2$ if and only if for every computation

$$c : X_1 = E_0 \underset{l_1}{\overset{\sigma_1}{\longrightarrow}} E_1 \ldots \underset{l_n}{\overset{\sigma_n}{\longrightarrow}} E_n$$

of Δ_1 without communication there exists a $K \in \mathbb{N}$ and a computation

$$c' : Y_1(\bar{0}) = E'_0 \underset{l'_1}{\overset{\sigma_1}{\longrightarrow}} E'_1 \ldots \underset{l'_n}{\overset{\sigma_n}{\longrightarrow}} E'_n$$

without communication of $\Delta_2^{\tau K}$ such that $i \leq^*_c j \iff i \leq^*_{c'} j$.

Proof: Straightforward from Lemma 65. □

Definition 67 For each BPP^τ_M family Δ define

$$\mathcal{L}^\tau_\Delta = \bigcup_{K \in \mathbb{N}} \mathcal{L}(\mathcal{A}_{\Delta^{\tau K}})$$

■

As expected and illustrated by the following example the language accepted by a family of approximation automata is not necessarily a recognisable set of trees.

Example 68 Consider the BPP^τ_M family Δ of Example 62. The language $\mathcal{L}^\tau_\Delta$ contains for each $M \in \mathbb{N}$ the tree

$$\underbrace{a - a - \cdots - a}_{M} - \underbrace{\tau - b - \tau - \cdots - \tau - b}_{2M},$$

whereas for each $M, N \in \mathbb{N}$ such that $M < N$ the tree

$$\underbrace{a - a - \cdots - a}_{M} - \underbrace{\tau - b - \tau - \cdots - \tau - b}_{2N}$$

is not contained in $\mathcal{L}^\tau_\Delta$. It follows from a standard pumping argument that the tree language $\mathcal{L}^\tau_\Delta$ cannot be recognisable. ■

Lemma 69 Given BPP^{τ}_{M} families Δ_1 and Δ_2 in normal form. Then

$$\Delta_1 \lesssim^{tree}_{pom} \Delta_2 \iff \mathcal{L}(\mathcal{A}_{\Delta_1}) \subseteq \mathcal{L}^{\tau}_{\Delta_2}$$

Proof: Follows from Lemma 66 and the lemmas of Appendix A of [25]. □

Let Δ_1 and Δ_2 be BPP^{τ}_{M} processes, we can then summarise the consequences of Proposition 57 and 61, and Lemma 69 as follows. The following problems are equivalent with respect to decidability:

(i) $\Delta_1 \sim_{pom} \Delta_2$,

(ii) $\Delta_1 \lesssim_{pom} \Delta_2$,

(iii) $\Delta_1 \lesssim^{tree}_{pom} \Delta_2$, and

(iv) $\mathcal{L}(\mathcal{A}_{\Delta_1}) \subseteq \mathcal{L}^{\tau}_{\Delta_2}$.

9 Conclusion

Continuing the systematic study initiated in [26], we have presented results illuminating the sometimes delicate bounds between the decidable and the undecidable in the setting of behavioural equivalences for infinite-state concurrent systems. In particular, we have shown that renaming and hiding may make a difference with respect to decidability and given the – to our best knowledge – first positive decidability result for a natural *weak* behavioural equivalence on the full class of BPP^{τ} processes.

Many other non-interleaving equivalences exist besides our chosen pomset and location equivalences, and which deserve to be explored. For instance, the augmentation closure of Pratt [22] is an obvious candidate. Also we would like to emphasise that we do not claim that our notion of location equivalence is the only natural capture of spatial distribution, other possibilities exist.

We showed that pomset and location equivalence are decidable on BPP (BPP^{τ}) with renaming and hiding. The natual next step is to look at BPP_M with renaming and hiding. We know that both equivalences are decidable on BPP_M. But, renaming seems to be considerably more intricate in the presence of communication. In particular, pushing the renaming combinator inwards does not work. For instance, consider the BPP_M processes

$$p = (a.0 \parallel \bar{b}.0)[f] \text{ and } p' = (a.0)[f] \parallel (\bar{b}.0)[f]$$

with $f(a) = b$ and the identity elsewhere, clearly p and p' are not even language equivalent because p' can do a τ-action whereas p cannot. Similarly for the hiding combinator: consider the BPP_M processes

$$q = (a.b.0 \parallel \bar{a}.c.0)\backslash\backslash\{a, \bar{a}\} \text{ and } q' = (a.b.0)\backslash\backslash\{a, \bar{a}\} \parallel (\bar{a}.c.0)\backslash\backslash\{a, \bar{a}\}$$

clearly q and q' are not even language equivalent because q can do a single τ-action and then a b-action followed by a c-action whereas q' need to do two τ-actions in order to do a b-action followed by a c-transition.

The automata characterisation, we gave in Section 8.2 was phrased in terms of a family of finite tree automata. The family which we are interested in of course has much more structure, and we could equally well have phrased the characterisation in terms of finite tree automata with weak counters, that is, counters that can only be partially tested for zero. Such counter machines have been intensively studied over words in terms of Petri nets and vector addition systems but we do not know of any generalisation to trees.

References

[1] S. Abramsky. Eliminating local non-determinism: Semantics for ccs. Technical Report Report no. 290, Computer Systems Laboratory, Queen Mary College, 1981.

[2] L. Aceto. A static view of localities. *Formal Aspects of Computing*, 6(2):202–222, 1994.

[3] J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information and Control*, 60:109–137, 1984.

