Information and Computation 207 (2009) 812–829

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Undecidable equivalences for basic parallel processes☆

Hans Hüttel [a,*], Naoki Kobayashi [b], Takashi Suto [b]

[a] Department of Computer Science, Selma Lagerlöfs Vej 300, 9220 Aalborg, Denmark
[b] Graduate School of Information Sciences (GSIS), Tohoku University, Aramaki aza Aoba 6-3-09, Aoba-ku Sendai-city Miyagi-pref. 980-8579, Japan

ARTICLE INFO

Article history:
Received 10 December 2007
Revised 18 November 2008
Available online 31 January 2009

ABSTRACT

The trace equivalence of BPP was shown to be undecidable by Hirshfeld. We show that all the preorders and equivalences except bisimulation in Glabbeek's linear time-branching time spectrum are undecidable for BPP. The results are obtained by extending Hirshfeld's encoding of Minsky machines into BPP. We also show that those preorders and equivalences are undecidable even for a restriction of BPP to 2-labels.

1. Introduction

1.1. The linear time-branching time spectrum

Much attention has been devoted to the study of process calculi and their behavioural semantics. A particular focus has been that of the study of the properties of behavioural equivalences and orderings, since these relations are widely used in the settings of program optimization and verification.

There are various criteria that can be used to compare the merits and deficiencies of behavioural equivalences. One important criterion is to classify equivalences according to their coarseness. The *linear time-branching time spectrum*, shown in Fig. 1, was identified by van Glabbeek in [19] and constitutes a classification of this kind. In the diagram, the coarsest, i.e., least discriminating, relations are at the bottom and arrows indicate inclusion. All equivalences below bisimilarity except n-bounded-tr-bisimulation are defined as the symmetric closure of a preorder, so it is not surprising that the same hierarchy exists for the corresponding preorders.

At the bottom of the diagram we find trace equivalence and completed trace equivalence (the language equivalence of classical automata theory). At the top of the diagram is bisimulation equivalence (or bisimilarity), introduced by Park [16] and subsequently widely used in process theories.

Note that n-bounded-tr-bisimulation actually constitutes an entire family of behavioural relations. 1-bounded-tr-bisimulation equivalence is trace equivalence and 2-bounded-tr-bisimulation corresponds to possible-futures equivalence.

1.2. Existing decidability results

Another relevant criterion for comparing behavioural relations is that of *decidability*.

Decidability problems for behavioural relations have been thoroughly studied for finite-state processes (i.e., finite automata) and Basic Process Algebra (BPA) [1]. For finite-state processes, all equivalences in Fig. 1 are decidable [11]. For BPA, the completed trace equivalence is undecidable (which follows from the well-known result that language equivalence is

☆ This is a revised version of [9] and [13].

* Corresponding author.
E-mail addresses: hans@cs.aau.dk (H. Hüttel), koba@ecei.tohoku.ac.jp (N. Kobayashi), tsuto@kb.ecei.tohoku.ac.jp (T. Suto).

0890-5401/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.12.011

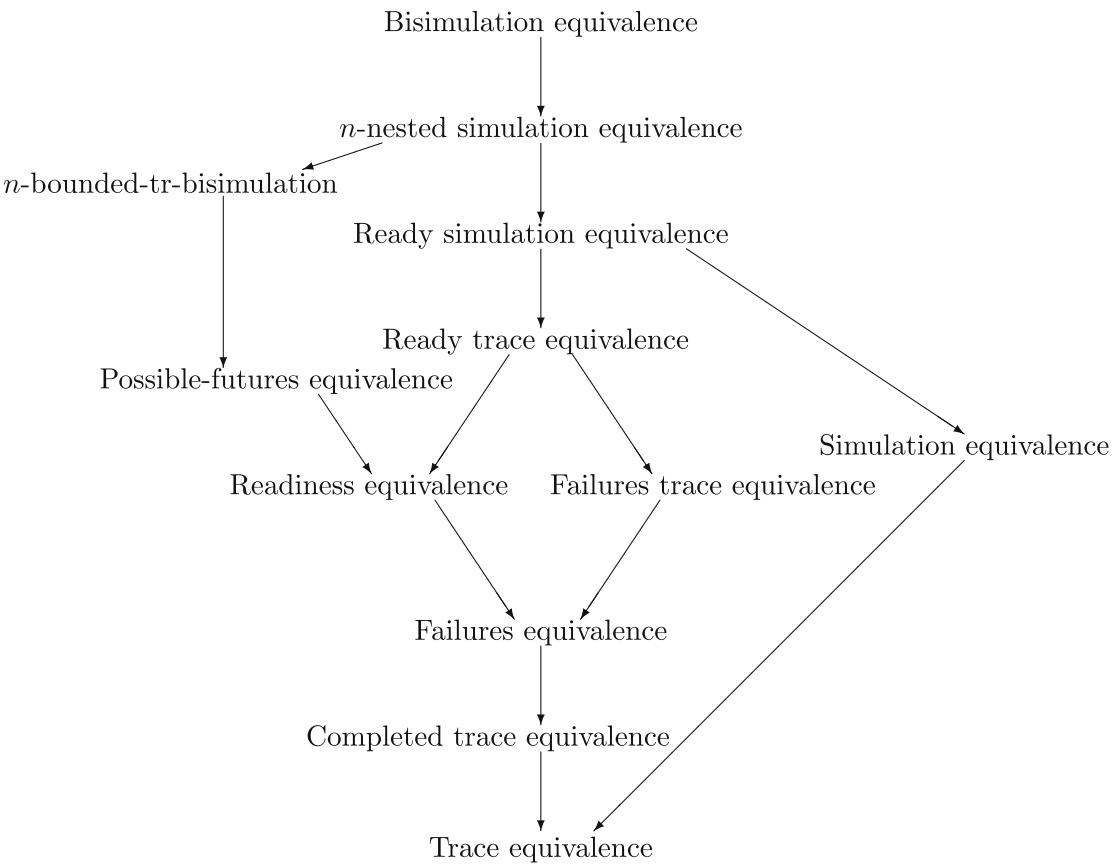

Fig. 1. The linear-time-branching time spectrum of behavioural relations.

undecidable for context-free grammars: see, e.g., [8]). On the other hand, strong bisimilarity is decidable for BPA processes [4]. Huynh and Tian [10] proved that readiness and failures equivalences are undecidable for BPA processes and, using this result, it was shown in [5] that in fact *none* of the equivalences below bisimilarity in Fig. 1 are decidable for normed BPA processes.

Attention has since focused on Basic Parallel Processes (BPP), another minimal process calculus where a non-communicating parallel (full merge) operator takes the place of sequencing. Decidability issues for BPP are of particular interest, since an undecidability result for this process calculus will imply undecidability of the corresponding problem for any concurrent process calculus that includes BPP as a sub-calculus. Decidability issues are important also because BPP has recently been used as the basis of a behavioural type system for the π-calculus [12,13], where types are expressed as BPP processes and the subtyping relation is a preorder for BPP.

A first, positive result was established by Christensen, Hirshfeld and Moller, who proved [3,2] that bisimulation equivalence is decidable for BPP. Hirshfeld [7] then showed that the trace equivalence is undecidable for BPP.

1.3. Contributions of this paper

In this paper, we show that *none* of the equivalences below bisimilarity and the corresponding preorders in the van Glabbeek hierarchy are decidable for BPP. This result first appeared in [9]; the present paper corrects a flaw in the reduction presented there.

We then strengthen our general undecidability result by showing that the undecidability results for the preorders between the trace preorder and the ready simulation hold even for BPP processes that only use two distinct labels. These strengthened results were first announced in [13].

A highlight of our results is the undecidability of all preorders between the trace preorder and ready simulation. This result is obtained by a reduction from the halting problem for two-counter Minsky machines. The encoding is based on an idea due to Hirshfeld [7]. Fig. 2 illustrates the idea of the proof. In the figure, $\leqslant_{tr}$ and $\leqslant_{ready}$ are the trace preorder and the ready simulation, and $\sim_{tr}$ and $\sim_{ready}$ are the corresponding equivalences. For any Minsky machine M, we construct two BPP

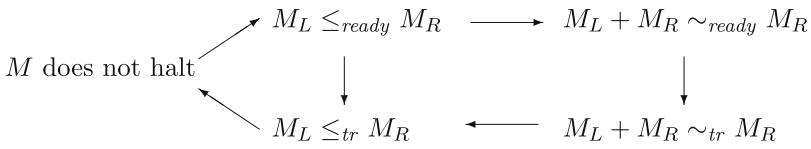

Fig. 2. The idea of the proof of the undecidability of preorders/equivalences between the ready simulation and trace preorders/equivalences for BPP.

processes M_L and M_R such that (i) $M_L \leqslant_{tr} M_R$ implies that M does not halt, and (ii) if M does not halt, $[\![M]\!]_L \leqslant_{ready} [\![M]\!]_R$ holds. Since $M_L \leqslant_{ready} M_R$ implies $M_L \leqslant_{tr} M_R$, it follows that for any preorder $\mathcal{R}$ between $\leqslant_{ready}$ and $\leqslant_{tr}$, $M_L \mathcal{R} M_R$ holds if and only if M does not halt. Thus, all the preorders between $\leqslant_{ready}$ and $\leqslant_{tr}$ are undecidable, since the halting problem of Minsky machines is undecidable. The undecidability results for equivalence relations also follow as a corollary, since $M_L \leqslant_{ready} M_R$ implies $M_L + M_R \sim_{ready} M_R$, and $M_L + M_R \sim_{tr} M_R$ implies $M_L \leqslant_{tr} M_R$ (see Fig. 2). The remaining preorders are covered by separate reductions. We first show that for any n, the simulation preorder problem can be reduced to that of n-nested simulation. Next, we show that the trace preorder problem can be reduced to that of n-bounded-tr-bisimulation preorder, and that the same applies to n-bounded-tr-bisimulation equivalence.

To show the undecidability for the 2-label case, we introduce two encodings $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_R$ of BPP processes into 2-label BPP processes such that (i) $P \leqslant_{ready} Q$ implies $[\![P]\!]_L \leqslant_{ready} [\![Q]\!]_R$, and (ii) $[\![P]\!]_L \leqslant_{tr} [\![Q]\!]_R$ implies $P \leqslant_{tr} Q$. From this and the relations shown in Fig. 2, it follows that for any preorder $\mathcal{R}$ between the trace preorder and the ready simulation on 2-label BPP, $[\![M_L]\!]_L \mathcal{R} [\![M_R]\!]_R$ if and only if M does not halt. Srba [18] proposed a general method for encoding a labeled transition system into a transition system with a *single* label, so that certain properties are preserved by the encoding. His encoding is, however, not applicable to BPP. We have therefore developed an entirely new encoding of BPP processes into 2-label processes for the scenario above.

The rest of the paper is organized as follows. Section 2 deals with the preliminaries: definitions of BPP processes, behavioural preorders and Minsky machines. In Section 3, we show that all the preorders between the trace preorder and the ready simulation are undecidable by a reduction from the halting problem for Minsky machines. In the remainder of the paper we consider the remaining equivalences/preorders, namely the families of n-nested simulations (Section 4) and n-bounded tr-bisimulations (Section 5), respectively. Finally, in Section 6 we strengthen the previously obtained results, showing that all the preorders and equivalences except bisimulation are undecidable even for 2-*label* BPP. Section 7 concludes the paper.

2. Preliminaries

2.1. BPP

The class of Basic Parallel Processes was first proposed by Christensen et al. [3]. The syntax of process expressions is given by:

$$P ::= \mathbf{0} \mid X \mid lP \mid (P|Q) \mid P + Q \mid \mu X.P$$

Here, X ranges over the set **Var** of *process variables* and l ranges over the set **Act** of *action labels*. We write $\mathbf{BPP}_{\mathbf{Act}}$ for the set of BPP processes whose action labels are in the set **Act**.

The process **0** does nothing. A process lP first performs l and then behaves like P. $P|Q$ is a parallel composition of P and Q. Note that there is no notion of communication in BPP, so our notion of parallel composition is the notion of full merge. Finally, $P + Q$ is an internal choice of P or Q. $\mu X.P$ stands for the recursive process X usually defined by the equation $X \stackrel{def}{=} P$. The variable X in $\mu X.P$ is bound by μX. We say that a process is *closed* if all the process variables are bound.

