Deciding Bisimilarity between BPA and BPP Processes[*]

Petr Jančar[1], Antonín Kučera[2], and Faron Moller[3]

[1] Dept. of Computer Science, FEI, Technical University of Ostrava,
17. listopadu 15, 70833 Ostrava, Czech Republic.
<Petr.Jancar@vsb.cz>
[2] Faculty of Informatics, Masaryk University,
Botanická 68a, 60200 Brno, Czech Republic,
<tony@fi.muni.cz>
[3] Dept. of Computer Science, University of Wales Swansea,
Singleton Park, Swansea SA2 8PP, Wales.
<F.G.Moller@swansea.ac.uk>

Abstract. We identify a necessary condition for when a given BPP process can be expressed as a BPA process. We provide an effective procedure for testing if this condition holds of a given BPP, and in the positive case we provide an effective construction for a particular form of one-counter automaton which is bisimilar to the given BPP. This in turn provides the mechanism to decide bisimilarity between a given BPP process and a given BPA process.

1 Introduction

During the last decade, a great deal of research effort has been devoted to the study of decidability and complexity issues for checking semantic equivalences, in particular bisimilarity, between various classes of processes. There have been several surveys presenting this work (eg, [16, 13, 12]), including a major Handbook chapter [3]. There is even now a project devoted to maintaining an up-to-date comprehensive overview of the state-of-the-art in this dynamic research topic [19].

Example classes of processes of particular interest in this study are pushdown automata, Petri nets, and the process algebra PA. Some milestones in the study, beginning with the decidability of bisimilarity over normed BPA [1] include the undecidability of bisimilarity over Petri nets [10]; the decidability of bisimilarity over normed PA [7]; and the decidability of bisimilarity over the class of strict deterministic grammars (a particular formulation of deterministic pushdown automata) [23]. This final result reinforces Sénizergues' solution [18] to the long-standing equivalence problem for deterministic pushdown automata. A closely related result is the decidability of bisimilarity over state-extended BPA [17, 22].

The motivation for the present study is to work towards a generalization of the above decidability result for normed PA to the whole class of PA processes. The process algebra PA includes operators for composing terms both sequentially and in parallel, and as

[*] The first two authors are supported by the Grant Agency of the Czech Republic, grant No. 201/03/1161.

described by Hirshfeld and Jerrum in [7], there are surprising interactions between sequential and parallel compositions. Indeed, one can express the sequential composition $X_1 \cdot X_2$ of two terms X_1 and X_2 as a parallel composition $Y_1 || Y_2$ of two other terms Y_1 and Y_2 in infinitely-many ways, using terms of unbounded complexity. By restricting to normed process terms, Hirshfeld and Jerrum were able to develop a structural theory which allowed them, in effect, to finitely characterize the infinite set of solutions to the equivalence $X_1 \cdot X_2 = Y_1 || Y_2$, and then use this characterization to provide their decidability result. However, it remains open as to how to extend their techniques to the unnormed PA case.

The process class BPA represents the subset of PA involving only sequential composition, while BPP represents the subset involving only parallel composition. As such, in light of the above observations, it becomes natural to consider the problem of comparing an arbitrary BPA term with an arbitrary BPP term. Decidability between such a pair of terms in the normed case of course follows from the above result, though this problem was already settled in [2, 5].

Bisimulation checking over normed process classes has typically proven to be far more tractable than over unnormed processes. For example, over both the BPA and BPP process classes, unique decomposability results in the spirit of [15] hold over the normed subclasses which allow for polynomial decision procedures in each case [8, 9]. Though decidability of bisimilarity in the unnormed cases has been known for some time, bounds on their complexity have been elusive. Recently both problems have been shown to be PSPACE hard [20, 21], and even more recently the problem for BPP has been shown to be PSPACE-complete [11]. Various novel techniques are developed in each of the above papers which contribute towards an understanding of the nature of these classes of sequential and parallel process.

