Undecidable Equivalences for Basic Process Algebra

To appear in *Information and Computation*

Jan Friso Groote*
Department of Philosophy, Utrecht University
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
e-mail: JanFriso.Groote@phil.ruu.nl

Hans Hüttel†
Laboratory for Foundations of Computer Science
James Clerk Maxwell Building, University of Edinburgh,
Edinburgh EH9 3JZ, Scotland
e-mail: hans@dcs.ed.ac.uk

Abstract

A recent theorem [3, 7, 19] shows that strong bisimilarity is decidable for the class of *normed* BPA processes, which correspond to a class of context-free grammars generating the ϵ-free context-free languages. In [21] Huynh and Tian have shown that readiness and failure equivalence are undecidable for BPA processes. In this paper we examine all other equivalences in the linear/branching time hierarchy [13] and show that *none* of them are decidable for normed BPA processes.

1 Introduction

In the field of process theory much attention has been devoted to the study of process calculi and in particular to behavioural semantics for these calculi. A variety of equiv-

*Supported by the European Communities under RACE project no. 1046 (SPECS) and ESPRIT Basic Research Action 3006 (CONCUR).This paper was written during a visit of the first author to Edinburgh.

†Supported via a position at Aalborg University and by the Danish Research Academy.

alences have been proposed in order to capture the behavioural aspects of processes better than language equivalence from traditional automata theory.

Various criteria exist for comparing the merits and deficiencies of these equivalences. A systematic approach consists of classifying the equivalences according to their coarseness. For this purpose van Glabbeek proposed the *linear/branching time spectrum* which is illustrated in Figure 1 [13]. The least discriminating equivalences are at the bottom of the diagram. Arrows indicate inclusion. The coarsest equivalences are trace

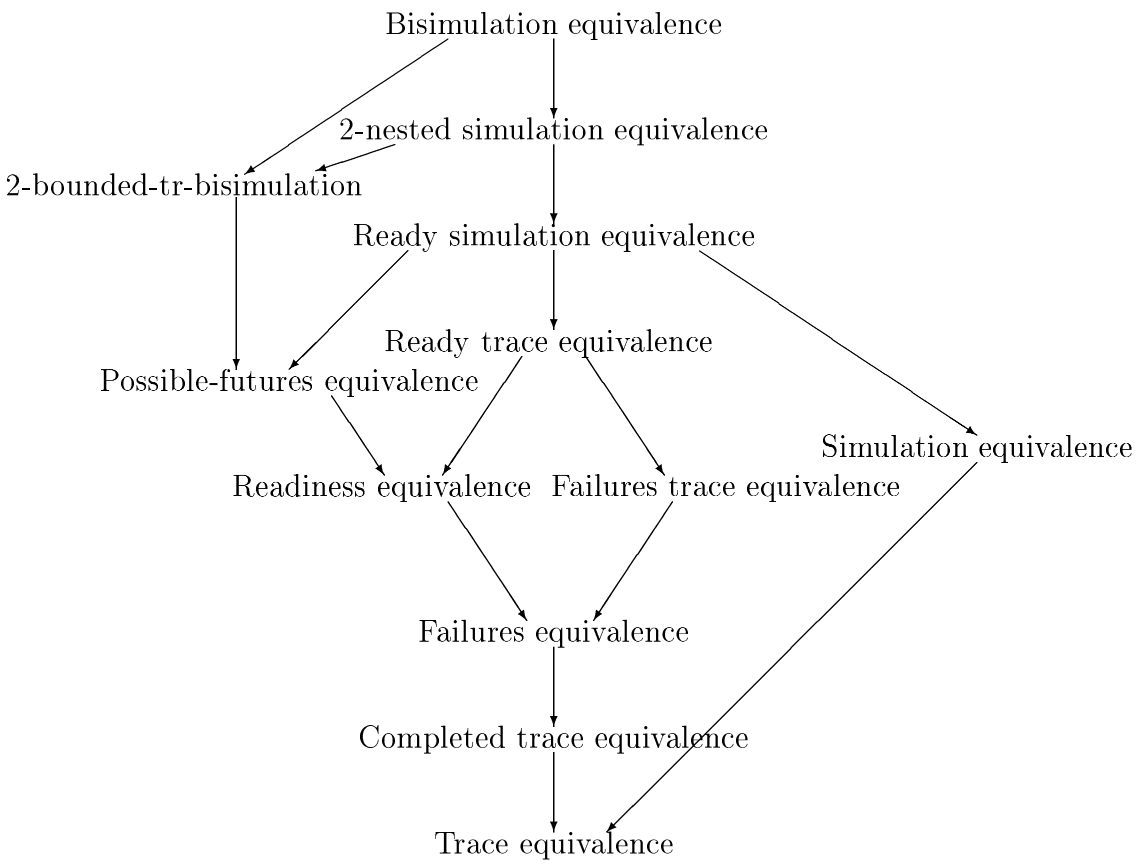

Figure 1: The linear-time/branching time hierarchy of equivalences.

equivalence and completed trace equivalence (=language equivalence). Directly above them we have the testing/failures equivalence investigated by De Nicola and Hennessy (see e.g. [17]). At the top of the diagram is bisimulation equivalence (or bisimilarity), a notion introduced by Park [28] and subsequently widely used in process theories.