We often omit **0** and just write a for $a\mathbf{0}$. We give a higher-precedence to unary prefixes than to $+$ and $|$, so that $l_1 P_1 | l_2 P_2$ means $(l_1 P_1)|(l_2 P_2)$.

We say that P is *guarded* by l in Q if Q is of the form lQ' and P is a sub-process of Q'. A recursive process $\mu X.P$ is *guarded* if all occurrences of the variable X in P are guarded by some action label. A process is *guarded* if all its recursive processes are guarded. In the rest of this paper, we consider only closed, guarded processes. Note that any recursive process can be transformed to a bisimilar, guarded recursive process. For example, $\mu X.(X \mid lX)$ is equivalent to the guarded process $\mu X.l(X \mid X)$. $\mu X.X$ is bisimilar to **0**.

The transition relation $P \stackrel{l}{\longrightarrow} Q$ is the least relation closed under the rules in Fig. 3. We write $P \stackrel{l_1 \cdots l_n}{\longrightarrow} Q$ if $P \stackrel{l_1}{\longrightarrow} \cdots \stackrel{l_n}{\longrightarrow} Q$. The sequence $l_1 \cdots l_n$ may be empty, so that $P \stackrel{\epsilon}{\longrightarrow} P$ always holds (where ϵ denotes the empty sequence).

2-label BPP is BPP where the set **Act** of action labels is restricted to the set $\{a, b\}$. Hence, the set of 2-label BPP processes is $\mathbf{BPP}_{\{a,b\}}$.

$$\frac{}{lP \xrightarrow{l} P} \quad (\text{Tr-Act}) \qquad \frac{[\mu X.P/X]P \xrightarrow{l} Q}{\mu X.P \xrightarrow{l} Q} \quad (\text{Tr-Rec})$$

$$\frac{P \xrightarrow{l} P'}{P \mid Q \xrightarrow{l} P' \mid Q} \quad (\text{Tr-ParL}) \qquad \frac{Q \xrightarrow{l} Q'}{P \mid Q \xrightarrow{l} P \mid Q'} \quad (\text{Tr-ParR})$$

$$\frac{P \xrightarrow{l} P'}{P + Q \xrightarrow{l} P'} \quad (\text{Tr-OrL}) \qquad \frac{Q \xrightarrow{l} Q'}{P + Q \xrightarrow{l} Q'} \quad (\text{Tr-OrR})$$

Fig. 3. Transition rules of BPP processes.

2.2. Behavioural preorders

We use the following definition of the ready simulation preorder:

Definition 2.1. Let $readies(P) = \{l \in \mathbf{Act} | P \xrightarrow{l}\}$. A relation R between processes is a *ready simulation* iff whenever PRQ then

- If $P \xrightarrow{l} P'$ then there exists an Q' s.t. $Q \xrightarrow{l} Q'$ with $P'RQ'$.
- $readies(P) = readies(Q)$

We say that Q ready simulates P, written $P \leqslant_{ready} Q$, iff there is a ready simulation R with PRQ. Two processes P and Q are *ready simulation equivalent*, written $P \sim_{ready} Q$, iff $P \leqslant_{ready} Q$ and $Q \leqslant_{ready} P$.

So, to prove that $P \leqslant_{ready} Q$ we need only establish a ready simulation R such that PRQ.

Definition 2.2. The trace language $traces(P)$ of P is given by

$$traces(P) = \{w \in \mathbf{Act}^* | \exists Q.P \xrightarrow{w} Q\}$$

We write $P \leqslant_{tr} Q$ if $traces(P) \subseteq traces(Q)$. Two processes P and Q are *trace equivalent*, written $P \sim_{tr} Q$, iff $P \leqslant_{tr} Q$ and $Q \leqslant_{tr} P$.

2.3. Minsky machines

Minsky machines, first defined in [15], are very simple imperative programs that constitute a universal model of computation.

A Minsky machine has two counters, and consists of a sequence of the following instructions:

- Type 1 instruction: $c_i := c_i + 1$; **goto** k.
- Type 2 instruction: **if** $c_i = 0$ **then goto** k_1 **else** $c_i := c_i - 1$; **goto** k_2.
- Halt instruction.

Formally, Minsky machines can be defined as follows.

Definition 2.3. A Minsky machine M is a mapping from a finite set $\mathbf{Addr}_M$ of natural numbers to the set of instructions:

$$\{\mathbf{Inc}_{i,k} \mid i \in \{0,1\}, k \in \mathbf{Addr}_M \cup \{\bot\}\} \cup \{\mathbf{Cond}_{i,k_1,k_2} \mid i \in \{0,1\}, k_1, k_2 \in \mathbf{Addr}_M \cup \{\bot\}\}.$$

We require that $0 \in \mathbf{Addr}_M$. A *state* of a Minsky machine is a triple $\langle k, m_0, m_1\rangle$. The *initial state* σ_I of a Minsky machine is $\langle 0,0,0\rangle$. The transition relation $\xrightarrow{l}_M$ (where $l \in \{inc_i, then_i, else_i \mid i \in \{0,1\}\}$) is defined by:

$$\frac{M(k) = \mathbf{Inc}_{i,k'} \qquad m'_i = m_i + 1 \qquad m'_{1-i} = m_{1-i}}{\langle k, m_0, m_1\rangle \xrightarrow{inc_i}_M \langle k', m'_0, m'_1\rangle}$$

$$\frac{M(k) = \mathbf{Cond}_{i,k_1,k_2} \qquad m_i = 0}{\langle k, m_0, m_1\rangle \xrightarrow{then_i}_M \langle k_1, m_0, m_1\rangle}$$

$$\frac{M(k) = \mathbf{Cond}_{i,k_1,k_2} \qquad m_i > 0 \qquad m'_i = m_i - 1 \qquad m'_{1-i} = m_{1-i}}{\langle k, m_0, m_1\rangle \xrightarrow{else_i}_M \langle k_2, m'_0, m'_1\rangle}$$

We write $\sigma \overset{l_1 \cdots l_n}{\Longrightarrow}_M \sigma'$ if $\sigma \xrightarrow{l_1}_M \cdots \xrightarrow{l_n}_M \sigma'$ for some $l_1, \ldots, l_n$. We also write $\sigma \Longrightarrow_M \sigma'$ if $\sigma \Longrightarrow_M \sigma'$ for some s. A Minsky machine M *halts* if there is a reduction sequence $\sigma_I \Longrightarrow_M \langle \perp, m_1, m_2\rangle$ for some m_1, m_2.

The halting problem of Minsky machines is known to be undecidable [15]. In the rest of this paper, we omit M and write $\xrightarrow{l}$ and $\overset{s}{\Longrightarrow}$ for $\xrightarrow{l}_M$ and $\overset{s}{\Longrightarrow}_M$, respectively.

3. Undecidability of BPP preorders and equivalence relations

In this section, we show that all the preorders between the trace preorder and the ready simulation preorder on BPP (i.e., all the preorders $\leqslant$ such that $\leqslant_{ready} \subseteq \leqslant \subseteq \leqslant_{tr}$) and all the equivalences between the trace equivalence and the ready simulation equivalence are undecidable.

The idea of the proof is as follows: given the initial state σ_I of a Minsky machine M, we construct two BPP processes $[\![M]\!]_L$ and $[\![M]\!]_R$ that satisfy the following conditions:

- If M halts, then $[\![M]\!]_L \not\leqslant_{tr} [\![M]\!]_R$.
- If M does not halt, $[\![M]\!]_L \leqslant_{ready} [\![M]\!]_R$.

Then, it is easy to deduce that, for any preorder $\leqslant$ such that $\leqslant_{ready} \subseteq \leqslant \subseteq \leqslant_{tr}$, a Minsky machine M halts if and only if $[\![M]\!]_L \leqslant [\![M]\!]_R$ holds. Since it is undecidable whether M halts, the preorder $\leqslant$ is also undecidable.

The basic idea of encodings is the same as that of Hirshfeld [7] used for proving the undecidability of the trace preorder (except for some new tricks to equalize the ready sets of $[\![M]\!]_L$ and $[\![M]\!]_R$). A state $\langle j, m_0, m_1\rangle$ of a Minsky machine is represented by processes of the form $L_j \mid C_0^{m_0} \mid C_1^{m_1}$ and $\overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$, where P^m denotes the parallel composition of m copies of P. Basically, L_j and $\overline{L}_j$ are defined so that $\overline{L}_j$ can do more actions than L_j, except the case $j = \perp$: $L_\perp$ can do a special action w, while $\overline{L}_\perp$ cannot. Thus, $\overline{L}_j$ can simulate the behaviour of L_j as long as the Minsky machine does not halt.

We first give the definition of $[\![M]\!]_L$:

Definition 3.1. Let M be a Minsky machine. $[\![M]\!]_L$ is defined by:

$$\begin{aligned}
[\![M]\!]_L &= L_0 \\
C_i &= l_{d_i} \\
L_j &= \begin{cases} l_l(C_i \mid L_k) + U & \text{if } M(j) = \mathbf{Inc}_{i,k} \\ l_{\langle_i} T_{i,k_1} + l_{[} E_{i,k_2} + U & \text{if } M(j) = \mathbf{Cond}_{i,k_1,k_2} \\ wU + U & \text{if } j = \perp \end{cases} \\
T_{i,k} &= l_{\rangle_i} L_k + U \\
E_{i,k} &= l_{]_i} L_k + U \\
U &= \sum_{a \in \mathbf{Act}_0} aU' \\
U' &= \sum_{a \in \mathbf{Act}_1} aU'
\end{aligned}$$

Here, $\mathbf{Act}_0 = \{l_l, l_{d_0}, l_{d_1}, l_{\langle_0}, l_{\langle_1}, l_{\rangle_0}, l_{\rangle_1}, l_{[}, l_{]_0}, l_{]_1}, w\}$ and $\mathbf{Act}_1 = (\mathbf{Act}_0 \setminus \{w\}) \cup \{l_\bigstar\}$.

Intuitively, a transition of label inc_i of a Minsky machine is simulated by the action l_l. A transition of label $then_i$ is simulated by the action sequence $l_{\langle_i} l_{\rangle_i}$, and a transition of label $else_i$ is simulated by $l_{[} l_{d_i} l_{]_i}$. If $M(j) = \mathbf{Inc}_{i,k}$, L_j performs the action l_l and increments the value of the counter i (by spawning a fresh copy of C_i). The role of the process U is to force the ready set to be always $\mathbf{Act}_0$. In simulating the Minsky machine, U is not intended to be used for simulating the Minsky machine; if U is used, the special action $l_\bigstar$ would be put into the ready set.

If $M(j) = \mathbf{Cond}_{i,k_1,k_2}$, then L_j executes the then-branch and the else-branch in a *non-deterministic* manner. Thus, $[\![M]\!]_L$ may execute invalid instructions that the machine M cannot execute. Such invalid transitions are simulated by the special process G of $[\![M]\!]_R$ given below.

Definition 3.2. Let M be a Minsky machine. $[\![M]\!]_R$ is defined by:

$$
\begin{array}{lcl}
[\![M]\!]_R & = & \overline{L}_0 \\
\overline{C}_i & = & l_{]_i} + l_{\langle_i} \\
\overline{L}_j & = & \begin{cases} l_I(\overline{C}_i \mid \overline{L}_k) + \sum_{i\in\{0,1\}} l_{d_i} G + U & \text{if } M(j) = \mathbf{Inc}_{i,k} \\ l_{\langle_i}\overline{T}_{i,k_1} + l_{[}\overline{E}_{i,k_2} + \sum_{i\in\{0,1\}} l_{d_i} G + l_{\rangle_i} G + U & \text{if } M(j) = \mathbf{Cond}_{i,k_1,k_2} \\ \mathbf{0} & \text{if } j = \perp \end{cases} \\
\overline{T}_{i,k} & = & l_{\rangle_i}\overline{L}_k + l_{d_{1-i}} G + U \\
\overline{E}_{i,k} & = & l_{d_i}\overline{L}_k + l_{d_{1-i}} G + l_{]_i} G + U \\
G & = & \sum_{a\in\mathbf{Act}_0} aG + U
\end{array}
$$

The main differences of $[\![M]\!]_R$ from $[\![M]\!]_L$ are as follows.