In this paper we continue the exposition of these classes of processes and consider the problem of when an arbitrary BPP process can be expressed as a BPA process term. To this end, we identify a property of (arbitrary) processes which cannot be modelled "sequentially"; essentially this property entails encoding two distinct, unrelated and sufficiently-large integer values. If our given BPP process can be expressed as a BPA term, then clearly it is necessary that the process does not possess this property. Furthermore, we demonstrate how to test if a given BPP process possesses this property, and in the case that it does not, we provide an effective construction of an equivalent one-counter automaton. As one-counter automata and BPA both constitute subclasses of state-extended BPA, we arrive at the decidability of bisimilarity between BPA and BPP processes from the afore-mentioned decidability of bisimilarity over state-extended BPA.

The structure of the paper is as follows. In Section 2 we present various preliminary definitions, and in Section 3 we explore the structure of BPA processes and provide the crucial technical result of the paper. Finally, in Section 4 we prove our decidability result by characterizing when a BPP process abides by the structural restrictions of BPA processes exposed in Section 3, and then demonstrating how to decide equivalence to a given BPA process in the case where this is true.

2 Preliminary Definitions and Results

2.1 Processes and Norms

Formally, a *process* is represented by (a state in) a *labelled transition system* defined as follows.

Definition 1. *A* labelled transition system (LTS) *is a triple $\mathcal{S} = (S, Act, \rightarrow)$ where S is a set of* states*, Act is a finite set of* actions*, and $\rightarrow \subseteq S \times Act \times S$ is a* transition relation.

We write $s \xrightarrow{a} \bar{s}$ instead of $(s, a, \bar{s}) \in \rightarrow$ and we extend this notation to elements of Act^* in the natural way. We also use $s \rightarrow \bar{s}$ to mean $s \xrightarrow{a} \bar{s}$ for some $a \in Act$. A state $\bar{s}$ is *reachable* from a state s if $s \rightarrow^* \bar{s}$, that is, if $s \xrightarrow{w} \bar{s}$ for some $w \in Act^*$.

The notion of "behavioural sameness" between two processes can be formally captured in many different ways (see, e.g., [6] for an overview). Among those behavioural equivalences, *bisimulation equivalence* enjoys special attention. Its formal definition is as follows.

Definition 2. *Let $\mathcal{S} = (S, Act, \rightarrow_{\mathcal{S}})$ and $\mathcal{T} = (T, Act, \rightarrow_{\mathcal{T}})$ be transition systems defined over the same action set Act. A binary relation $\mathcal{R} \subseteq S \times T$ is a* bisimulation relation *iff whenever $(s, t) \in R$, we have that*

- *for each transition $s \xrightarrow{a}_{\mathcal{S}} \bar{s}$ there is a transition $t \xrightarrow{a}_{\mathcal{T}} \bar{t}$ such that $(\bar{s}, \bar{t}) \in \mathcal{R}$; and*
- *for each transition $t \xrightarrow{a}_{\mathcal{T}} \bar{t}$ there is a transition $s \xrightarrow{a}_{\mathcal{S}} \bar{s}$ such that $(\bar{s}, \bar{t}) \in \mathcal{R}$.*

Processes s and t are bisimulation equivalent (bisimilar)*, written $s \sim t$, iff they are related by some bisimulation.*

An important subclass of processes are the *normed* processes, which are those for which from any state there is a sequence of transitions leading to a state having no transitions leading out of it; the *norm* of a process state is then traditionally defined to be the length of a shortest sequence of transitions leading to such a deadlocked state. We can generalize the notion of a norm as follows. (We let $\mathbb{N} = \{0, 1, 2, \ldots\}$ represent the set of natural numbers, $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$, and $\mathbb{N}_{\omega,-1} = \mathbb{N} \cup \{\omega, -1\}$, where $\omega + \omega = \omega + k = \omega - k = \omega - \omega = \omega$, and $k < \omega, \omega \leq \omega$ for every $k \in \mathbb{N} \cup \{-1\}$.)