A somewhat less tangible criterion is that of observability. For instance it has been argued that while bisimulation equivalence has many nice mathematical properties it fails to have a computational justification in that (in)equivalence seems not to be intuitively observable. All existing proposals to justify bisimulation as an observational equivalence have used unnatural operators. In [1] global testing operators were necessary and in [14] lookahead in combination with the possibility to check for the absence of activity was needed. Bloom, Istrail and Meyer argue that only completed traces should count as observations and define an equivalence which is a completed trace congruence under a 'reasonable' set of process constructs [5]. This equivalence is ready simulation equivalence or 2/3-bisimulation and it is clearly below bisimulation equivalence.

Decidability is another relevant criterion for evaluating behavioural equivalences. For finite-state processes, i.e. finite automata, all equivalences in Figure 1 are decidable [22]. However, it is well known (see e.g. [18]) that completed trace equivalence becomes undecidable when one moves beyond finite automata to context-free languages, which correspond to processes specified in Basic Process Algebra (BPA) [2]. A recent theorem [3] (but see also [7, 15, 19]) shows that strong bisimilarity is decidable for *normed* BPA processes. This result has been extended to BPA processes in general in [10] and similar results have been obtained for Basic Parallel Processes [8, 9]. In [21] Huynh and Tian have shown that readiness and failures equivalence are undecidable for BPA processes. In this paper we examine all other equivalences in Figure 1 and show that *none* of them are decidable for normed BPA processes. We also present a slightly more elegant version of the undecidability proof of [21].

2 Preliminaries

2.1 Normed recursive BPA processes

The BPA (Basic Process Algebra) processes or process expressions [3] are given by the abstract syntax

$$p ::= a \mid X \mid p_1 + p_2 \mid p_1 \cdot p_2$$

Here a ranges over a set Act of atomic actions, and X over a set Var of variables. The symbol $+$ is the non-deterministic choice while $p_1 \cdot p_2$ represents the sequential composition of p_1 and p_2 (we often omit the '$\cdot$').

We say that a process expression is *guarded* iff every variable occurrence in p occurs in a subexpression aq of p. Recursive processes are defined by a finite set Δ of guarded equations

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

where the X_i are distinct process variables, and the p_i are guarded BPA expressions with free variables in $Var(\Delta) = \{X_1, \ldots, X_k\}$. One variable (generally X_1) is singled out as the *root*.

The operational semantics of a BPA process expression, given a finite system of guarded equations Δ, is a transition relation $\longrightarrow_\Delta$ containing the transitions provable by the following rules (ϵ denotes the empty process with the convention that ϵq is q):

$$\frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$\frac{p \xrightarrow{a} p'}{pq \xrightarrow{a} p'q} \qquad a \xrightarrow{a} \epsilon \quad a \in Act$$

$$\frac{p \xrightarrow{a} p'}{X \xrightarrow{a} p'} \qquad X \stackrel{\text{def}}{=} p \in \Delta$$

We omit the subscript Δ if it is clear from the context. A process expression together with an associated transition relation is called a process. If it is important that a process and its operational semantics are generated by the rules above, we speak explicitly about a BPA process.

Notation 2.1. We use the notations

- $p \xrightarrow{a}_\Delta p'$ for $p \xrightarrow{a} p' \in \longrightarrow_\Delta$,
- $p \xrightarrow{a}_\Delta$ for $\exists p' : p \xrightarrow{a}_\Delta p'$,
- $p \overset{a}{\not\longrightarrow}_\Delta$ for not $p \xrightarrow{a}_\Delta$,
- $p \not\longrightarrow_\Delta$ for $\forall a \in Act : p \overset{a}{\not\longrightarrow}_\Delta$,
- $p \xrightarrow{a_1 \cdots a_n}_\Delta p'$ for $p \xrightarrow{a_1}_\Delta \xrightarrow{a_2}_\Delta \cdots \xrightarrow{a_n}_\Delta p'$ and
- $p \equiv q$ iff p and q are syntactically the same.

In this paper we restrict our attention to *normed* BPA process expressions. As all normed BPA processes are also BPA processes, all our undecidability results immediately carry over to BPA in general.

Definition 2.2. The *norm* of a process p is defined by

$$|p| = \min\{length(w) \mid p \xrightarrow{w} \epsilon\}.$$

A finite set Δ of guarded equations is *normed* if for all $X \in Var(\Delta)$ it holds that $|X|$ is finite. A BPA process is called *normed*, if it has been generated via a normed set of guarded equations.

Note that the class of normed BPA processes does not include all the regular processes (such as $X \stackrel{\text{def}}{=} aX$). Nevertheless, it is a very rich family, including processes with infinitely many states.

Deterministic processes play an important role in our proofs.

Definition 2.3. We say that a process p is *deterministic* iff if $p \xrightarrow{w} p_1$ and $p \xrightarrow{w} p_2$, then $p_1 \equiv p_2$.

2.2 Normed recursive BPA processes in Greibach Normal Form

In [3] it is shown that any set of guarded equations can be effectively presented in the following normal form

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

where α_{ij} is a sequence of variables, such that the root of Δ' is bisimulation equivalent to that of Δ. Moreover, Δ is normed exactly when Δ' is. By analogy with context-free grammars this normal form is called *GNF* (Greibach Normal Form). Whenever the α_{ij}'s have length less than three, we talk of *restricted GNF* or 3-GNF.

There is an obvious relationship between the normed sets of guarded equations over BPA and the class of context-free grammars where all nonterminals have productions and all productions are guarded and none are useless or empty. Variables correspond to non-terminals and actions correspond to terminals. Each set $\{X_i = \sum_{j=1}^{n_i} a_{ij}\alpha_{ij} | 1 \leq i \leq m\}$ of guarded recursive equations corresponds directly to the family of productions $\{X_i \to a_{ij}\alpha_{ij} \mid 1 \leq i \leq m,\ 1 \leq j \leq n_i\}$.