- $\overline{L}_\perp$ cannot do any action, so that the process $\overline{L}_\perp \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ (which corresponds to the halting state of the Minsky machine) fails to simulate $L_\perp \mid C_0^{m_0} \mid C_1^{m_1}$.
- A process of the form lG is added to $\overline{L}_j$, where G is a kind of universal process that can simulate any behaviour. G is used to simulate transitions of $L_j \mid C_0^{m_0} \mid C_1^{m_1}$ that are invalid with respect to the Minsky machine. For example, in the state $L_j \mid C_0^{m_0} \mid C_1^{m_1}$ where $M(j) = \mathbf{Inc}_{i,k}$, the only valid action corresponding to an action of the Minsky machine is l_I, but $L_j \mid C_0^{m_0} \mid C_1^{m_1}$ can also perform an action l_{d_i} if $m_i > 0$. Such an action is simulated by the sub-process $l_{d_i} G$ of $\overline{L}_j$.
- The roles of $l_{]_i}$ and l_{d_i} in $\overline{C}_i$ and $E_{i,k}$ have been swapped: this is to take care of the case where $L_j \mid C_0^{m_0} \mid C_1^{m_1}$ performs invalid transitions $l_{[}l_{]_i}$ (recall that the valid transition sequence is $l_{[}l_{d_i}l_{]_i}$). In that case, $\overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ can simulate the transitions by $\overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1} \xrightarrow{l_{[}} \overline{E}_{j,k} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1} \xrightarrow{l_{]_i}} G \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$.
- $\overline{C}_i$ can also perform $l_{\langle_i}$ action. This is to take care of the case where $M(j)$ is $\mathbf{Cond}_{i,k_1,k_2}$, $m_i > 0$ but $L_j \mid C_0^{m_0} \mid C_1^{m_1}$ performs the actions $l_{\langle_i}l_{\rangle_i}$. That invalid transition sequence can be simulated by $\overline{L}_j \mid \overline{C}_i^{m_i} \mid \overline{C}_{1-i}^{m_{1-i}} \xrightarrow{l_{\langle_i}} \overline{L}_j \mid \overline{C}_i^{m_i-1} \mid \overline{C}_{1-i}^{m_{1-i}} \xrightarrow{l_{\rangle_i}} G \mid \overline{C}_i^{m_i-1} \mid \overline{C}_{1-i}^{m_{1-i}}$.

Remark 3.3. In Hirshfeld's encoding [7], a then-branch is modeled by a single action l_i (instead of the two actions $l_{\langle_i}l_{\rangle_i}$ in our case), and $\overline{C}_i$ is of the form $l_{]_i} + l_i G$. That encoding does not seem to work for the result below.

Now we state the main result.

Theorem 3.4.

1. *If* $\langle 0,0,0\rangle \Longrightarrow_M \langle \perp, m_0, m_1\rangle$, *then* $[\![M]\!]_L \not\leqslant_{tr} [\![M]\!]_R$.
2. *If* $\langle 0,0,0\rangle \not\Longrightarrow_M \langle \perp, m_0, m_1\rangle$, *then* $[\![M]\!]_L \leqslant_{ready} [\![M]\!]_R$.

Proof. See Section 3.1. □

Corollary 3.5. *If $\mathcal{R}$ is a preorder on BPP processes and $\leqslant_{ready} \subseteq \mathcal{R} \subseteq \leqslant_{tr}$, then the relation $\mathcal{R}$ is undecidable.*

Proof. Trivial from the fact that M does not halt if and only if $[\![M]\!]_L \mathcal{R} [\![M]\!]_R$. □

Corollary 3.6. *If $\mathcal{R}$ is an equivalence relation on BPP processes and $\sim_{ready} \subseteq \mathcal{R} \subseteq \sim_{tr}$, then the relation $\mathcal{R}$ is undecidable.*

Proof. It suffices to show that M does not halt if and only if $[\![M]\!]_L + [\![M]\!]_R \mathcal{R} [\![M]\!]_R$. If M halts, then $[\![M]\!]_L \not\leqslant_{tr} [\![M]\!]_R$, which implies that there is a sequence s such that $s \in traces([\![M]\!]_L)$ but $s \notin traces([\![M]\!]_R)$. Thus, $[\![M]\!]_L + [\![M]\!]_R \not\sim_{tr} [\![M]\!]_R$, which implies $\neg([\![M]\!]_L + [\![M]\!]_R \mathcal{R} [\![M]\!]_R)$.

If M does not halt, then $[\![M]\!]_L \leqslant_{ready} [\![M]\!]_R$. Let $\mathcal{R}_1$ be a ready simulation such that $([\![M]\!]_L, [\![M]\!]_R) \in \mathcal{R}_1$. Then, $\mathcal{R}_1 \cup \{([\![M]\!]_L + [\![M]\!]_R, [\![M]\!]_R)\} \cup \mathbf{Id}$ (where $\mathbf{Id}$ is the identity relation) is a ready simulation, which implies $[\![M]\!]_L + [\![M]\!]_R \leqslant_{ready} [\![M]\!]_R$. $[\![M]\!]_R \leqslant_{ready} [\![M]\!]_L + [\![M]\!]_R$ also holds, since $\{([\![M]\!]_R, [\![M]\!]_L + [\![M]\!]_R)\} \cup \mathbf{Id}$ is a ready simulation. Thus, we have $[\![M]\!]_L + [\![M]\!]_R \sim_{ready} [\![M]\!]_R$, which implies $[\![M]\!]_L + [\![M]\!]_R \mathcal{R} [\![M]\!]_R$. □

3.1. Proof of Theorem 3.4

The rest of this section is devoted to the proof of Theorem 3.4.

Definition 3.7. Let $\sigma = \langle j, m_0, m_1\rangle$ be a state of a Minsky machine M. The BPP processes $[\![\sigma]\!]_L$ and $[\![\sigma]\!]_R$ are defined by:

$$[\![\langle j,m,n\rangle]\!]_L = L_j \mid C_0^m \mid C_1^n$$
$$[\![\langle j,m,n\rangle]\!]_R = \overline{L}_j \mid \overline{C}_0^m \mid \overline{C}_1^n$$

Here, L_j, $\overline{L}_j$, C_i, and $\overline{C}_i$ are those defined in Definition 3.1.

Definition 3.8. Let λ be an element of $\{inc_i, then_i, else_i \mid i \in \{0,1\}\}$. Then, $[\![\lambda]\!]$ is defined by:

$$[\![inc_i]\!] = l_I \qquad [\![then_i]\!] = l_{\langle_i} l_{\rangle_i} \qquad [\![else_i]\!] = l_{[} l_{d_i} l_{]_i}$$

The following lemma means that $[\![\sigma]\!]_L$ can simulate transitions of the Minsky machine.

Lemma 3.9. *If* $\sigma \xrightarrow{\lambda} \sigma'$, *then* $[\![\sigma]\!]_L \xrightarrow{[\![\lambda]\!]} [\![\sigma']\!]_L$.

Proof. Trivial by the definition of $[\![\sigma]\!]_L$. □

The following lemma means that $[\![M]\!]_R$ can simulate a transition of the Minsky machine only either by either performing valid actions, or by using the universal process U.

Lemma 3.10. *If* $\sigma \xrightarrow{\lambda} \sigma'$ *and* $[\![\sigma]\!]_R \xrightarrow{[\![\lambda]\!]} P$, *then* $P = [\![\sigma']\!]_R$ *or* $P = U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ *for some* m_0 *and* m_1.

Proof. Let $\sigma = \langle j, m_0, m_1\rangle$ and $\sigma' = \langle j', m_0', m_1'\rangle$. Case analysis on λ.

- Case $\lambda = inc_i$. In this case, $M(j) = \mathbf{Inc}_{i,j'}$ and $[\![\sigma]\!]_R = \overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$, with $m_i' = m_i + 1$ and $m_{1-i}' = m_{1-i}$. By the definition of $\overline{L}_j$, P must be either $(\overline{C}_i \mid \overline{L}_{j'}) \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1} = [\![\sigma']\!]_R$ or $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$.
- Case $\lambda = then_i$. In this case, $M(j) = \mathbf{Cond}_{i,j',k}$ and $[\![\sigma]\!]_R = \overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$, with $m_i = m_i' = 0$ and $m_{1-i}' = m_{1-i}$. Suppose $[\![\sigma]\!]_R \xrightarrow{l_{\langle_i}} P_1 \xrightarrow{l_{\rangle_i}} P$. By the definition of $\overline{L}_j$, P_1 must be either $\overline{T}_{i,j'} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ or $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. In the latter case, P must be $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. In the former case, by the definition of $\overline{T}_{i,j'}$, P must be either $\overline{L}_{j'} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ or $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$.
- Case $\lambda = else_i$. In this case, $M(j) = \mathbf{Cond}_{i,k,j'}$ and $[\![\sigma]\!]_R = \overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$, with $m_i' = m_i - 1 \geqslant 0$ and $m_{1-i}' = m_{1-i}$. Suppose $[\![\sigma]\!]_R \xrightarrow{l_{[}} P_1 \xrightarrow{l_{d_i}} P_2 \xrightarrow{l_{]_i}} P$. By the definition of $\overline{L}_j$, P_1 must be either $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ or $\overline{E}_{i,j'} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. In the former case, P must be of the form $U' \mid \overline{C}_0^{m_0''} \mid \overline{C}_1^{m_1''}$ for some m_0'' and m_1''. In the latter case, P_2 must be either $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ or $\overline{L}_{j'} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. In the former case, P must be $U' \mid \overline{C}_0^{m_0''} \mid \overline{C}_1^{m_1''}$ for some m_0'' and m_1''. In the latter case, P must be either $U' \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ or $\overline{L}_{j'} \mid \overline{C}_0^{m_0'} \mid \overline{C}_1^{m_1'}$. □

We can now prove the first part of Theorem 3.4.

Lemma 3.11. *If* $\langle 0,0,0\rangle \stackrel{s}{\Longrightarrow} \langle \bot, m_0, m_1\rangle$, *then* $[\![M]\!]_L \not\leqslant_{tr} [\![M]\!]_R$.

Proof. Suppose $\langle 0,0,0\rangle \stackrel{s}{\Longrightarrow} \langle \bot, m_0, m_1\rangle$. By Lemma 3.9, we have $[\![s]\!]w \in traces([\![M]\!]_L)$. Suppose $[\![M]\!]_R \xrightarrow{[\![s]\!]} P$. Then, by Lemma 3.10, P must be $[\![\langle \bot, m_0, m_1\rangle]\!]_R = \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ or $U' \mid \overline{C}_0^{m_0'} \mid \overline{C}_1^{m_1'}$. In either case, $P \stackrel{w}{\not\longrightarrow}$. Therefore, $[\![s]\!]w \notin traces([\![M]\!]_R)$. □

To show the second part of Theorem 3.4, we use the following up-to technique.

Definition 3.12 (*ready simulation up to*)**.** A binary relation $\mathcal{R}$ is a *ready simulation up to* $\leqslant_{ready}$, if it satisfies the following conditions.

- If $P \mathcal{R} Q$ and $P \xrightarrow{l} P'$, then $Q \xrightarrow{l} Q'$ and $P' \leqslant_{ready} \mathcal{R} \leqslant_{ready} Q'$ for some Q'.
- If $P \mathcal{R} Q$, then $readies(P) = readies(Q)$.

Lemma 3.13. *If* $\mathcal{R}$ *is a ready simulation up to* $\leqslant_{ready}$, *then* $\mathcal{R} \subseteq \leqslant_{ready}$.

Proof. This follows from the fact that $\mathcal{R} \cup (\leqslant_{ready} \mathcal{R} \leqslant_{ready})$ is a ready simulation. □

The following lemma states that G is universal in the sense that it can simulate any (even invalid) state resulting from $[\![M]\!]_L$ (except for a process of the form $U' \mid C_0^{m_0} \mid C_1^{m_1}$).