Definition 3. *Let $\mathcal{S} = (S, Act, \rightarrow)$ be a transition system, and $d : S \rightarrow \mathbb{N}_\omega$ a function. We say that d is a* norm *iff for all $s \in S$ we have the following:*

- *If $s \rightarrow s'$, then $d(s') \geq d(s){-}1$; and*
- *If $0 < d(s) < \omega$, then there is $s \rightarrow s'$ such that $d(s') = d(s){-}1$.*

*In the latter clause, we call such a transition a d-*reducing transition.

It is possible to construct new norms out of already existing ones; for example, if d and d' are norms, then so is $\min(d, d')$. For our purposes, the following construction is particularly important. Firstly, for all $s, s' \in S$ we define the *distance* from s to s', denoted $dist(s,s')$, to be the length of a shortest sequence of transitions leading from state s to state s':

$$dist(s,s') \;=\; \min\left\{\, length(w) : s \xrightarrow{w} s' \,\right\}.$$

Adhering to the convention that $\min\emptyset = \omega$, we note that $dist(s,s') = \omega$ when there is no such sequence.

Given a tuple of norms $\mathcal{F} = (d_1,\ldots,d_k)$, each transition $s \xrightarrow{a} s'$ determines a unique *change* of $\mathcal{F}$, denoted $\delta^{\mathcal{F}}(s \xrightarrow{a} s')$, which is a k-tuple of values from $\mathbb{N}_{\omega,-1}$ defined by $\delta^{\mathcal{F}}(s \xrightarrow{a} s') = \big(d_1(s')-d_1(s),\ldots,d_k(s')-d_k(s)\big)$. For each triple $(a,\mathcal{F},\delta)$, where $a \in Act$, $\mathcal{F} = (d_1,\ldots,d_k)$ is a tuple of norms, and $\delta \in \mathbb{N}^k_{\omega,-1}$, we define the function $dd_{(a,\mathcal{F},\delta)} : S \to \mathbb{N}_\omega$ to be the distance to a state for which all norms of $\mathcal{F}$ are finite, and for which there is no a-transition with the change δ:

$$\begin{aligned} dd_{(a,\mathcal{F},\delta)}(s) = \min\big\{\, dist(s,s') :\; & d_i(s') \neq \omega \text{ for all } i, \text{ and} \\ & \delta^{\mathcal{F}}(s' \xrightarrow{a} s'') \neq \delta \text{ for all } s' \xrightarrow{a} s'' \,\big\}. \end{aligned}$$

Obviously, each such $dd_{(a,\mathcal{F},\delta)}$ is a norm.

Some norms are not semantically relevant in the sense that bisimilarity does not necessarily preserve them. In this paper we are mainly interested in *bisimulation-invariant* norms.

Definition 4. *We say that a given norm d is* bisimulation-invariant *if $s \sim s'$ implies $d(s) = d(s')$.*

A simple example of a bisimulation-invariant norm is the function dd_a (where $a \in Act$) defined as follows:

$$dd_a(s) = \min\left\{\, dist(s,s') : s' \overset{a}{\not\rightarrow} \,\right\}.$$

Definition 5. *The set of* DD-functions *is defined inductively as follows:*

- *dd_a is a DD-function for every $a \in Act$;*
- *if $\mathcal{F} = (d_1,\cdots,d_k)$ is a tuple of DD-functions, $\delta \in \mathbb{N}^k_{\omega,-1}$, and $a \in Act$, then $dd_{(a,\mathcal{F},\delta)}$ is also a DD-function.*

A simple observation is that if $d_1,\cdots,d_k$ are bisimulation-invariant norms, then $dd_{(a,\mathcal{F},\delta)}$ is also a bisimulation-invariant norm for each triple $(a,\mathcal{F},\delta)$. This in turn implies that all DD-functions are bisimulation-invariant. For each DD-function d we further define the sets $\mathcal{D}(d)$ and $\mathcal{C}(d)$ of all DD-functions and changes which are employed during the construction of d:

- $\mathcal{D}(dd_a) = \mathcal{C}(dd_a) = \emptyset$ for every action a;
- if $d = dd_{(a,\mathcal{F},\delta)}$, where $\mathcal{F} = (d_1,\cdots,d_k)$, then
  - $\mathcal{D}(d) = \mathcal{D}(d_1) \cup \cdots \cup \mathcal{D}(d_k) \cup \{d_1,\cdots,d_k\}$,
  - $\mathcal{C}(d) = \mathcal{C}(d_1) \cup \cdots \cup C(d_k) \cup \{\delta\}$.