The following definition is an adaptation to process theory of the well-known notion of an accepted language.

Definition 2.4. The language $L(p)$ *accepted* by a process p is given by $L(p) = \{w \mid p \xrightarrow{w} \epsilon\}$. The trace language $Tr(p)$ of a process p is given by $Tr(p) = \{w \mid p \xrightarrow{w}\}$. Two processes p and q are *completed trace equivalent*, notation $p \sim_{ctr} q$, (or language equivalent) iff $L(p) = L(q)$. Two processes p and q are *trace equivalent*, notation $p \sim_{tr} q$ iff $Tr(p) = Tr(q)$.

For unnormed processes it is commonly accepted that language equivalence and completed trace equivalence are not equivalent. For instance, the processes generated by $X = a\,X$ and $Y = b\,Y$ are generally not thought to be trace equivalent, whereas they generate the same language (namely $\emptyset$).

It is well known that it is undecidable whether two context-free grammars define the same language [18]. As empty productions and non-productive non-terminals can effectively be removed, we can state this in our terminology as $p \sim_{ctr} q$ for BPA processes p and q. It is also undecidable whether two BPA processes are trace equivalent – in our notation whether $p \sim_{tr} q$. For we can reduce language equivalence to trace equivalence, since $L(p) = L(q)$ iff $Tr(p\surd) = Tr(p\surd)$ where $\surd$ is a new action. So we have:

Theorem 2.5. *Completed trace equivalence (or language equivalence) and trace equivalence are undecidable for normed BPA processes.*

A *simple grammar* is a context-free grammar in restricted GNF such that there are no two distinct productions $A \to a\alpha_1, A \to a\alpha_2$ for any nonterminal A [12]. Thus, simple grammars correspond to those sets of guarded equations over BPA that generate *deterministic* BPA processes. For simple grammars the language equivalence problem is decidable [23]. Consequently we cannot effectively transform a set of guarded equations over BPA into a deterministic set of equations, while retaining language equivalence. If such an effective transformation would exist, we could reduce language equivalence for non-deterministic processes to language equivalence for deterministic processes.

As shown by Friedman the language containment problem for simple grammars is undecidable [12]. Formulated in our setting this theorem reads:

Theorem 2.6. *Let p and q be two normed and deterministic BPA processes. It is undecidable whether $L(p) \subseteq L(q)$.*

However, it is important to note that for deterministic BPA processes trace equivalence and bisimulation equivalence coincide [11] (where $\sim$ denotes strong bisimulation equivalence):

Proposition 2.7. *If p and q are deterministic processes, then $Tr(p) = Tr(q)$ iff $p \sim q$.*

Proof. $\{(p, q) \mid Tr(p) = Tr(q)\}$ is a bisimulation. □

Consequently, in the deterministic case the linear/branching time hierarchy collapses, and since language equivalence is decidable, *all* equivalences are *decidable* in this case.

3 Simulation and simulation equivalence

In this and the subsequent sections all equivalences in figure 1, except strong bisimulation and completed trace equivalence, are considered and proven undecidable for normed BPA processes by reducing language containment for deterministic processes or language equivalence to them. We start off with simulation and simulation equivalence.

Definition 3.1. A relation R between processes is a *simulation* iff whenever pRq then for each $a \in Act$ $p \xrightarrow{a} p' \;\Rightarrow\; \exists q : q \xrightarrow{a} q' \wedge p'Rq'$. A process p is *simulated* by a process q, notation $p \subsetneq\!\!\!\!\rightarrow q$, iff there is a simulation relation R with pRq. Two processes p and q are *simulation equivalent*, notation $p \leftrightarrows q$, iff $p \subsetneq\!\!\!\!\rightarrow q$ and $q \subsetneq\!\!\!\!\rightarrow p$.

We first show that simulation is undecidable for deterministic normed BPA processes. This is a direct corollary of Theorem 2.6.

Theorem 3.2. *Simulation is undecidable for deterministic normed BPA processes.*

Proof. Let p and q be normed deterministic BPA processes and let $\surd$ be a 'fresh' action. It is straightforward to show that

$$L(p) \subseteq L(q) \quad \text{iff} \quad p\surd \subsetneq q\surd$$

as p and q are deterministic. With this observation, we reduce language containment for deterministic normed BPA processes to simulation preorder for deterministic normed BPA processes. □

Theorem 3.3. *Simulation equivalence is undecidable for normed BPA processes.*

Proof. We can reduce simulation to simulation equivalence by the following observation:

$$p \subsetneq q \quad \text{iff} \quad p + q \leftrightarrows q.$$

□

4 n-nested simulation equivalence

The notion of n-nested simulation equivalence was introduced by Groote and Vaandrager [16] 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 $n \in \mathbf{N}$, *n-nested simulation*, written $\subsetneq^n$, is inductively defined by

- $p \subsetneq^0 q$ for all processes p and q,
- $p \subsetneq^{n+1} q$ iff there is a simulation relation $R \subseteq (\subsetneq^n)^{-1}$ with pRq.

Two processes p and q are *n-nested simulation equivalent*, written $p \leftrightarrows^n q$, iff $p \subsetneq^n q$ and $q \subsetneq^n p$.

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

Lemma 4.2. *For all $n \in \mathbf{N}$, n-nested simulation is a precongruence under action prefixing and $+$.*

Proof. Induction on n.