Lemma 3.14 *(properties of G). The following conditions hold for any m_0, m_1, n_0, n_1.*

$$L_j \mid C_0^{m_0} \mid C_1^{m_1} \leqslant_{ready} G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1} \qquad T_{i,j} \mid C_0^{m_0} \mid C_1^{m_1} \leqslant_{ready} G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$$
$$E_{i,j} \mid C_0^{m_0} \mid C_1^{m_1} \leqslant_{ready} G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1} \qquad U \mid C_0^{m_0} \mid C_1^{m_1} \leqslant_{ready} G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$$
$$U' \mid C_0^{m_0} \mid C_1^{m_1} \leqslant_{ready} U' \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$$

Proof. It suffices to show that the following relation $\mathcal{R}$ is a ready simulation.

$$\begin{aligned}
\mathcal{R} = {} & \{(L_j \mid C_0^{m_0} \mid C_1^{m_1},\ G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}) \mid j \in \mathbf{Addr}_M \cup \{\bot\}, m_0, m_1, n_0, n_1 \geq 0\} \\
& \cup \{(T_{i,j} \mid C_0^{m_0} \mid C_1^{m_1},\ G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}) \mid i \in \{0,1\}, j \in \mathbf{Addr}_M \cup \{\bot\}, m_0, m_1, n_0, n_1 \geqslant 0\} \\
& \cup \{(E_{i,j} \mid C_0^{m_0} \mid C_1^{m_1},\ G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}) \mid i \in \{0,1\}, j \in \mathbf{Addr}_M \cup \{\bot\}, m_0, m_1, n_0, n_1 \geqslant 0\} \\
& \cup \{(U \mid C_0^{m_0} \mid C_1^{m_1},\ G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}) \mid m_0, m_1, n_0, n_1 \geq 0\} \\
& \cup \{(U' \mid C_0^{m_0} \mid C_1^{m_1},\ U' \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}) \mid m_0, m_1, n_0, n_1 \geq 0\}
\end{aligned}$$

Suppose that (P, Q) is in the $\mathcal{R}$. It is easy to check that $readies(P) = readies(Q)$: if (P, Q) is in the fifth set, then $readies(P) = readies(Q) = \mathbf{Act}_1$. Otherwise, $readies(P) = readies(Q) = \mathbf{Act}_0$.

It remains to show that $P \xrightarrow{l} P'$ implies that $Q \xrightarrow{l} Q'$ and $(P', Q') \in \mathcal{R}$ for some Q'. The case where the transition $P \xrightarrow{l} P'$ comes from an action of C_i is trivial. For other cases, we perform case analysis on P. We show only the main cases; the other cases are similar or trivial.

- Case $P = L_j \mid C_0^{m_0} \mid C_1^{m_1}$: In this case, we have one of the following conditions:
  - $l = l_I$ and $P' = L_k \mid C_0^{m_0'} \mid C_1^{m_1'}$
  - $l = l_{\langle i}$ and $P' = T_{i,k} \mid C_0^{m_0} \mid C_1^{m_1}$
  - $l = l_[$ and $P' = E_{i,k} \mid C_0^{m_0} \mid C_1^{m_1}$
  - $l = w$ and $P' = U \mid C_0^{m_0} \mid C_1^{m_1}$
  - $P' = U' \mid C_0^{m_0} \mid C_1^{m_1}$

  Let Q' be $U' \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$ in the last case, and $G \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$ in the other cases. Then, we have $Q \xrightarrow{l} Q'$ and $(P', Q') \in \mathcal{R}$ as required.
- Case $P = U \mid C_0^{m_0} \mid C_1^{m_1}$. In this case, $P' = U' \mid C_0^{m_0} \mid C_1^{m_1}$. Thus, the required result holds for $Q' = U' \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$.
- Case $P = U' \mid C_0^{m_0} \mid C_1^{m_1}$. In this case, $Q = U' \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$ and $P' = P$. The required result holds for $Q' = Q$.

We are ready to prove Theorem 3.4:

"Let M be a Minsky machine.

1. If $\langle 0,0,0\rangle \Longrightarrow_M \langle \bot, m_0, m_1\rangle$, then $[\![M]\!]_L \not\leqslant_{tr} [\![M]\!]_R$.
2. If $\langle 0,0,0\rangle \not\Longrightarrow_M \langle \bot, m_0, m_1\rangle$, then $[\![M]\!]_L \leqslant_{ready} [\![M]\!]_R$."

Proof of Theorem 3.4.

1. This has been proved in Lemma 3.11.
2. Suppose $\langle 0,0,0\rangle \not\Longrightarrow_M \langle \bot, m_0, m_1\rangle$. Let $\mathcal{R}$ be:

$$\begin{aligned}
& \mathbf{Id} \\
\cup\ & \{([\![\sigma]\!]_L,\ [\![\sigma]\!]_R) \mid \sigma_I \Longrightarrow_M \sigma\} \\
\cup\ & \{(T_{i,k_1} \mid C_0^{m_0} \mid C_1^{m_1}, \overline{T}_{i,k_1} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}) \\
& \qquad \mid \sigma_I \Longrightarrow_M \langle j, m_0, m_1\rangle \wedge M(j) = \mathbf{Cond}_{i,k_1,k_2} \wedge m_i = 0\} \\
\cup\ & \{(T_{i,k_1} \mid C_0^{m_0} \mid C_1^{m_1}, \overline{L}_j \mid \overline{C}_0^{m_0'} \mid \overline{C}_1^{m_1'}) \\
& \qquad \mid \sigma_I \Longrightarrow_M \langle j, m_0, m_1\rangle \wedge M(j) = \mathbf{Cond}_{i,k_1,k_2} \wedge m_i' = m_i - 1 \wedge m_{1-i}' = m_{1-i}\} \\
\\
\cup\ & \{(E_{i,k_2} \mid C_0^{m_0} \mid C_1^{m_1}, \overline{E}_{i,k_2} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}) \\
& \qquad \mid \sigma_I \Longrightarrow_M \langle j, m_0, m_1\rangle \wedge M(j) = \mathbf{Cond}_{i,k_1,k_2}\} \\
\cup\ & \{(E_{i,k_2} \mid C_0^{m_0} \mid C_1^{m_1}, \overline{L}_{k_2} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}) \\
& \qquad \mid \sigma_I \Longrightarrow_M \langle j, m_0, m_1\rangle \wedge M(j) = \mathbf{Cond}_{i,k_1,k_2} \wedge m_i' = m_i - 1 \wedge m_{1-i}' = m_{1-i}\}
\end{aligned}$$

Due to Lemma 3.13, it suffices to show that $\mathcal{R}$ is a ready simulation up to $\leqslant_{ready}$. Suppose (P, Q) is in $\mathcal{R}$. We can immediately obtain $readies(P) = readies(Q)$ (note that if $(P, Q) \notin \mathbf{Id}$, then $readies(P) = readies(Q) = \mathbf{Act}_0$).

It remains to show that $P \xrightarrow{l} P'$ implies that $Q \xrightarrow{l} Q'$ and $P' \leqslant_{ready} \mathcal{R} \leqslant_{ready} Q'$ for some Q'. We perform case analysis on (P, Q). The case where $(P, Q) \in \mathbf{Id}$ is trivial. In the other cases, if U is involved in the transition $P \xrightarrow{l} P'$, then P' must be of the form $U' \mid C_0^{m_0} \mid C_1^{m_1}$. By Lemma 3.14, the result holds for $Q' = U' \mid \overline{C}_0^{n_0} \mid \overline{C}_1^{n_1}$. Suppose that U is not involved in the transition $P \xrightarrow{l} P'$.

- Case where (P, Q) is in the second set. In this case, we have $P = L_j \mid C_0^{m_0} \mid C_1^{m_1}$ and $Q = \overline{L}_j \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$, with $\sigma_I \Longrightarrow_M \langle j, m_0, m_1 \rangle$ and $j \neq \perp$. There are four cases to consider.
  - Case $l = l_i$: In this case, $M(j) = \mathbf{Inc}_{i,k}$ and $P' = L_k \mid C_0^{m'_0} \mid C_1^{m'_1}$ with $m'_i = m_i + 1$ and $m'_{1-i} = m_{1-i}$. The required result holds for $Q' = \overline{L}_k \mid \overline{C}_0^{m'_0} \mid \overline{C}_1^{m'_1}$.
  - Case $l = l_{\langle i}$: In this case, $M(j) = \mathbf{Cond}_{i,k_1,k_2}$ and $P' = T_{i,k_1} \mid C_0^{m_0} \mid C_1^{m_1}$. Let Q' be $\overline{T}_{i,k_1} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$ if $m_i = 0$, and be $\overline{L}_j \mid \overline{C}_0^{m'_0} \mid \overline{C}_1^{m'_1}$ where $m'_i = m_i - 1$ and $m'_{1-i} = m_{1-i}$. Then, the result follows.
  - Case $l = l_{\rangle}$: In this case, $M(j) = \mathbf{Cond}_{i,k_1,k_2}$ and $P' = E_{i,k_2} \mid C_0^{m_0} \mid C_1^{m_1}$. Let Q' be $\overline{E}_{i,k_2} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. Then, the result follows, since (P', Q') is in the fifth set of $\mathcal{R}$.
  - Case $l = l_{d_i}$: In this case, $P' = L_j \mid C_0^{m'_0} \mid C_1^{m'_1}$ with $m'_i = m_i - 1$ and $m'_{1-i} = m_{1-i}$. Let $Q' = G \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. Then, $Q \xrightarrow{l} Q'$ and $P' \leqslant_{ready} Q'$. Since $\mathbf{Id} \subseteq \mathcal{R}$ and $\leqslant_{ready}$ is reflexive, we have $P' \leqslant_{ready} \mathcal{R} \leqslant_{ready} Q'$ as required.
- Case where (P, Q) is in the third set: In this case, we have $P = T_{i,k_1} \mid C_0^{m_0} \mid C_1^{m_1}$ and $Q = \overline{T}_{i,k_1} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$, with $\sigma_I \Longrightarrow_M \langle j, m_0, m_1 \rangle$, $M(j) = \mathbf{Cond}_{i,k_1,k_2}$, and $m_i = 0$. There are two cases to consider.
  - Case $l = l_{\rangle_i}$. In this case, $P' = L_{k_1} \mid C_0^{m_0} \mid C_1^{m_1}$. The required result holds for $Q' = \overline{L}_{k_1} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$.
  - Case $l = l_{d_{1-i}}$. In this case, $P' = T_{i,k_1} \mid C_0^{m'_0} \mid C_1^{m'_1}$ with $m'_i = 0$ and $m'_{1-i} = m_{1-i} - 1$. Let Q' be $G \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. Then, the result follows.
- Case where (P, Q) is in the fourth set: In this case, we have $P = T_{i,k_1} \mid C_0^{m_0} \mid C_1^{m_1}$ and $Q = \overline{L}_j \mid \overline{C}_0^{m'_0} \mid \overline{C}_1^{m'_1}$ with $M(j) = \mathbf{Cond}_{i,k_1,k_2}$, $m'_i = m_i - 1$ and $m'_{1-i} = m_{1-i}$. l must be either $l_{\rangle_i}$ or $l_{d_{i'}}$. Let Q' be $G \mid \overline{C}_0^{m'_0} \mid \overline{C}_1^{m'_1}$. Then, the result follows, since $P' \leqslant_{ready} Q'$ by Lemma 3.14.
- Case where (P, Q) is in the fifth set: In this case, we have:

$$\begin{array}{ll} P = E_{i,k_2} \mid C_0^{m_0} \mid C_1^{m_1} & Q = \overline{E}_{i,k_2} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1} \\ \sigma_I \Longrightarrow_M \langle j, m_0, m_1 \rangle & M(j) = \mathbf{Cond}_{i,k_1,k_2} \end{array}$$

  There are two cases to consider.
  - Case $l = l_{d_i}$: Then, $P' = E_{i,k_2} \mid C_0^{m'_0} \mid C_1^{m'_1}$ with $m'_i = m_i - 1$ and $m'_{1-i} = m_{1-i}$. The required result holds for $Q' = \overline{L}_{k_2} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$.
  - Case l is $l_{d_{1-i}}$ or $l_{\rangle_i}$. The required result holds for $Q' = G \mid C_0^{m_0} \mid C_1^{m_1}$.
- Case where (P, Q) is in the sixth set: In this case, we have:

$$\begin{array}{ll} P = E_{i,k_2} \mid C_0^{m'_0} \mid C_1^{m'_1} & Q = \overline{L}_{k_2} \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1} \\ \sigma_I \Longrightarrow_M \langle j, m_0, m_1 \rangle & M(j) = \mathbf{Cond}_{i,k_1,k_2} \\ m'_i = m_i - 1 & m'_{1-i} = m_{1-i} \end{array}$$

  There are two cases to consider.
  - Case $l = l_{\rangle_i}$. In this case, $P' = L_{k_2} \mid C_0^{m'_0} \mid C_1^{m'_1}$. The required result holds for $Q' = \overline{L}_{k_2} \mid \overline{C}_0^{m'_0} \mid \overline{C}_1^{m'_1}$.
  - Case $l = l_{d_{i'}}$. In this case, the required result holds for $Q' = G \mid \overline{C}_0^{m_0} \mid \overline{C}_1^{m_1}$. □

4. n-nested simulation equivalence

This section shows that the n-nested simulation preorder/equivalence are undecidable for every $n > 0$. The notion of 2-nested simulation equivalence, or rather the more general notion of n-nested simulation equivalence, was introduced

by Groote and Vaandrager [6] in their study of the *tyft/tyxt*-format for structured operational semantics because 2-nested simulation equivalence is the completed trace congruence for this format.