2.2 BPA, BPP, Petri Nets and One Counter Automata

A BPA process is defined by a context-free grammar in Greibach normal form. Formally this is given by a triple $G = (V, A, \Gamma)$, where V is a finite set of *variables* (*nonterminal symbols*), A is a finite set of *labels* (*terminal symbols*), and $\Gamma \subseteq V \times A \times V^*$ is a finite set of *rewrite rules* (*productions*); it is assumed that every variable has at least one associated rewrite rule. Such a grammar gives rise to the LTS $\mathcal{S}_G = (V^*, A, \rightarrow)$ in which the states are sequences of variables, the actions are the labels, and the transition relation is given by the rewrite rules extended by the *prefix rewriting rule*: if $(X, a, \alpha) \in \Gamma$ then $X\beta \xrightarrow{a} \alpha\beta$ for all $\beta \in V^*$. In this way, concatenation of variables naturally represents sequential composition.

A BPP process is defined in exactly the same fashion from such a grammar. However, in this case elements of V^* are read modulo commutativity of concatenation, so that concatenation is interpretted as parallel composition rather than sequential composition. The states of the BPP process associated with a grammar are thus given not by sequences of variables but rather by multisets of variables.

In either case, BPA or BPP, the usual notion of the *norm* of a state $\alpha \in V^*$, denoted $|\alpha|$, is the length of a shortest path to the empty process ε: $|\alpha| = dist(\alpha, \varepsilon)$. If all variables of the underlying grammar have finite norm, then the process is said to be *normed*; otherwise it is *unnormed*.

As an example, Figure 1 depicts the (normed) BPA and BPP processes defined by

$$\begin{tikzcd}
A \arrow[r, "a"] \arrow[d, "c"] & AB \arrow[r, "a"] \arrow[d, "c"] & ABB \arrow[r, "a"] \arrow[d, "c"] & \cdots \\
\varepsilon & B \arrow[l, "b"] & BB \arrow[l, "b"] & \cdots \arrow[l, "b"]
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, shift left, "a"] \arrow[d, "c"] & AB \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] \arrow[d, "c"] & ABB \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] \arrow[d, "c"] & \cdots \arrow[l, shift left, "b"] \\
\varepsilon & B \arrow[l, "b"] & BB \arrow[l, "b"] & \cdots \arrow[l, "b"]
\end{tikzcd}$$

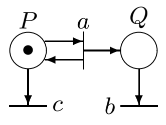
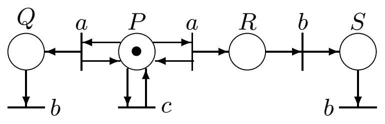


Fig. 1. BPA and BPP processes defined by the grammar $A \xrightarrow{a} AB$, $A \xrightarrow{c} \varepsilon$, $B \xrightarrow{b} \varepsilon$, along with a BPP net equivalent to the BPP.

the same grammar given by the three rules $A \xrightarrow{a} AB$, $A \xrightarrow{c} \varepsilon$ and $B \xrightarrow{b} \varepsilon$. It can easily be shown [3] that the BPA process cannot be expressed by any BPP process, and equally the BPP process cannot be expressed by any BPA process. Figure 2 on the other hand

$$\begin{array}{lll} A \xrightarrow{a} BA & A \xrightarrow{a} BBA & A \xrightarrow{c} A \\ B \xrightarrow{a} BB & B \xrightarrow{a} BBB & B \xrightarrow{c} B \\ & & B \xrightarrow{b} \varepsilon \end{array}
\qquad\qquad
\begin{array}{lll} P \xrightarrow{a} PQ & P \xrightarrow{a} PR & P \xrightarrow{c} P \\ Q \xrightarrow{b} \varepsilon & R \xrightarrow{b} S & S \xrightarrow{b} \varepsilon \end{array}$$

$$\begin{tikzcd}
A \arrow[r, shift left, "a"] \arrow[rr, bend left, "a"] \arrow[loop below, "c"] & BA \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] \arrow[rr, bend left, "a"] \arrow[loop below, "c"] & BBA \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] \arrow[r, bend left, "a"] \arrow[loop below, "c"] & \cdots \arrow[l, shift left, "b"]
\end{tikzcd}$$