$n = 1$: This simply states that $\subsetneq$ is a precongruence under action prefixing and $+$. Clearly if $p \subsetneq q$ and $p \subsetneq q'$ we have that $ap \subsetneq aq$ and $p + p' \subsetneq q + q'$.

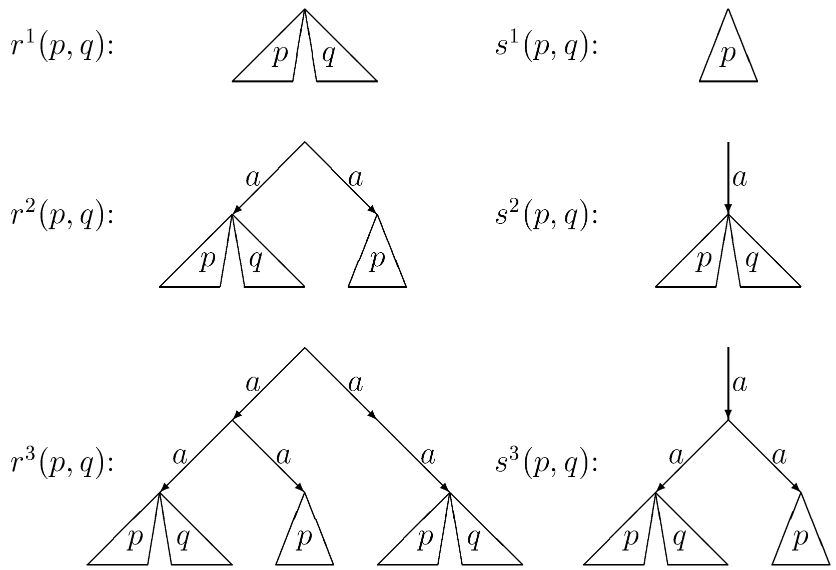

Figure 2: $r^n(p,q)$ and $s^n(p,q)$ for $n = 1, 2, 3$

Step - assuming for n: Here if $p \underset{\to}{\subset}^{n+1} q$ we have that there is a simulation $R \subseteq (\underset{\to}{\subset}^n)^{-1}$ such that pRq. But then, since $q \underset{\to}{\subset}^n p$ we must by induction hypothesis have $aq \underset{\to}{\subset}^n ap$ and thus $R \cup \{(ap, aq)\}$ is a simulation with $R \cup \{(ap, aq)\} \subseteq (\underset{\to}{\subset}^n)^{-1}$. The proof for $+$ is entirely similar.

$\square$

The class of processes defined in the following can be used to reduce simulation to both n-nested simulation and n-nested simulation equivalence. Some of these processes are depicted in Figure 2.

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

$$r^1(p,q) = p + q, \qquad s^1(p,q) = p,$$

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

Observe that if p and q are normed BPA processes, $r^n(p,q)$ and $s^n(p,q)$ are also normed.

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

1. $s^n(p,q) \underset{\to}{\subset}^n r^n(p,q)$,
2. $r^n(p,q) \underset{\to}{\subset}^n s^n(p,q)$ *iff* $q \underset{\to}{\subset} p$.

Proof. Both proofs proceed by induction. For 1 we get

$n = 1$: The lemma then reduces to $p \subsetneq p + q$ which obviously holds.

Step - assuming for n: Define for processes p and q the simulation

$$R = \{(a\, r^n(p, q), a\, r^n(p, q) + a\, s^n(p, q))\} \cup Id$$

where Id is the identity relation. $R \subseteq (\subsetneq^n)^{-1}$, as we have $r^n(p, q) \subsetneq^n r^n(p, q)$ and $s^n(p, q) \subsetneq^n r^n(p, q)$ by induction hypothesis, which by Lemma 4.2 gives us $a\, r^n(p, q) + a\, s^n(p, q) \subsetneq^n a\, r^n(p, q)$.

The proof for 2 has

$n = 1$: The lemma here reduces to $p + q \subsetneq p$ iff $q \subsetneq p$. If there is a simulation R with qRp then $\{(p + q, p)\} \cup R \cup Id$ is a simulation. And if $p + qRp$ for a simulation R, then $\{(q, p)\} \cup R$ is a simulation.

Step - assuming for n: For the 'if' direction suppose for processes p and q that $q \subsetneq p$ and $r^n(p, q) \subsetneq^n s^n(p, q)$. Then define the simulation

$$R = \{(a\, r^n(p, q) + a\, s^n(p, q), a\, r^n(p, q))\} \cup (\subsetneq^n)^{-1}$$

$R \subseteq (\subsetneq^n)^{-1}$ since Lemma 4.2 gives us $a\, r^n(p, q) \subsetneq^n a\, r^n(p, q) + a\, s^n(p, q)$, which means that $r^{n+1}(p, q) \subsetneq^{n+1} s^{n+1}(p, q)$. For the 'only if' case suppose $q \not\subsetneq p$. By the induction hypothesis $r^n(p, q) \not\subsetneq^n s^n(p, q)$. Then we cannot have

$$r^{n+1}(p, q) \subsetneq^{n+1} s^{n+1}(p, q),$$

for any candidate simulation would be one containing the pair

$$(a\, r^n(p, q) + a\, s^n(p, q), a\, r^n(p, q)).$$

As $r^{n+1}(p, q) \xrightarrow{a} s^n(p, q)$ can only simulated by $s^{n+1}(p, q) \xrightarrow{a} r^n(p, q)$ it must be that $(s^n(p, q), r^n(p, q)) \in R$. But as $R \subseteq (\subsetneq^n)^{-1}$ we would have $r^n(p, q) \subsetneq^n s^n(p, q)$.