Definition 4.1. For all $n \in N$, *n-nested simulation*, written $\sqsubseteq^n$, is inductively defined by

- $E \sqsubseteq^0 F$ for all processes E and F,
- $E \sqsubseteq^{n+1} F$ iff there is a simulation relation $R \subseteq (\sqsubseteq^n)^{-1}$ with *ERF*.

Two processes E and F are *n-nested simulation equivalent*, written $E \leftrightarrows^n F$, iff $E \sqsubseteq^n F$ and $F \sqsubseteq^n E$.

Note that 1-nested simulation is just simulation and that therefore 1-nested simulation equivalence is simulation equivalence.

The proof that follows is essentially that of [5] and as the details are very similar, they have been omitted.

Lemma 4.2. *For all $n \in N$, n-nested simulation is preserved by action prefixing and $+$.*

The class of processes defined in the following (and first used in [5]) can be used to reduce simulation to both n-nested simulation and n-nested simulation equivalence.

Definition 4.3. Let E and F be processes and let a be an action. The processes $r^n(E,F)$ and $s^n(E,F)$ for $n > 0$ are inductively defined by:

$$r^1(E,F) = E + F, \qquad s^1(E,F) = E,$$

$$r^{n+1}(E,F) = a\,r^n(E,F) + a\,s^n(E,F), \qquad s^{n+1}(E,F) = a\,r^n(E,F).$$

Observe that if E and F are BPP processes, then so are $r^n(E,F)$ and $s^n(E,F)$.

Lemma 4.4. *Let E and F be processes. For all $n > 0$ it holds that*

1. $s^n(E,F) \sqsubseteq^n r^n(E,F)$,
2. $r^n(E,F) \sqsubseteq^n s^n(E,F)$ *iff* $F \sqsubseteq E$.

Proof. Induction in n. See [5] for details. □

Theorem 4.5. *For $n > 0$, n-nested simulation and n-nested simulation equivalence are undecidable for BPP processes.*

Proof. We reduce simulation to n-nested simulation using the following observation:

$$F \sqsubseteq E \quad \text{iff} \quad r^n(E,F) \sqsubseteq^n s^n(E,F).$$

We reduce simulation to n-nested simulation equivalence using the following observation:

$$F \sqsubseteq E \quad \text{iff} \quad r^n(E,F) \leftrightarrows^n s^n(E,F).$$

Since $n > 0$, both facts follow directly from Lemma 4.4. As simulation is undecidable, n-nested simulation and n-nested simulation equivalence must be undecidable. □

An interesting fact is that the limit of the n-nested simulation equivalences for $n \to \omega$ is strong bisimulation equivalence:

Theorem 4.6 [6]**.** *For any finitely branching labelled transition graph we have*

$$\sim \; = \bigcap_{n=0}^{\omega} \leftrightarrows^n$$

So we here have the odd situation, because of Theorem 4.6 and the result of [3], that while $\sim$ is decidable for BPP, it is the limit of a series of *undecidable* approximations.

5. n-bounded-tr-bisimulation

We now consider 2-bounded-tr-bisimulation or rather its generalization to n-bounded-tr-bisimulation. Here, we deal with the preorder and the equivalence separately.

5.1. The preorder

Here, we are dealing with a family of preorders that constitute a generalization of the trace preorder.

Definition 5.1. The *n-bounded-tr-bisimulation* preorder, written $\leqslant^n_{tr}$, is defined inductively as follows.

- $E \leqslant^0_{tr} F$ for all processes E and F.
- $E \leqslant^{n+1}_{tr} F$ iff whenever $E \xrightarrow{w} E'$ then $\exists F'$ such that $F \xrightarrow{w} F'$ and $E' \leqslant^n_{tr} F'$.

From this definition, it is obvious that $\leqslant^1_{tr}$ is just the trace preorder.
We now show that all non-trivial preorders in the family are undecidable. Here, the following lemma is essential.

Lemma 5.2. *Let E and F be processes. For all $n > 0$ it holds that*

$$E \leqslant^n_{tr} F \text{ iff } a\,E + a\,F \leqslant^{n+1}_{tr} a\,F$$

Theorem 5.3. *$\leqslant^n_{tr}$ is undecidable for all $n > 0$.*

Proof. Induction in n. The base case, $n = 1$, follows from Theorem 3.4, since $\leqslant^1_{tr}$ is the trace inclusion preorder. The induction step follows from Lemma 5.2. □

5.2. The equivalence

The equivalence we are dealing with here is a generalisation of *trace equivalence* and *possible futures equivalence*, in the sense that 1-bounded-tr-bisimulation corresponds to *trace equivalence* and 2-bounded-tr-bisimulation is the *possible futures equivalence* of [17].

Definition 5.4. *n-bounded-tr-bisimulation*, written $\sim^n_{tr}$, is defined inductively as follows:

- $E \sim^0_{tr} F$ for all processes E and F,
- $E \sim^{n+1}_{tr} F$ iff
  - if $E \xrightarrow{w} E'$ then $\exists F'$ such that $F \xrightarrow{w} F'$ and $E' \sim^n_{tr} F'$ and
  - if $F \xrightarrow{w} F'$ then $\exists E'$ such that $E \xrightarrow{w} E'$ and $E' \sim^n_{tr} F'$.

This notion of equivalence also arises naturally as the consecutive approximations of bisimulation equivalence [14]. For finitely branching transition graphs (and thus for BPP processes) the limit of the n-bounded-tr-bisimulations for $n \to \omega$ coincides with bisimulation equivalence:

Theorem 5.5 [14]**.** *For any finitely branching labelled transition graph we have*

$$\sim \; = \bigcap_{n=0}^{\omega} \sim^n_{tr}$$

In the following, we use the same reduction that was employed in [11] to show that n-bounded tr-bisimulation for regular processes is PSPACE-complete. The two lemmas given below are crucial.

Lemma 5.6 [11]**.**

$$E \sim^n_{tr} F \text{ iff } E + F \sim^n_{tr} E \text{ and } E + F \sim^n_{tr} F$$

Lemma 5.7 [11]**.** *Let E and F be processes. For all $n > 0$ it holds that*

$$E \sim^n_{tr} F \text{ iff } a\,(E + F) \sim^{n+1}_{tr} a\,E + a\,F$$

Theorem 5.8. *For $n > 0$ n-bounded-tr-bisimulation is undecidable for BPP processes.*

Proof. Induction in n. The base case $n = 1$ follows from the fact that 1-bounded-tr-bisimulation is trace equivalence, which is undecidable. For the induction step we reduce n-bounded-tr-bisimulation to $n + 1$-bounded-tr-bisimulation by applying Lemma 5.7. □

The consequence of the above result is again the rather odd one that $\sim$is decidable while none of these non-trivial approximations are!

6. Undecidability of 2-label BPP preorders and equivalences

In this section, we strengthen the results of Sections 3–5, showing that all the preorders and equivalences except bisimulation in Glabbeek's linear time-branching time spectrum are undecidable even for *2-label* BPP, a restriction of BPP where the number of labels are restricted to two.

The key theorem in Section 3 was Theorem 3.4, which says that for any Minsky machine M, there exist BPP processes P_M and Q_M such that:

- If M halts, then $P_M \not\leqslant_{tr} Q_M$.
- If M does not halt, then $P_M \leqslant_{ready} Q_M$.

We shall show that there exist *2-label* processes P_M and Q_M that satisfy the above conditions, from which it follows that all the 2-label BPP preorders between the trace preorder and the ready simulation are undecidable. By Theorem 3.4, it suffices to find two encodings $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_R$ from general BPP into 2-label BPP, such that the following conditions are satisfied for any BPP processes P and Q.

- If $P \not\leqslant_{tr} Q$, then $[\![P]\!]_L \not\leqslant_{tr} [\![Q]\!]_R$.
- If $P \leqslant_{ready} Q$, then $[\![P]\!]_L \leqslant_{ready} [\![Q]\!]_R$.

The rest of this section is structured as follows. Section 6.1 defines the encodings $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_R$. Section 6.2 shows that the encodings satisfy the above properties. From those properties, Section 6.3 derives undecidability results for 2-label BPP.

6.1. Encodings

In the encodings of general BPP processes into 2-label BPP processes, each action of label l is simulated by a sequence of labels consisting of the two labels a, b. Before giving our encodings $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_R$, we discuss why the encodings are not so trivial. Let us consider the label set $\{l_0, l_1, l_2\}$, and a naive encoding of them: $[\![l_0]\!] = a, [\![l_1]\!] = ba, [\![l_2]\!] = bb$. Let us define the process encoding $[\![P]\!]$ as the homomorphism induced by the label encoding. Then, the encoding fails to preserve $\not\leqslant_{tr}$ and $\leqslant_{ready}$, as shown by the following counterexamples.

- Let P and Q be $(l_1 \mid l_1) + l_2 l_0 l_0$ and $l_1 \mid l_1$, respectively.
  Then, $P \not\leqslant_{tr} Q$ (as $l_2 l_0 l_0$ is not a trace of Q). However, $[\![P]\!] = (ba \mid ba) + bbaa$ and $[\![Q]\!] = ba \mid ba$ satisfy $[\![P]\!] \leqslant_{tr} [\![Q]\!]$.
- Let P and Q be $l_1 \mid l_1$ and $l_1 l_1$ respectively. Then, $P \leqslant_{ready} Q$ but $[\![P]\!] \leqslant_{ready} [\![Q]\!]$ does not hold, since there is no transition of Q that corresponds to $[\![P]\!] = ba \mid ba \xrightarrow{b} \xrightarrow{b}$.

In the first counterexample, an interleaving execution of two encoded actions is confused with an execution of a single encoded action. In the second counterexample, an interleaving execution of two encoded actions reveals whether the two actions occur concurrently (as in $l_1 \mid l_1$) or sequentially (as in $l_1 l_1$).

We now introduce our encodings. The first problem mentioned above can be avoided by choosing a specific encoding of labels. The second problem can be avoided by augmenting $[\![\cdot]\!]_R$ with processes that can simulate any interleaving execution of encoded labels. In the rest of this section, we use meta-variables $P, Q, \ldots$ for processes in $\mathbf{BPP}_{\{l_0,\ldots,l_{N-1}\}}$, and use meta-variables $E, F, \ldots$ for processes in $\mathbf{BPP}_{\{a,b\}}$.

We first define encoding of labels. Since the number of labels occurring in a given process is finite, we assume here that the set **Act** of action labels is a finite set $\{l_0, \ldots, l_{N-1}\}$.

Definition 6.1. A mapping $[\![\cdot]\!]$ from **Act** to $\{a, b\}^*$ is defined by:

$$[\![l_i]\!] = ab^i ab^{2N-1-i}$$

Here, b^i stands for the sequence of b of length i. For example, $b^3 = bbb$.