$\mathcal{R} = \left\{ (PQ^iR^jS^k, B^{i+2j+k}A) : i, j, k \geq 0 \right\}$ is a bisimulation showing that $P \sim A$.

Fig. 2. A BPA process, and an equivalent BPP process with its associated BPP net.

depicts an example of an (unnormed) process which is definable both as a BPA process and as a BPP process.

An equivalent formulation of BPP, and one which we adopt to aid in distinguishing between BPA and BPP processes, is as labelled Petri nets in which each transition has a unique input place. Formally, a labelled Petri net is a tuple $\mathcal{N} = (P, T, F, A, \ell)$ where P and T are finite and disjoint sets of *places* and *transitions*, respectively, $F : (P \times T \cup T \times P) \rightarrow \mathbb{N}$ is a *flow function*, A is a set of *labels*, and $\ell : T \rightarrow A$ is a *labelling*. A *marking* is a function $M : P \rightarrow \mathbb{N}$ which associates to each place a finite number of *tokens*. A transition t is *enabled* at a marking M if $M(p) \geq F(p,t)$ for each place p. If t is enabled at M, it can be *fired* from M, producing a new marking M' defined by $M'(p) = M(p) - F(p,t) + F(t,p)$. This is written as $M \xrightarrow{t} M'$. To every labelled Petri net $\mathcal{N}$ we associate a transition system where the set of states is the set of all markings, A is the set of all labels, and $M \xrightarrow{a} M'$ iff there is transition t such that $\ell(t) = a$ and $M \xrightarrow{t} M'$.

Petri nets are often depicted as graphs with two kinds of nodes (corresponding to places and transitions) where the flow function is indicated by (multiple) arcs between places and transitions. A *BPP net* is a Petri net where for every $t \in T$ there is exactly one place $Pre(t)$ such that $F(Pre(t), t) = 1$ and $F(p,t) = 0$ for every other place p. The equivalence of these BPP nets to BPP is easily seen from the Petri net presented in Figures 1 and 2.

We shall find the following Petri net concepts useful.

Definition 6. *For a set Q of places of a Petri net, we define* $\text{NORM}(Q)(M)$ *to be the length of a shortest sequence of transitions from M which leaves all of the places of Q empty:*

$$\text{NORM}(Q)(M) \;=\; \min\left\{\, dist(M, M') : M'(p) = 0 \textit{ for every } p \in Q \,\right\}.$$

We may readily observe that for every set Q of places, $\text{NORM}(Q)$ is a norm in the sense of Definition 3. In the case of BPP nets, $\text{NORM}(Q)(M)$ is just a weighted sum of tokens in M; that is, to each place p we can effectively associate some $c_p \in \mathbb{N}_\omega$ (which depends only on Q) so that $\text{NORM}(Q)(M) = \sum_{p \in P} c_p \cdot M(p)$ for every marking M.

Definition 7. *A set Q of places of a Petri net is called a* trap *iff for all transitions t we have that if $\sum_{p \in Q} F(p,t) > 0$ then $\sum_{p \in Q} F(t,p) > 0$.*

Thus a "marked" trap, that is, a trap containing at least one token, can never become unmarked. This then implies that for any trap Q, $\text{NORM}(Q)(M)$ is either 0 or ω.

We can note that $\emptyset$ is a trap, and that the union of traps is again a trap. This justifies the following definition.

Definition 8. $\text{MAXTRAP}(Q)$ *denotes the maximal trap contained in the set of places Q.*

Finally, we shall have cause to consider the following class of one-counter automata.