□

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

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

$$q \subsetneq p \quad \text{iff} \quad r^n(p, q) \subsetneq^n s^n(p, q).$$

We reduce simulation to n-nested simulation equivalence using:

$$q \subsetneq p \quad \text{iff} \quad r^n(p, q) \leftrightarrows^n s^n(p, q).$$

Because $n > 0$ both facts follow directly from lemma 4.4. As simulation is undecidable, n-nested simulation and n-nested simulation equivalence cannot be decidable. □

One should notice here that the limit of the n-nested simulation equivalences for $n \to \omega$ is strong bisimulation equivalence:

Theorem 4.6. [16] *For any finitely branching labelled transition graph we have*

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

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

5 n-bounded-tr-bisimulation

We now consider n–bounded-tr-bisimulation. This equivalence is a generalisation of *trace equivalence* and *possible futures equivalence*, in that 1-bounded-tr-bisimulation corresponds to *trace equivalence* and 2-bounded-tr-bisimulation is the *possible futures equivalence* of [30].

Definition 5.1. We define *n-bounded-tr-bisimulation*, written $\sim_{tr}^n$, inductively as follows.

- $p \sim_{tr}^0 q$ for all processes p and q,
- $p \sim_{tr}^{n+1} q$ iff
  - if $p \xrightarrow{w} p'$ then $\exists q'$ such that $q \xrightarrow{w} q'$ and $p' \sim_{tr}^n q'$ and
  - if $q \xrightarrow{w} q'$ then $\exists p'$ such that $p \xrightarrow{w} p'$ and $p' \sim_{tr}^n q'$.

This notion of equivalence also arises naturally as the consecutive approximations of bisimulation equivalence [25, 26]. For finitely branching transition graphs, and therefore for BPA processes, the limit of the n-bounded-tr-bisimulations for $n \to \omega$ coincides with bisimulation equivalence:

Theorem 5.2. [26] *For any finitely branching labelled transition graph we have*

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

The following proof uses the same reduction that was employed in [22] to show that n-bounded tr-bisimulation for regular processes is PSPACE-complete. The following easy lemma is crucial:

Lemma 5.3. [22]

$$p \sim_{tr}^n q \textit{ iff } p + q \sim_{tr}^n p \textit{ and } p + q \sim_{tr}^n q$$

Lemma 5.4. [22] *Let p and q be processes. For all $n > 0$ it holds that*

$$p \sim_{tr}^{n} q \textit{ iff } a\,(p+q) \sim_{tr}^{n+1} a\,p + a\,q$$

Proof. For the 'if' half we prove the contrapositive, stating that $p \not\sim_{tr}^{n} q$ implies that $a(p+q) \not\sim_{tr}^{n+1} ap+aq$. Assume $p \not\sim_{tr}^{n} q$. Then $a(p+q) \xrightarrow{a} p+q$ whereas $ap+aq \xrightarrow{a} p$ and $ap+aq \xrightarrow{a} q$. Since $p \not\sim_{tr}^{n} q$ we have by the previous lemma that either $p+q \not\sim_{tr}^{n} p$ or $p+q \not\sim_{tr}^{n} q$, so clearly $a(p+q) \not\sim_{tr}^{n+1} ap+aq$. The 'only if' half of the proof also proceeds by contraposition, showing that $a(p+q) \not\sim_{tr}^{n+1} ap+aq$ implies $p \not\sim_{tr}^{n} q$. Assuming $a(p+q) \not\sim_{tr}^{n+1} ap+aq$, the action string that distinguishes $a(p+q)$ and $ap+aq$ must be a, since for any longer string w the w-derivatives are identical. Thus, either $p+q \not\sim_{tr}^{n} p$ or $p+q \not\sim_{tr}^{n} q$, and again by the previous lemma we get $p \not\sim_{tr}^{n} q$. □

Theorem 5.5. *For $n > 0$ n-bounded-tr-bisimulation is undecidable for normed BPA processes.*

Proof. We reduce n-bounded-tr-bisimulation to $n+1$-bounded-tr-bisimulation using Lemma 5.4. Since 1-bounded-tr-bisimulation is trace equivalence, which is undecidable, the result follows. □

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 Failures, readiness, failure-trace and ready-trace equivalences

In their paper [21] Huynh and Tian show that failures equivalence [6] and readiness equivalence [4, 27] are undecidable for normed BPA processes. Here we give an alternative account of their technique, using a simpler transformation than that of [21] to show that ready trace and failure trace equivalence are undecidable.

Definition 6.1. For any process p, define

$$\mathit{failures}(p) = \{(w, X) \mid \exists p' : p \xrightarrow{w} p', \forall a \in X : p' \stackrel{a}{\not\longrightarrow}\},$$

$$\mathit{readies}(p) = \{(w, X) \mid \exists p' : p \xrightarrow{w} p', p' \xrightarrow{a} \iff a \in X\}.$$

Processes p and q are *failures equivalent*, written $p \sim_f q$, iff $\mathit{failures}(p) = \mathit{failures}(q)$. Processes p and q are *readiness equivalent*, written $p \sim_r q$, iff $\mathit{readies}(p) = \mathit{readies}(q)$.

The proof technique of Huynh and Tian involves defining a class of processes, called *locally unary* processes, for which failures and readiness equivalence coincide with completed trace equivalence.