For instance, let N be 3. Then, l_0, l_1, l_2 would be encoded as:

$$[\![l_0]\!] = aabbbbb \qquad [\![l_1]\!] = ababbbb \qquad [\![l_2]\!] = abbabbb$$

Note that the first problem mentioned above does not occur for this encoding. Suppose that s_1 and s_2 are prefixes of $[\![l_i]\!]$ and $[\![l_j]\!]$, and that $[\![l_k]\!]$ is a shuffle of s_1 and s_2. If neither s_1 nor s_2 is empty, then s_1 and s_2 must be of the form ab^{k_1} and ab^{k_2} with

$k_1 + k_2 = 2N - 1$. However, by the definition of the encodings, it must be the case that $k_1 \leqslant N - 1$ and $k_2 \leqslant N - 1$, which contradict with $k_1 + k_2 = 2N - 1$. Thus, either s_1 or s_2 must be empty, so that $i = k \wedge s_1 = [\![l_k]\!]$, or $j = k \wedge s_2 = [\![l_k]\!]$.

We now give encodings $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_R$ for processes.

Definition 6.2. A mapping from $\mathbf{BPP}_{\{l_0,\ldots,l_{N-1}\}}$ to $\mathbf{BPP}_{\{a,b\}}$ is defined by:

$$\begin{array}{rcl} [\![P]\!]_L & = & [\![P]\!]_{L'} \mid A \\ A & = & \mu X.aX \end{array}$$

$$\begin{array}{rcl} [\![\mathbf{0}]\!]_{L'} & = & \mathbf{0} \\ [\![X]\!]_{L'} & = & X \\ [\![lP]\!]_{L'} & = & [\![P]\!]_{L'}^{[\![l]\!]} \\ [\![P \mid Q]\!]_{L'} & = & [\![P]\!]_{L'} \mid [\![Q]\!]_{L'} \\ [\![P + Q]\!]_{L'} & = & [\![P]\!]_{L'} + [\![Q]\!]_{L'} \\ [\![\mu X.P]\!]_{L'} & = & \mu X.[\![P]\!]_{L'} \end{array} \qquad \begin{array}{rcl} [\![P]\!]_{L'}^{\epsilon} & = & [\![P]\!]_{L'} \\ [\![P]\!]_{L'}^{as} & = & a[\![P]\!]_{L'}^{s} \\ [\![P]\!]_{L'}^{bs} & = & b[\![P]\!]_{L'}^{s} \end{array}$$

$[\![\cdot]\!]_{L'}$ is the homomorphism defined by $[\![lP]\!]_{L'} = [\![l]\!][\![P]\!]_{L'}$. The process $[\![P]\!]_{L'}^{s}$ represents an intermediate state for the transition $[\![lP]\!]_{L'} \xrightarrow{[\![l]\!]} [\![P]\!]_{L'}$, where s is the remaining sequence of actions to complete $[\![l]\!]$-transitions. The role of A running in parallel with $[\![\cdot]\!]_{L'}$ is to adjust the ready set (so that $[\![P]\!]_L$ and its derivatives always have a in their ready sets).

We now give the encoding $[\![\cdot]\!]_R$ for the righthand side process. $[\![P]\!]_{L'}^{s}$ is replaced by $[\![P]\!]_{R'}^{s}$, so that any interleaving transition sequence can be simulated by $[\![P]\!]_{R'}^{s}$.

Definition 6.3. A mapping from $\mathbf{BPP}_{\{l_0,\ldots,l_{N-1}\}}$ to $\mathbf{BPP}_{\{a,b\}}$ is defined by:

$$[\![P]\!]_R = [\![P]\!]_{R'} \mid A$$

$$\begin{array}{rcl} [\![\mathbf{0}]\!]_{R'} & = & \mathbf{0} \\ [\![X]\!]_{R'} & = & X \\ [\![lP]\!]_{R'} & = & [\![P]\!]_{R'}^{[\![l]\!]} \\ [\![P \mid Q]\!]_{R'} & = & [\![P]\!]_{R'} \mid [\![Q]\!]_{R'} \\ [\![P + Q]\!]_{R'} & = & [\![P]\!]_{R'} + [\![Q]\!]_{R'} \\ [\![\mu X.P]\!]_{R'} & = & \mu X.[\![P]\!]_{R'} \end{array} \qquad \begin{array}{rcl} [\![P]\!]_{R'}^{\epsilon} & = & [\![P]\!]_{R'} \\ [\![P]\!]_{R'}^{as} & = & \begin{cases} a[\![P]\!]_{R'}^{s} & (\text{if } \#a(s) = 1) \\ a[\![P]\!]_{R'}^{s} + aH^{(\#b(s)-1)} + aG_1 & (\text{if } \#a(s) = 0) \end{cases} \\ [\![P]\!]_{R'}^{bs} & = & \begin{cases} b[\![P]\!]_{R'}^{s} + aH^{(\#b(s))} + aG_1 & (\text{if } \#a(s) = 1) \\ b[\![P]\!]_{R'}^{s} + aG_1 + aG_2 & (\text{if } \#a(s) = 0) \end{cases} \\ H^{(k)} & = & \begin{cases} bG_1 + aG_1 + aG_2 & (\text{if } k = 1) \\ bH^{(k-1)} + bG_1 + aG_1 + aG_2 & (\text{if } k > 1) \end{cases} \\ G_1 & = & aG_1 + aG_2 \\ G_2 & = & aG_1 + aG_2 + bG_1 + bG_2 \end{array}$$

Here, $\#a(s)$ ($\#b(s)$, respectively) denotes the number of occurences of a (b, respectively) in s.

In the definition of $[\![P]\!]_{R'}^{s}$, the role of summands such as $aH^{(\#b(s))} + aG_1$ is to simulate invalid transition sequences (caused by interleaving execution of multiple encoded actions). G_1 is an *almighty* process that can simulate any process, as long as the initial ready set of the process does not contain b. Similarly, G_2 can simulate any process, as long as the initial ready set of the process contains b. $H^{(k)}$ is a process that can simulate at most k initial b-actions, and then become an almighty process G_1 or G_2.

Example 6.4. Let $N = 3$, $P \stackrel{def}{=} l_1 \mid l_1$ and $Q \stackrel{def}{=} l_1 l_1$. $[\![P]\!]_L$ and $[\![Q]\!]_R$ are given by:

$$[\![P]\!]_L = ababbbb \mid ababbbb \mid A$$
$$[\![Q]\!]_R = (a(b(a(b(b[\![l_1]\!]_{R'}^{bb} + aG_1 + aG_2) + aG_1 + aG_2) + aH^{(3)} + aG_1) + aH^{(4)} + aG_1)) \mid A$$

The valid transition sequence $[\![P]\!]_L \xrightarrow{abab^4} \xrightarrow{abab^4} \mathbf{0} \mid \mathbf{0} \mid A$ can be simulated by $[\![Q]\!]_R \xrightarrow{abab^4} [\![l_1]\!]_R \mid A \xrightarrow{abab^4} \mathbf{0} \mid A$. On the other hand, an invalid transition sequence $[\![P]\!]_L \xrightarrow{ab} [\![0]\!]_{L'}^{ab^4} \mid [\![l_1]\!]_L \mid A \xrightarrow{a} [\![0]\!]_{L'}^{ab^4} \mid [\![0]\!]_{L'}^{bab^4} \mid A$ is simulated by $[\![Q]\!]_R \xrightarrow{ab} [\![l_1]\!]_{R'}^{ab^4} \mid A \xrightarrow{a} H^{(3)} \mid A$. □

6.2. Properties of the encodings

As mentioned in the beginning of this section, to derive the undecidability of preorders between the trace preorder and the ready simulation for 2-label BPP, it suffices to prove the following properties of the encodings.

Theorem 6.5. *Let P and Q be BPP processes.*

1. *If $P \not\leqslant_{tr} Q$, then $[\![P]\!]_L \not\leqslant_{tr} [\![Q]\!]_R$.*
2. *If $P \leqslant_{ready} Q$, then $[\![P]\!]_L \leqslant_{ready} [\![Q]\!]_R$.*

The rest of this subsection is devoted to the proof of the above theorem.

We first state the following simple properties.

Lemma 6.6. *Let $m \in \{L', R'\}$ and P be a BPP process. Then, $[\![P]\!]_m \stackrel{b}{\not\longrightarrow}$ and $[\![P]\!]_m \stackrel{ab^N}{\not\longrightarrow}$.*

Proof. Suppose $[\![P]\!]_m \stackrel{t}{\longrightarrow} E$ where $t \in \{a, b\}$. Then, $t \neq b$ and $E \stackrel{b^N}{\not\longrightarrow}$ follow by straightforward induction on the derivation of $[\![P]\!]_m \stackrel{t}{\longrightarrow} E$. □

The following property follows immediately from the definition of the encodings.

Lemma 6.7. *If $P \stackrel{l}{\longrightarrow} Q$, then $[\![P]\!]_{L'} \stackrel{[\![l]\!]}{\longrightarrow} [\![Q]\!]_{L'}$ and $[\![P]\!]_{R'} \stackrel{[\![l]\!]}{\longrightarrow} [\![Q]\!]_{R'}$.*

Proof. Straightforward induction on the derivation of $P \stackrel{l}{\longrightarrow} Q$. □

The following lemma states a property of $[\![P]\!]^s_m$.

Lemma 6.8. *Let $m \in \{L', R'\}$. Suppose also that s is a suffix of $[\![l]\!]$ for some l. If $[\![P]\!]^s_m \stackrel{s'}{\longrightarrow} E$ with $\#a(s) = \#a(s')$ and $\#b(s) = \#b(s')$, then $s = s'$ and $E = [\![P]\!]_m$.*

Proof. Since the case where $m = L'$ is trivial, we show only the case for $m = R'$. The proof proceeds by induction on the structure of s. Note that since s is a suffix of $[\![l]\!]$, $\#a(s) \leqslant 2$.

- Case $s = as_1$: If $\#a(s_1) = 1$, then $[\![P]\!]^s_{R'} = a[\![P]\!]^{s_1}_{R'}$; hence the result follows immediately from the induction hypothesis. If $\#a(s_1) = 0$, then $[\![P]\!]^s_{R'} = a[\![P]\!]^{s_1}_{R'} + aH^{(\#b(s_1)-1)} + aG_1$. Thus, s' must be of the form as'_1, with $[\![P]\!]^{s_1}_{R'} \stackrel{s'_1}{\longrightarrow} E$, $H^{(\#b(s_1)-1)} \stackrel{s'_1}{\longrightarrow} E$, or $G_1 \stackrel{s'_1}{\longrightarrow} E$. Note that $\#a(s'_1) = \#a(s') - 1 = \#a(s) - 1 = 0$. Moreover, since $\#b(s'_1) = \#b(s_1) \geqslant 1$, it cannot be the case that $H^{(\#b(s_1)-1)} \stackrel{s'_1}{\longrightarrow}$ or $G_1 \stackrel{s'_1}{\longrightarrow}$. Thus, we get $[\![P]\!]^{s_1}_{R'} \stackrel{s'_1}{\longrightarrow} E$, from which the result follows by the induction hypothesis.
- Case $s = bs_1$: Since s is a proper suffix of $[\![l]\!]$, $\#a(s) \leqslant 1$. If $\#a(s_1) = 1$, then $[\![P]\!]^s_{R'} = b[\![P]\!]^{s_1}_{R'} + aH^{(\#b(s_1))} + aG_1$. If s' is of the form bs'_1, then it must be the case that $[\![P]\!]^{s_1}_{R'} \stackrel{s'_1}{\longrightarrow} E$, and the result follows immediately from the induction hypothesis. If s' is of the form as'_1, then either $H^{(\#b(s_1))} \stackrel{s'_1}{\longrightarrow} E$ or $G_1 \stackrel{s'_1}{\longrightarrow} E$. Neither cannot be the case, however, since $\#b(s'_1) = \#b(s') = \#b(s) = \#b(s_1) + 1$ and $\#a(s'_1) = 0$.