Definition 9. *A* one-counter automaton with resets *(OCR) is a tuple $\mathcal{P} = (Q, A, I, Z, \delta)$ where Q is a finite set of* control states, *A is a finite* input alphabet, *I and Z are* counter symbols *and δ is a finite set of* rules *which are of one of the following three forms:*

– $pZ \xrightarrow{a} qI^kZ$ *where* $p, q \in Q$*,* $a \in A$*, and* $k \in \mathbb{N}$*; these rules are called* zero *rules.*
– $pI \xrightarrow{a} qI^k$ *where* $p, q \in Q$*,* $a \in A$*, and* $k \in \mathbb{N}$*; these rules are called* positive *rules.*
– $pI \xrightarrow{a} qZ$ *where* $p, q \in Q$ *and* $a \in A$*; these rules are called* resets*.*

Hence, Z acts as a bottom symbol (which cannot be removed), and the number of I's which are stored in the stack above the topmost occurrence of Z represents the counter value. The reset (i.e., setting the counter back to zero) is implemented by pushing the symbol Z onto the stack.

To the OCR $\mathcal{P}$ we associate the transition system $\mathcal{S}_{\mathcal{P}}$ where $Q \times \{I, Z\}^*$ is the set of states, A is the set of actions, and the transition relation is determined by

$$pX\alpha \xrightarrow{a} q\beta\alpha \text{ iff } pX \xrightarrow{a} q\beta \in \delta.$$

3 Prefix-Encoded Norms over BPA

In this section we demonstrate that large values of DD-functions are represented by large (normed) prefixes of BPA states. This will provide a necessary condition for when a BPP process can be bisimilar to a BPA process.

Definition 10. *We define the* pseudo-norm *(or* prefix-norm*)* $pn(\alpha)$ *of a BPA process* α *as follows:*

$$pn(\alpha) = \max \{ |\beta| : \alpha = \beta\gamma \text{ and } |\beta| < \omega \}.$$

We call the transition $X\beta \xrightarrow{a} \gamma\beta$ *a* pn-reducing step *iff* $|\gamma| = |X| - 1 < \omega$*.*

Note that pn is *not* a norm in the sense of Definition 3, and that a step $X\beta \xrightarrow{a} \gamma\beta$ such that $pn(\gamma\beta) = pn(X\beta) - 1$ is not necessarily pn-reducing.

Definition 11. *We say that a norm* d *is* prefix-encoded *for a given BPA process* G *iff there is a constant* $C \in \mathbb{N}$ *such that for every process* α *of* G *with* $C < d(\alpha) < \omega$*, the* d*-reducing steps from* α *are exactly the* pn*-reducing steps from* α*. In this case, we say that* d *is* prefix-encoded above C*.*

Given a BPA process, we define the value MAXSTEP as the maximal value $|\beta| - |X|$ where $X \xrightarrow{a} \beta$ is a rule with $|\beta| < \omega$. For any norm d we easily observe the following.

Proposition 12.

(a) $d(\alpha\beta) \leq |\alpha| + d(\beta)$.
(b) If $d(\alpha\beta) < |\alpha| + d(\beta)$ *then* $\alpha \rightarrow^* \alpha'$ *with* $d(\alpha'\beta) = 0$.
(c) If d *is prefix-encoded above* C *and* $d(\alpha\beta) \geq |\alpha| + C$ *then* $d(\alpha\beta) = |\alpha| + d(\beta)$*; and if* $\alpha \rightarrow \alpha'$ *with* $|\alpha'| < \omega$ *then* $d(\alpha\beta) - 1 \leq d(\alpha'\beta) \leq d(\alpha\beta) + \text{MAXSTEP}$.
(d) If $d(X\alpha) < \omega$ *and there is a transition from* $X\alpha$ *which is* d*-reducing or* pn*-reducing but not both, then there is a maximal sequence* $X\alpha \xrightarrow{w} \beta$ *of* d*-reducing transitions such that* $X \xrightarrow{w} \gamma$ *with* $\gamma \neq \varepsilon$*; this implies that* $X\alpha \xrightarrow{w} \gamma\alpha$ *and* $d(\gamma\alpha) = 0$.