Definition 6.2. [21] A process p is *locally unary* iff for each p' with $p \xrightarrow{w} p'$ there is at most one $a \in Act$ such that $p' \xrightarrow{a}$.

Lemma 6.3. [21] *If p and q are locally unary normed processes then*

$$p \sim_r q \quad \text{iff} \quad p \sim_f q \quad \text{iff} \quad L(p) = L(q).$$

Proof. [21] As $\sim_r \subseteq \sim_f \subseteq \sim_{ctr}$, it is sufficient to show that $L(p) = L(q)$ implies $p \sim_r q$. Suppose $L(p) = L(q)$ and $(w, X) \in readies(p)$. If $X = \emptyset$ we have $w \in L(p)$ and hence, $(w, \emptyset) \in readies(q)$. Otherwise, since q is locally unary we have $X = \{a\}$ for some $a \in Act$. Since p is normed, there must be a $w' \in Act^*$ such that $waw' \in L(q)$. Since $L(p) = L(q)$ and q is locally unary, we must therefore also have that $(w, \{a\}) \in readies(q)$. □

The idea is now, given a Δ to construct a locally unary Δ' containing the variables of Δ such that $L(\alpha) = L(\beta)$ in Δ if and only if $L(\alpha) = L(\beta)$ in Δ'. The following construction accomplishes this. The idea is simply to precede any action by a # that indicates that a nondeterministic choice has been made.

Definition 6.4. Given a system of equations Δ in GNF let Δ' have the action set $Act \cup \{\#\}$ (where # is a new action) and process variables $Var \cup \{X_a \mid a \in Act\}$. For every process equation in Δ

$$X_i \stackrel{\text{def}}{=} \sum a_j \alpha_j$$

create the equations

$$\begin{aligned} X_i &\stackrel{\text{def}}{=} \sum \# X_{a_j} \alpha_j \\ X_{a_j} &\stackrel{\text{def}}{=} a_j \end{aligned}$$

in the new system Δ'.

It is obvious that Δ' is normed iff Δ is (in fact if $|X| = k$ in Δ then $|X| = 2k$ in Δ). We now have

Proposition 6.5. *Δ' is locally unary.*

Proof. If a state is of the form $X\gamma$ with $X \in Var$ we must have $X\gamma \xrightarrow{\#}$ and nothing else. Otherwise, if it is of the form $X_a\gamma$ we can only have $X_a\gamma \xrightarrow{a}$ □

The following is now obvious from the definition of Δ'.

Proposition 6.6. *For $\alpha \in Var^*$ we have $\alpha \xrightarrow{a}_{\Delta} \alpha'\alpha''$ iff $\alpha \xrightarrow{\#}_{\Delta'} X_a \alpha'\alpha'' \xrightarrow{a}_{\Delta'} \alpha'\alpha''$.*

We therefore also see that

Proposition 6.7. *For $\alpha \in Var^*$ $b_1 b_2 ... b_n \in L(\alpha)$ relative to Δ iff $\#b_1\#b_2...\#b_n \in L(\alpha)$ relative to Δ'.*

Theorem 6.8. *[21] Failure and ready equivalence are undecidable for locally unary normed BPA processes.*

Proof. From Proposition 6.7 we can reduce language equivalence to language equivalence for locally unary normed processes and the theorem now follows from Lemma 6.3. □

The above ideas can also be used to prove that failure trace and ready trace equivalence are undecidable. For normed BPA, failure trace equivalence [13] coincides with the notion of refusal testing [29].

Definition 6.9. The *refusal relation* $\xrightarrow{A}$ for $A \subseteq Act$ is defined for any processes p, q by $p \xrightarrow{A} q$ iff $p = q$ and whenever $a \in A$, $p \stackrel{a}{\not\longrightarrow}$. The *failure trace* relations $\xrightarrow{u}$ for $u \in (Act \cup \mathcal{P}(Act))^*$ are defined as the reflexive and transitive closure of the refusal and transition relations. Define

$$\textit{failure-traces}(p) = \{u \in (Act \cup \mathcal{P}(Act))^* \mid \exists p' : p \xrightarrow{u} p'\}.$$

Two processes p and q are *failure-trace equivalent*, written $p \sim_{\text{ftr}} q$ iff $\textit{failure-traces}(p) = \textit{failure-traces}(q)$.

Lemma 6.10. *If p and q are locally normed unary processes then $p \sim_{\text{ftr}} q$ iff $L(p) = L(q)$.*

Proof. The 'only if' direction is straightforward. We only show the 'if' direction. Define $h : (Act \cup \mathcal{P}(Act)) \to Act^*$ as the homomorphic extension of

$$h(u) = \begin{cases} \epsilon & \text{if } u \in \mathcal{P}(Act) \\ u & \text{otherwise.} \end{cases}$$

Then, if $u \in \textit{failure-traces}(p)$ and p normed, clearly for some $v \in Act^*$, $h(u)v \in L(p)$. Thus, since $p \sim_{ctr} q$ we must have $h(u)v \in L(q)$ and since q is locally unary and normed, it is now easy to see that $u \in \textit{failure-traces}(q)$. □

Corollary 6.11. *Failure trace equivalence is undecidable for locally unary normed BPA processes.*

The definition of ready trace equivalence that is used here is a characterisation presented in [13].