  If $\#a(s_1) = 0$, then $[\![P]\!]^s_{R'} = b[\![P]\!]^{s_1}_{R'} + aG_1 + aG_2$. Since $\#a(s') = \#a(s) = \#a(s_1) = 0$, the assumption $[\![P]\!]^s_m \stackrel{s'}{\longrightarrow} E$ implies $s' = bs'_1$ and $[\![P]\!]^{s_1}_{R'} \stackrel{s'_1}{\longrightarrow} E$. Thus, the result follows immediately from the induction hypothesis. □

Next, we shall prove the converse of Lemma 6.7, meaning that if $[\![P]\!]_{L'}$ or $[\![P]\!]_{R'}$ can make an $[\![l]\!]$-transition, then P can also make an l-transition. To state the lemma, we introduce the notion of (evaluation) contexts.

Definition 6.9 *(contexts).* The set of (evaluation) contexts is defined by:

$$C ::= [\,] \mid (C \mid P) \mid (P \mid C)$$

We write $C[P]$ for the process obtained by replacing $[\,]$ in C with P.

The definitions of $[\![\cdot]\!]_{L'}$ and $[\![\cdot]\!]_{R'}$ are extended to contexts by $[\![[\,]]\!]_m = [\,]$. The following is the converse of Lemma 6.7. The condition about the intermediate state E implies that $[\![P]\!]_m$ can make $[\![l]\!]$-transitions only by reducing a subprocess $[\![P']\!]^{s_2}_m$. In other words, $H^{(k)}$, G_1, and G_2 cannot be involved in any valid transition sequence.

Lemma 6.10. *Let $m \in \{L', R'\}$. If $[\![P]\!]_m \stackrel{s_1}{\longrightarrow} E \stackrel{s_2}{\longrightarrow} F$ with $s_1 s_2 = [\![l]\!]$ and $s_1 \neq \epsilon$, then there exist C and P' that satisfy the following conditions.*

1. *$P \stackrel{l}{\longrightarrow} C[P']$*
2. *$E = [\![C]\!]_m[[\![P']\!]^{s_2}_m]$ and $F = [\![C[P']]\!]_m$.*

Proof. The proof proceeds by induction on the derivation tree of the first step of $[\![P]\!]_m \stackrel{s_1}{\longrightarrow} E$, with case analysis on the last rule used.

- Case Tr-Act: By the definition of the encoding, it must be the case that $[\![P]\!]_m = [\![P_1]\!]_m^{[\![l']\!]}$ and $P = l'P_1$. By Lemma 6.8, it must be the case that $F = [\![P_1]\!]_m$ and $[\![l]\!] = [\![l']\!]$. By the definition of the encoding, it must also be the case that $E = [\![P]\!]_m^{s_2}$. Thus, the required result holds for $C = [\,]$ and $P' = P_1$.
- Case Tr-OrL: By the definition of the encoding, it must be the case that $P = P_1 + P_2$ with $[\![P_1]\!]_m \xrightarrow{s_1} E \xrightarrow{s_2} F$. By the induction hypothesis, there must exist C_1 and P'_1 such that $P_1 \xrightarrow{l} C_1[P'_1]$ with $E = [\![C_1]\!]_m[[\![P'_1]\!]_m^{s_2}]$ and $F = [\![C_1]\!]_m[[\![P'_1]\!]_m]$. The required result holds for $C = C_1$ and $P' = P'_1$.
- Case Tr-OrR: Similar to the case above.
- Case Tr-ParL: By the definition of the encoding, it must be the case that $P = P_1 \mid P_2$. By the condition $[\![P_1 \mid P_2]\!]_m \xrightarrow{s_1} E \xrightarrow{s_2} F$, there must exist $s_{11}, s_{12}, s_{21}, s_{22}$ such that:

$$\begin{array}{cc} [\![P_1]\!]_m \xrightarrow{s_{11}} E_1 \xrightarrow{s_{21}} F_1 & [\![P_2]\!]_m \xrightarrow{s_{12}} E_2 \xrightarrow{s_{22}} F_2 \\ E = E_1 \mid E_2 & F = F_1 \mid F_2 \\ s_i \text{ is a shuffle of } s_{i1} \text{ and } s_{i2} & s_{11} \neq \epsilon \end{array}$$

  Suppose $s_{12}s_{22} \neq \epsilon$. Then, by Lemma 6.6, $s_{11}s_{21} = ab^{x_1}$ and $s_{12}s_{22} = ab^{x_2}$ where $x_1, x_2 \leqslant N-1$. This contradicts with the condition that $s_1 s_2$ is a shuffle of $s_{11}s_{21}$ and $s_{12}s_{22}$, since $\#b(s_1s_2) = \#b([\![l]\!]) = 2N-1$. Thus, we have $s_{12}s_{22} = \epsilon$. By the induction hypothesis, there exist C_1 and P'_1 such that $P_1 \xrightarrow{l} C_1[P'_1]$ with $E_1 = [\![C_1]\!]_m[[\![P'_1]\!]_m^{s_{21}}]$ and $F = [\![C_1[P'_1]]\!]_m$. The required result holds for $C = C_1 \mid P_2$ and $P' = P'_1$.
- Case Tr-ParR: Similar to the above case.
- Case Tr-Rec: In this case, we have $P = \mu X.P_1$ and $[\![[P/X]P_1]\!]_m = [[\![P]\!]_m/X][\![P_1]\!]_m \xrightarrow{s_1} E \xrightarrow{s_2} F$. By the induction hypothesis, we have C_1 and P'_1 such that $[P/X]P_1 \xrightarrow{l} C_1[P'_1]$ with $E = [\![C]\!]_m[[\![P']\!]_m^{s_2}]$ and $F = [\![C_1[P'_1]]\!]_m$. The required result holds for $C = C_1$ and $P' = P'_1$. □

Lemmas 6.7 and 6.10 above imply that for the "valid" transitions of $[\![P]\!]_m$ coincides with those of P. Next, we study properties of $H^{(k)}$ and G_i. We introduce a variation of the ready simulation for that purpose.

Definition 6.11. $\leqslant_{ready-a}$ is the union of all the binary relations $\mathcal{R}$ that satisfies the following conditions.

- $\forall P, Q, l.(P\mathcal{R}Q \wedge P \xrightarrow{l} P' \Longrightarrow \exists Q'.(P'\mathcal{R}Q' \wedge Q \xrightarrow{l} Q'))$
- $\forall P, Q.(P\mathcal{R}Q \Longrightarrow \mathit{readies}(P)\backslash\{a\} = \mathit{readies}(Q)\backslash\{a\})$

The next lemma states that $H^{(k)}$, G_1, and G_2 work as "almighty" processes for certain classes of processes, in the sense that they can ready-simulate any process belonging to those classes. Those properties will be used for proving that $[\![Q]\!]_{R'}$ can simulate all the "invalid" transitions of $[\![P]\!]_{L'}$.

Lemma 6.12.

1. *If* $E \xrightarrow{b} \not\xrightarrow{b^k}$, *then* $E \leqslant_{ready-a} H^{(k)}$.
2. *If* $E \not\xrightarrow{b}$, *then* $E \leqslant_{ready-a} G_1$.
3. *If* $E \xrightarrow{b}$, *then* $E \leqslant_{ready-a} G_2$.

Proof. It suffices to show that the following relation $\mathcal{R}$ satisfies the conditions in Definition 6.11.

$$\begin{array}{rcl} \mathcal{R} & = & \{(E, H^{(k)}) \mid E \xrightarrow{b} \not\xrightarrow{b^k}\} \\ & \cup & \{(E, G_1) \mid E \not\xrightarrow{b}\} \\ & \cup & \{(E, G_2) \mid E \xrightarrow{b}\} \end{array}$$

Suppose $(E, F) \in \mathcal{R}$.

- Case where (E, F) is in the first set of $\mathcal{R}$:

  In this case, $E \xrightarrow{b} \not\xrightarrow{b^k}$ and $F = H^{(k)}$. We have $\mathit{readies}(E)\backslash\{a\} = \mathit{readies}(F)\backslash\{a\} = \{b\}$. Suppose $E \xrightarrow{t} E'$. If $t = b$ and $b \in \mathit{readies}(E')$, then $(E', H^{(k-1)})$ is in the first set. If $t = a$ and $b \in \mathit{readies}(E')$, then $(E', G_2) \in \mathcal{R}$ and $F \xrightarrow{t} G_2$. Otherwise, $(E', G_1) \in \mathcal{R}$ and $F \xrightarrow{t} G_1$.
- Case where (E, F) is in the second set of $\mathcal{R}$:

  In this case, $E \not\xrightarrow{b}$ and $F = G_1$. We have $\mathit{readies}(E)\backslash\{a\} = \mathit{readies}(F)\backslash\{a\} = \emptyset$. Suppose $E \xrightarrow{t} E'$. Then, t must be a. Let G be G_1 if $b \notin \mathit{readies}(E')$, and G_2 otherwise. Then, we have $F \xrightarrow{t} G$ and (E', G) as required.

- Case where (E,F) is in the third set of $\mathcal{R}$:
  In this case, $E \xrightarrow{b}$ and $F = G_2$. We have $readies(E)\backslash\{a\} = readies(F)\backslash\{a\} = \{b\}$. Suppose $E \xrightarrow{t} E'$. Let G be G_1 if $b \notin readies(E')$, and G_2 otherwise. Then, we have $F \xrightarrow{t} G$ and (E', G) as required. □

Lemma 6.13. *If $[\![P]\!]_R \xrightarrow{[\![l]\!]} E$, then $E = [\![Q]\!]_R$ with $P \xrightarrow{l} Q$.*

Proof. By the definition of $[\![\cdot]\!]_R$, $[\![P]\!]_R = [\![P]\!]_{R'} \mid A$. By the assumption $[\![P]\!]_R \xrightarrow{[\![l]\!]} E$, E must be of the form $E_1 \mid A$ where:

$[\![P]\!]_{R'} \xrightarrow{s_1} E_1 \quad A \xrightarrow{s_2} A$
$[\![l]\!]$ is a shuffle of s_1 and s_2.

Suppose that s_2 is non-empty. Then it must be the case that $s_1 = ab^{2N-1}$ and $s_2 = a$. However, $[\![P]\!]_{R'} \not\xrightarrow{s_1}$ by Lemma 6.6; hence a contradiction. Therefore, we have $[\![P]\!]_R \xrightarrow{[\![l]\!]} E_1$. Thus, the required result follows from Lemma 6.10. □

Lemma 6.14. *Let E and F be 2-label BPP processes. If $E \leqslant_{ready-a} F$, then $E \mid A \leqslant_{ready} F \mid A$.*

Proof. It suffices to show that the following relation $\mathcal{R}$ is a ready simulation.

$$\mathcal{R} \quad = \quad \{(E \mid A, F \mid A) \mid E \leqslant_{ready-a} F\}$$

$E \leqslant_{ready-a} F$ holds for all E and F such that $(E \mid A, F \mid A) \in \mathcal{R}$. Therefore, we have $readies(E \mid A) = readies(F \mid A) = readies(E) \cup \{a\}$ by $readies(E)\backslash\{a\} = readies(F)\backslash\{a\}$ and the definition of A. Suppose $E \mid A \xrightarrow{t} E'$.

- Case where $E \xrightarrow{t} E_0$ with $E' = E_0 \mid A$.
  In this case, there exists F_0 such that $F \xrightarrow{t} F_0$ and $E_0 \leqslant_{ready-a} F_0$ because of $E \leqslant_{ready-a} F$. As a result, $(E', F_0 \mid A) \in \mathcal{R}$.
- Case where $A \xrightarrow{t} A$ with $E' = E \mid A$.
  In this case, we have $F \xrightarrow{t} F$ and $(E', F) \in \mathcal{R}$ as required. □

We are now ready to prove Theorem 6.5.

Proof of Theorem 6.5. We first recall the statement of the theorem:

"Let P and Q be BPP processes.

1. If $P \not\leqslant_{tr} Q$, then $[\![P]\!]_L \not\leqslant_{tr} [\![Q]\!]_R$.
2. If $P \leqslant_{ready} Q$, then $[\![P]\!]_L \leqslant_{ready} [\![Q]\!]_R$."