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

[5] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[6] S. Christensen. Distributed bisimilarity is decidable for a class of infinite-state systems. In W.R. Cleaveland, editor, *CONCUR 92*, pages 148–161. Springer-Verlag, 1992. Lecture Notes in Computer Science, Vol. 630.

[7] S. Christensen. *Decidability and Decomposition in Process Algebras.* PhD thesis, University of Edinburgh, 1993.

[8] S. Christensen and H. Hüttel. Decidability issues for infinite-state processes - a survey. *EATCS Bulletin*, 51:156–166, 1993.

[9] J. Esparza. Petri nets, commutative context-free grammars and basic parallel processes. In *Proceedings of Fundamentals of Computation Theory, (FCT'95)*. Springer-Verlag, Lecture notes vol. 965, 1995.

[10] Y Hirshfeld. Petri nets and the equivalence problem. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Computer Science Logic: 7th Workshop, CSL '93 Selected Papers*, pages 165–174. Springer-Verlag, 1994. Lecture Notes in Computer Science, Vol. 832.

[11] Y. Hirshfeld and F. Moller. Decidability results in automata and process theory. In G. Birtwistle and F. Moller, editors, *Proceedings of Logics for Concurrency: Automata vs Structure. The VIII Banff Higher Order Workshop*, Lecture Notes in Computer Science. Springer-Verlag, 1994. To appear.

[12] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[13] P. Jancar. High undecidability of weak bisimilarity for Petri nets. *Lecture Notes in Computer Science*, 915:349–363, 1995.

[14] P. Jancar, A. Kucera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. Technical report, Institut für Informatik, Technische Universitt München, 1998.

[15] L. Jategaonkar and A. Meyer. Deciding true concurrency equivalences on finite safe nets. In *ICALP '93*, pages 519–531. Springer-Verlag, 1993. Lecture Notes in Computer Science, Vol. 700.

[16] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31:697–718, 1994.

[17] A. Kiehn and M. Hennessy. On the decidability of non-interleaving process equivalences. In B. Jonsson and J. Parrow, editors, *Concur '94: Concurrency Theory 5th International conference Proceedings*. Springer-Verlag, 1994. Lecture Notes in Computer Science, Vol.836.

[18] R. Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proceedings of FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, volume 1180. Lecture Notes in Computer Science, Springer–Verlag, 1996.

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

[20] M.L. Minsky. *Computation - Finite and Infinite Machines*. Prentice Hall, 1967.

[21] E.R. Olderog and C.A.R. Hoare. Specification- oriented semantics for communicating processes. *Acta Informatica*, 23:9–66, 1986.

[22] V.R. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.

[23] W Reisig. *Petri Nets - an Introduction*. EATCS Monograph in Computer Science, Springer, 1985.

[24] C. Reutenauer. *Mathematics of Petri Nets*. Masson and Prentice-Hall, 1990.

[25] K. Sunesen and M. Nielsen. Behavioural equivalence for infinite systems – partially decidable! BRICS Report Series RS-95-55, Aarhus University, 1995.

[26] K. Sunesen and M. Nielsen. Behavioural equivalence for infinite systems — partially decidable! *Lecture Notes in Computer Science*, 1091:460–479, 1996.

[27] D. Taubner. *Finite Representations of CCS and CSP programs by Automata and Petri Nets*. Springer-Verlag, 1989. Lecture Notes in Computer Science, Vol. 369.

Recent BRICS Report Series Publications

RS-98-6 Kim Sunesen. *Further Results on Partial Order Equivalences on Infinite Systems*. March 1998. 48 pp.

RS-98-5 Olivier Danvy. *Formatting Strings in ML*. March 1998. 3 pp. This report is superseded by the later report BRICS RS-98-12.

RS-98-4 Mogens Nielsen and Thomas S. Hune. *Deciding Timed Bisimulation through Open Maps*. February 1998.

RS-98-3 Christian N. S. Pedersen, Rune B. Lyngsø, and Jotun Hein. *Comparison of Coding DNA*. January 1998. 20 pp. To appear in *Combinatorial Pattern Matching: 9th Annual Symposium*, CPM '98 Proceedings, LNCS, 1998.

RS-98-2 Olivier Danvy. *An Extensional Characterization of Lambda-Lifting and Lambda-Dropping*. January 1998.

RS-98-1 Olivier Danvy. *A Simple Solution to Type Specialization (Extended Abstract)*. January 1998. 7 pp.

RS-97-53 Olivier Danvy. *Online Type-Directed Partial Evaluation*. December 1997. 31 pp. Extended version of an article to appear in *Third Fuji International Symposium on Functional and Logic Programming*, FLOPS '98 Proceedings (Kyoto, Japan, April 2–4, 1998), pages 271–295, World Scientific, 1998.

RS-97-52 Paola Quaglia. *On the Finitary Characterization of π-Congruences*. December 1997. 59 pp.

RS-97-51 James McKinna and Robert Pollack. *Some Lambda Calculus and Type Theory Formalized*. December 1997. 43 pp.

RS-97-50 Ivan B. Damgård and Birgit Pfitzmann. *Sequential Iteration of Interactive Arguments and an Efficient Zero-Knowledge Argument for NP*. December 1997. 19 pp. To appear in *25th International Colloquium on Automata, Languages, and Programming*, ICALP '98 Proceedings, LNCS, 1998.

RS-97-49 Peter D. Mosses. *CASL for ASF+SDF Users*. December 1997. 22 pp. Appears in Sellink, Editor, *2nd International Workshop on the Theory and Practice of Algebraic Specifications, Electronic Workshops in Computing*, ASF+SDF '97 Proceedings, Springer-Verlag, 1997.