Definition 6.12. Define

$$ready\text{-}trace(p) = \{A_0 a_1 A_1 \ldots a_n A_n \mid \exists p_0, \ldots, p_n : p = p_0 \xrightarrow{a_1} p_1 \cdots \xrightarrow{a_n} p_n,\ p_i \xrightarrow{a} \iff a \in A_i, 0 \leq i \leq n\}.$$

Two processes p and q are *ready trace equivalent*, written $p \sim_{rtr} q$, iff $ready\text{-}trace(p) = ready\text{-}trace(q)$.

Lemma 6.13. *If p and q are locally unary processes then $p \sim_{rtr} q$ iff $L(p) = L(q)$.*

Proof. The 'only if' direction is straightforward. We only show the 'if' direction. First note that if $A_0 a_1 A_1 a_2 \ldots a_n A_n$ is a ready trace for a locally unary process, all A_i $(0 \leq i < n)$ are singleton sets and A_n is the empty or a singleton set. Now define $h : (Act \cup \mathcal{P}(Act))^* \to Act^*$ as in Lemma 6.10. Then if $u \in ready\text{-}trace(p)$ we have, as p is normed, a $v \in Act^*$ such that $h(u)v \in L(p)$. Since $L(p) = L(q)$ we must have $h(u)v \in L(q)$, and since q is locally unary we get $u \in ready\text{-}trace(q)$. □

Corollary 6.14. *Ready trace equivalence is undecidable for locally unary normed BPA processes.*

7 Ready-simulation or 2/3-bisimulation

The notion of ready simulation (or 2/3-bisimulation) originated in work by Bloom, Istrail and Meyer [5] and Larsen and Skou [24]. It is the completed trace and the trace congruence induced by the GSOS-format [5].

Definition 7.1. A relation R between processes is a *ready simulation* iff it is a simulation and whenever pRq then for each $a \in Act$ we have $p \xrightarrow{a}$ if $q \xrightarrow{a}$. We say that q ready simulates p, written $p \subsetneq\!\!\!\rightarrow_r q$, iff there is a ready simulation R with pRq. Two processes p and q are *ready simulation equivalent*, written $p \leftrightarrows_r q$, iff $p \subsetneq\!\!\!\rightarrow_r q$ and $q \subsetneq\!\!\!\rightarrow_r p$.

The idea behind the proof is to find a class of processes where the ready simulation and simulation preorders coincide. The following class of processes is essential here:

Definition 7.2. A process p is said to be *two-step deterministic* iff whenever $p \xrightarrow{w} p_1 \xrightarrow{b}$ and $p \xrightarrow{w} p_2 \xrightarrow{b}$ then $p_1 = p_2$

Note that the notion of being two-step deterministic is strictly weaker than that of being deterministic.

We now show that for locally unary, two step deterministic processes language inclusion coincides with ready simulation. We use the construction of Definition 6.4 to show that language inclusion for deterministic processes can be reduced to language inclusion for unary locally, two-step deterministic processes. This enables us to show that ready simulation is undecidable for locally unary, two step deterministic processes.

The following two results follow immediately from Propositions 6.6 and 6.7.

Proposition 7.3. *If Δ is deterministic, then Δ' is two-step deterministic.*

Lemma 7.4. *Let Δ and Δ' be given as in Definition 6.4. Then for $\alpha, \beta \in Var^*$ $L(\alpha) \subseteq L(\beta)$ in Δ iff $L(\alpha) \subseteq L(\beta)$ in Δ'.*

Proof. Direct from Proposition 6.7. □

The next lemma states the correspondence between simulation and ready-simulation that we are seeking.

Lemma 7.5. *Let Δ define a normed BPA process. If Δ is locally unary and two-step deterministic, then for any $\alpha, \beta \in Var^*$ we have $L(\alpha) \subseteq L(\beta)$ iff $\alpha\surd \subsetneq_r \beta\surd$ (where $\surd$ is a new action not occurring in Act).*

Proof. The 'if' half is obvious, so it suffices to prove the 'only if' half. We define the relation

$$R = \{(\alpha\surd, \beta\surd) \mid L(\alpha) \subseteq L(\beta)\}$$

and show that it is a ready simulation. This is easy for the pair $(\surd, \surd)$. So we only consider pairs $(\alpha\surd, \beta\surd)$ where $\alpha, \beta \neq \epsilon$.

First we show that if $\beta\surd \xrightarrow{a}$ then $\alpha\surd \xrightarrow{a}$. So, assume $\beta\surd \xrightarrow{a}$. First observe, as α is normed, that $\alpha\surd \xrightarrow{b_1 \cdots b_n \surd}$ for some $n > 0$. As $L(\alpha) \subseteq L(\beta)$, $\beta\surd \xrightarrow{b_1 \cdots b_n \surd}$. As β is locally unary, it must be that $a = b_1$. Hence, $\alpha\surd \xrightarrow{a}$.

Now we show that R is a simulation relation. Assume $(\alpha\surd, \beta\surd) \in R$ and $\alpha\surd \xrightarrow{a} \alpha'$. There is exactly one action b such that $\alpha' \xrightarrow{b}$. If $b = \surd$ then $\alpha' = \surd$. Moreover, there is some β' such that $\beta\surd \xrightarrow{a} \beta' \xrightarrow{\surd}$. Clearly $\beta' = \surd$, so $(\alpha', \beta') \in R$. If $b \neq \surd$ then $\alpha' = \alpha''\surd$ for some α'' and

$$L(\alpha''\surd) = \{bw \mid abw \in L(\alpha\surd), w \in (Act^*)\surd\}.$$

As $\beta\surd$ is two step deterministic, there is exactly one β' such that $\beta\surd \xrightarrow{a} \beta' \xrightarrow{b}$, $\beta' = \beta''\surd$ and $L(\beta''\surd) = \{bw \mid abw \in L(\beta\surd)\}$. As clearly $L(\alpha''\surd) \subseteq L(\beta''\surd)$ it follows that $(\alpha''\surd, \beta''\surd) \in R$. □