1. It suffices to show that $[\![P]\!]_L \leqslant_{tr} [\![Q]\!]_R$ implies $P \leqslant_{tr} Q$. Suppose $[\![P]\!]_L \leqslant_{tr} [\![Q]\!]_R$ and $l_1 \cdots l_k \in traces(P)$. By Lemma 6.7, we have $[\![l_1]\!] \cdots [\![l_k]\!] \in traces([\![P]\!]_{L'}) \subseteq traces([\![P]\!]_L)$. By the assumption $[\![P]\!]_L \leqslant_{tr} [\![Q]\!]_R$, we have $[\![l_1]\!] \cdots [\![l_k]\!] \in [\![Q]\!]_R$. By Lemma 6.13, we obtain $l_1 \cdots l_k \in traces(Q)$ as required.
2. By Lemma 6.14, it suffices to show that $P \leqslant_{ready} Q$ implies $[\![P]\!]_{L'} \leqslant_{ready} [\![Q]\!]_{R'}$. Let $\mathcal{R}$ be the following binary relation on 2-label BPP.

$$\begin{array}{ll} & \{(E,F) \mid E,F \in \mathbf{BPP}_{\{a,b\}} \wedge E \leqslant_{ready-a} F\} \\ \cup & \{([\![P]\!]_{L'}, [\![Q]\!]_{R'}) \mid P \leqslant_{ready} Q\} \\ \cup & \{(E,F) \mid P \leqslant_{ready} Q \wedge P' \leqslant_{ready} Q' \wedge s_1 s_2 = [\![l]\!] \wedge s_1, s_2 \neq \epsilon \wedge \\ & \quad [\![P]\!]_{L'} \xrightarrow{s_1} E \xrightarrow{s_2} [\![P']\!]_{L'} \wedge [\![Q]\!]_{R'} \xrightarrow{s_1} F \xrightarrow{s_2} [\![Q']\!]_{R'}\} \end{array}$$

We show that $\mathcal{R} \subseteq \leqslant_{ready-a}$. Suppose that $(E,F) \in \mathcal{R}$. The case where (E,F) is in the first set is trivial.

- Case (E,F) is in the second set.
  In this case, $E = [\![P]\!]_{L'}$ and $F = [\![Q]\!]_{R'}$, with $P \leqslant_{ready} Q$. By the definition of the encodings, we have $readies(E)\backslash\{a\} = readies(F)\backslash\{a\} = \emptyset$.
  Suppose $E \xrightarrow{t} E'$. By the definition of $[\![\cdot]\!]_{L'}$, $t = a$ and $E \xrightarrow{a} E' \xrightarrow{s} E''$ with $as = [\![l]\!]$. By Lemma 6.10, we have $E'' = [\![P']\!]_{L'}$ and $P \xrightarrow{l} P'$ for some P'. By the condition $P \leqslant_{ready} Q$, we have $Q \xrightarrow{l} Q'$ and $P' \leqslant_{ready} Q'$ for some Q'. By Lemma 6.7, we have $F = [\![Q]\!]_{R'} \xrightarrow{t} F' \xrightarrow{s} [\![Q']\!]_{R'}$ for some F'. The result follows, since (E', F') is in the third set of $\mathcal{R}$.
- Case (E,F) is in the third set.
  In this case, we have:

$$\begin{array}{l} P \leqslant_{ready} Q \quad P' \leqslant_{ready} Q' \quad s_1 s_2 = [\![l]\!] \quad s_1, s_2 \neq \epsilon \\ [\![P]\!]_{L'} \xrightarrow{s_1} E \xrightarrow{s_2} [\![P']\!]_{L'} \quad [\![Q]\!]_{R'} \xrightarrow{s_1} F \xrightarrow{s_2} [\![Q']\!]_{R'} \end{array}$$

By Lemma 6.10, we have:

$$E = [\![C_1]\!]_{L'}([\![P_1]\!]_{L'}^{s_2}) \qquad P' = C_1[P_1]$$
$$F = [\![C_2]\!]_{R'}([\![Q_1]\!]_{R'}^{s_2}) \qquad Q' = C_2[Q_1]$$

Let t_2 be the first label of s_2. Then, we have $readies(E)\backslash\{a\} = readies(F)\backslash\{a\} = \{t_2\}\backslash\{a\}$.
Suppose $E \xrightarrow{t} E'$. Case analysis on whether the transition comes from $[\![P_1]\!]_{L'}^{s_2}$ or $[\![C_1]\!]_{L'}$.

– Case where $[\![P_1]\!]_{L'}^{s_2} \xrightarrow{t} E_0$ with $E' = [\![C_1]\!]_{L'}[E_0]$.

By the definition of the encoding, $s_2 = ts_2'$ and $E_0 = [\![P_1]\!]_{L'}^{s_2'}$ hold. Let $F' = [\![C_2]\!]_{R'}([\![Q_1]\!]_{R'}^{s_2'})$. Then we have $F \xrightarrow{t} F'$ and $E'\mathcal{R}F'$ as required.

– Case where $[\![C_1]\!]_{L'} \xrightarrow{t} C_1'$ with $E' = C_1'[[\![P_1]\!]_{L'}^{s_2}]$.
In this case, $t = a$. Let us define F' by:

$$F' = \begin{cases} [\![C_2]\!]_{R'}[G_2] & \text{if } s_2 = b^k \\ [\![C_2]\!]_{R'}[G_1] & \text{if } s_2 = ab^k \text{ and } b \notin readies(E') \\ [\![C_2]\!]_{R'}[H^{(\#b(s_2)-1)}] & \text{otherwise} \end{cases}$$

By the definition of the encoding, it is easy to see that $F \xrightarrow{t} F'$. $E' \leqslant_{ready-a} F'$ follows from Lemma 6.12: for the first case, since $readies(E')\backslash\{a\} = \{b\}$, we have $E' \leqslant_{ready-a} G_2 \leqslant_{ready-a} F'$. For the second case, since $readies(E')\backslash\{a\} = \phi$, we have $E' \leqslant_{ready-a} G_1 \leqslant_{ready-a} F'$. For the third case, let $s_2 = b^iab^j$ (where $i+j = \#b(s_2)$). Since s_2 is a suffix of $[\![l]\!]$, $j \geqslant N$. Because $C_1 \overset{b}{\not\longrightarrow}$ and $[\![P_1]\!]_{L'}^{s_2} \overset{b^{i+1}}{\not\longrightarrow}$, it must be the case that $E' \overset{b^{\#b(s_2)}}{\not\longrightarrow}$. Thus, by Lemma 6.12, we have $E' \leqslant_{ready-a} H^{(\#b(s_2)-1)} \leqslant_{ready-a} F'$ as required. □

6.3. Undecidability results for 2-label BPP

The following theorem follows as an immediate corollary of Theorems 3.4 and 6.5.

Theorem 6.15. *Let M be a Minsky machine. Then, there exist 2-label BPP processes P_M and Q_M that satisfy the following properties.*

- *If M halts, then $P_M \not\leqslant_{tr} Q_M$.*
- *If M does not halt, then $P_M \leqslant_{ready} Q_M$.*

As a corollary, we obtain undecidability of any preorders between the trace preorder and the ready simulation.

Corollary 6.16. *Let $\mathcal{R}$ be a binary relation on 2-label BPP such that $\leqslant_{ready} \subseteq \mathcal{R} \subseteq \leqslant_{tr}$. Then, the relation $\mathcal{R}$ is undecidable.*

Proof. By Theorem 6.15, M halts if and only if $P_M\mathcal{R}Q_M$ holds. Since the halting problem of Minsky machines is undecidable, so is $\mathcal{R}$. □

Similarly, we also obtain undecidability of any equivalence relations between the trace equivalence and the ready simulation equivalence.

Corollary 6.17. *Let $\mathcal{R}$ be a binary relation on 2-label BPP such that $\sim_{ready} \subseteq \mathcal{R} \subseteq \sim_{tr}$. Then, the relation $\mathcal{R}$ is undecidable.*

Proof. Note that $P \not\leqslant_{tr} Q$ implies $P + Q \not\sim_{tr} Q$, and that $P \leqslant_{ready} Q$ implies $P + Q \sim_{ready} Q$. Thus, by Theorem 6.15, M does not halt if and only if $P_M + Q_M\mathcal{R}Q_M$ holds. The relation $\mathcal{R}$ is therefore undecidable. □

Undecidability of n-nested simulation and n-bounded-tr-bisimulation can be proved in exactly the same way as in Sections 4 and 5.

Theorem 6.18. *For $n > 0$, n-nested simulation, n-nested simulation equivalence and n-bounded-tr-bisimulation are undecidable for 2-label BPP.*

7. Conclusion

In this paper, we have shown that all the preorders/equivalence relations in Glabbeek's linear time-branching time spectrum except the bisimulation equivalence are undecidable for general BPP. We have also shown that they are undecidable even for 2-label BPP.

Our results thus give a full account of the decidability of behavioural equivalences for BPP. The undecidability results for simulation-like equivalences follow from a reduction from the halting problem for Minsky machines, whereas the undecidability results for the equivalences that correspond to the approximants of bisimilarity are proved using the same method as used in [5] to prove the similar results for BPA.

Acknowledgment

We thank Jiří Srba and the anonymous referees for useful comments and suggestions.

References

[1] J.A. Bergstra, J.W. Klop, Process algebra for synchronous communication, Information and Control, 60 (1–3) (1984) 109–137.
[2] S. Christensen, Y. Hirshfeld, F. Moller, Bisimulation is decidable for all basic parallel processes, Proceedings of CONCUR'93, Lecture Notes in Computer Science, vol. 715, Springer-Verlag, 1993.
[3] S. Christensen, Y. Hirshfeld, F. Moller, Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes, Proceedings of IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, 1993, pp. 386–396.
[4] S. Christensen, H. Hüttel, C. Stirling, Bisimulation equivalence is decidable for all context-free processes, Information and Computation 121 (2) (1995) 143–148.
[5] J.F. Groote, H. Hüttel, Undecidable equivalences for basic process algebra, Information and Computation 115 (2) (1994) 354–371.
[6] J.F. Groote, F.W. Vaandrager, Structural operational semantics and bisimulation as a congruence (extended abstract), in: G. Ausiello, M. Dezani-Ciancaglini, S.R.D. Rocca (Eds.), ICALP, Lecture Notes in Computer Science, vol. 372, Springer, 1989, pp. 423–438.
[7] Y. Hirshfeld, Petri nets and the equivalence problem, CSL 1993, Lecture Notes in Computer Science, vol. 832, Springer-Verlag, 1993, pp. 165–174.
[8] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979..
[9] H. Hüttel, Undecidable equivalence for basic parallel processes, Proceedings of TACS94, Lecture Notes in Computer Science, vol. 789, Springer-Verlag, 1994, pp. 454–464.
[10] D.T. Huynh, L. Tian, On deciding readiness and failure equivalences for processes, Information and Computation 117 (2) (1995) 193–205.
[11] P. Kanellakis, S. Smolka, CCS expressions, finite state processes, and three problems of equivalence, Information and Computation 86 (1990) 43–68.
[12] N. Kobayashi, Type systems for concurrent programs, Proceedings of UNU/IIST 20th Anniversary Colloquium, Lecture Notes in Computer Science, vol. 2757, Springer-Verlag, 2003, pp. 439–453.
[13] N. Kobayashi, T. Suto, Undecidability of 2-label BPP equivalences and behavioral type systems for the π-calculus, Proceedings of ICALP 2007, Lecture Notes in Computer Science, vol. 4596, Springer-Verlag, 2007, pp. 740–751.
[14] R. Milner, Communication and Concurrency, Prentice Hall, 1989..
[15] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice Hall, 1967..
[16] D. Park, Concurrency and automata on infinite sequences, Proceedings 5th GI Conference, Lecture Notes in Computer Science, vol. 104, Springer-Verlag, 1981, pp. 167–183.
[17] W.C. Rounds, S.D. Brookes, Possible futures, acceptances, refusals, and communicating processes, Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, 1981, pp. 140–149.
[18] J. Srba, On the power of labels in transition systems, Proceedings of CONCUR 2001, Lecture Notes in Computer Science, vol. 2154, Springer-Verlag, 2001, pp. 277–291.
[19] R. van Glabbeek, The linear time–branching time spectrum, in: J. Baeten, J. Klop (Eds.), Proceedings of CONCUR'90, Lecture Notes in Computer Science, vol. 458, Springer-Verlag, 1990, pp. 278–297.