We now have the following:

Theorem 7.6. *Ready simulation is undecidable for locally unary, two-step deterministic and normed BPA processes.*

Proof. We reduce language containment for deterministic processes to ready simulation for locally unary, two step deterministic processes. Given a deterministic Δ, let $\alpha, \beta \in Var^*$. We now have the following (strongly relying upon Lemmas 7.4 and 7.5):

$$\begin{array}{lll} L(\alpha) \subseteq L(\beta) \text{ in } \Delta & \text{iff} & L(\alpha) \subseteq L(\beta) \text{ in } \Delta' \\ & \text{iff} & \alpha\surd \subsetneq_r \beta\surd \text{ in } \Delta' \end{array}$$

Here $\surd$ is a new action. □

Theorem 7.7. *Ready simulation equivalence is undecidable for locally unary normed BPA processes.*

Proof. We reduce ready simulation to ready simulation equivalence by the following observation:

$$\alpha \underrightarrow{\subset}_r \beta \quad \text{iff} \quad \alpha + \beta \rightleftarrows_r \beta.$$

□

Acknowledgements. We would like to thank Didier Caucal for useful discussions and for pointing out several major mistakes in an earlier version of this paper. Also thanks to Rob van Glabbeek, Kim Larsen and Colin Stirling for useful discussions and comments.

References

[1] S. Abramsky. Observational equivalence as a testing equivalence. *Theoretical Computer Science*, 53:225–241, 1987.

[2] J.A. Bergstra and J.W. Klop Process algebra for synchronous communication *Information and Computation*, 60:109–137, 1984.

[3] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Proceedings PARLE conference,* Eindhoven*, Vol. II (Parallel Languages)*, volume 259 of *Lecture Notes in Computer Science*, pages 94–113. Springer-Verlag, 1987.

[4] J.A. Bergstra, J.W. Klop, and E.-R. Olderog. Readies and failures in the algebra of communicating processes. *SIAM J. on Comput.*, 17:1134–1177, 1988.

[5] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. Technical Report 90-1150, Department of Computer Science, Cornell University, Ithaca, New York, August 1990.

[6] S.D. Brookes, C.A.R. Hoare, and W. Roscoe. A theory of communicating sequential processes. *JACM*, 31:560–599, 1984.

[7] D. Caucal. Graphes canoniques de graphes algébriques. *Theoretical Informatics and Applications* 24:339–352, 1990.

[8] S. Christensen. Decidability and Decomposition in Process Algebras. Forthcoming Ph.D. thesis.

[9] S. Christensen, Y. Hirschfeld, F. Moller. Decomposability, decidability, and axiomatisability for bisimulation equivalence on basic parallel processes. To appear in *Proceedings* 8^{th} *Annual Symposium on Logic in Computer Science,* Montreal, Canada. IEEE, 1993. Also published as LFCS Report ECS-LFCS-92-244, University of Edinburgh, 1992.

[10] S. Christensen, H. Hüttel, C.P. Stirling. Bisimilarity is decidable for all context-free processes. In R. Cleaveland, editor, *Proceedings of CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*, pages 138–147, 1992.

[11] J. Engelfriet Determinacy $\rightarrow$ (observation equivalence = trace equivalence) *Theoretical Computer Science* 36:21–25, 1985.

[12] E.P. Friedman. The inclusion problem for simple languages. *Theoretical Computer Science*, 1:297–316, 1976.

[13] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.

[14] J.F. Groote. Transition system specifications with negative premises. Report CS-R8950, CWI, 1989. An extended abstract appeared in J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90*, Amsterdam, LNCS 458, pages 332–341. Springer-Verlag, 1990.

[15] J.F. Groote. A short proof of the decidibility of bisimulation for normed BPA-processes. *Information Processing Letters*, 42:167–171, 1992.

[16] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence (extended abstract). In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Del la Rocca, editors, *Proceedings of ICALP89*, volume 372 of *LNCS*, pages 423–438. Springer-Verlag, 1989. Full version to appear in *Information and Computation.*

[17] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, Cambridge, Massachusetts, 1988.

[18] J. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[19] H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, The Netherlands, pages 376–386. IEEE Computer Society Press, 1991.

[20] H. Hüttel. Dedidability, Behavioural Equivalences and Infinite Transition Graphs Ph.D. thesis. Report ECS-LFCS-91-191, University of Edinburgh, 1991.

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

[22] P.C. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[23] A.J. Korenjak and J.E. Hopcroft Simple Deterministic Languages In *Proc. Seventh Annual IEEE Symposium on Switching and Automata Theory*, pages 36–46, 1966.

[24] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Proceedings* 16^{th} *ACM Symposium on Principles of Programming Languages,* Austin, Texas, pages 344–352, 1989.

[25] Milner, R. A Calculus of Communicating Systems Springer-Verlag LNCS 92, 1980.

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

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

[28] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI Conference LNCS 104*, pages 167–183. Springer, 1981.

[29] I.C.C. Philips. Refusal testing. *Theoretical Computer Science*, 50:241–284, 1987.

[30] W.S. Rounds and S.D. Brookes. Possible futures, acceptances, refusals and communicating processes. In *Proc. 22nd Annual Symposium on Foundations of Computer Science*, pages 140–149, New York, 1981. IEEE.