Lemma 13. *For any BPA process, each DD-function is prefix-encoded.*

Proof. We assume a given BPA process, and show the claim by contradiction. We first define some technical notions. For a DD-function d, we say that two states α_1 and α_2 are (d, C)*-diff-large* (for $C \in \mathbb{N}$) iff for each $d' \in \{d\} \cup \mathcal{D}(d)$ such that $d'(\alpha_1) \neq d'(\alpha_2)$ we have the following inequalities:

$$C < d'(\alpha_1) < \omega;$$
$$C < d'(\alpha_2) < \omega; \quad \text{and} \quad |d'(\alpha_2) - d'(\alpha_1)| > C.$$

We say that the states α_1 and α_2 are d*-bad* iff for some $\gamma \neq \varepsilon$:

$$0 = d(\gamma\alpha_1) < d(\gamma\alpha_2) \quad \text{or}$$
$$0 = d(\gamma\alpha_2) < d(\gamma\alpha_1).$$

Claim 1. If d is not prefix-encoded, then for any $C \in \mathbb{N}$ there are states α_1 and α_2 which are (d, C)-diff-large and d-bad.

Proof of Claim 1. If d is not prefix-encoded, then there is a sequence of states $\beta_1, \beta_2, \beta_3, \ldots$ with $d(\beta_1) < d(\beta_2) < d(\beta_3) < \cdots$ such that each β_i can make a step which is d-reducing or pn-reducing but not both.

Using the pigeonhole principle, we can assume (i.e., extract a subsequence) $\beta_1 = X\alpha_1$, $\beta_2 = X\alpha_2$, $\beta_3 = X\alpha_3, \ldots$ for a variable X (obviously $|X| < \omega$). We can furthermore assume (by repeated subsequence extractions) that for each $d' \in \{d\} \cup \mathcal{D}(d)$ we have either $d'(\alpha_1) = d'(\alpha_2) = d'(\alpha_3) = \cdots$, or else $d'(\alpha_1) < d'(\alpha_2) < d'(\alpha_3) < \cdots$.

Hence, for any given C there are $i < j$ such that α_i and α_j are (d, C)-diff-large. From Proposition 12(d), and the fact that $d(X\alpha_i) < d(X\alpha_j)$, we easily derive that α_i and α_j are d-bad. ($\Box$)

We now let d be a non-prefix-encoded DD-function on a minimal level. Choose $C \in \mathbb{N}$ so that:

- each $d' \in \mathcal{D}(d)$ is prefix-encoded above $(C - z_0)$, where z_0 is the maximum of the finite components of changes in $\mathcal{C}(d)$;
- $C >$ MAXSTEP; and
- $C > d'(\beta)$ whenever $d' \in \mathcal{D}(d)$ and $X \to \beta$ with $d'(\beta) < \omega = |\beta|$.

We take $d_1 \in \{d\} \cup \mathcal{D}(d)$ on a minimal level such that we can choose α_1 and α_2 which are (d_1, C)-diff-large and d_1-bad (as guaranteed by Claim 1). Assume that $0 = d_1(\gamma\alpha_1) < d_1(\gamma\alpha_2)$. If $d'(\gamma\alpha_2) = \omega$ for some $d' \in \mathcal{D}(d_1)$, then $d'(\alpha_2) = \omega = d'(\alpha_1)$. Since $d'(\gamma\alpha_1) < \omega$, there is some β such that $0 = d'(\beta\alpha_1) < d'(\beta\alpha_2)$; but this means that α_1 and α_2 are d'-bad, which contradicts the level-minimality of d_1.

Thus for some $d' \in \mathcal{D}(d_1)$ and z being a component of a change in $\mathcal{C}(d)$ there is a step $\gamma \to \gamma'$ such that

$$d'(\gamma\alpha_1) + z \neq d'(\gamma'\alpha_1) \quad \text{and} \quad d'(\gamma\alpha_2) + z = d'(\gamma'\alpha_2)\,. \tag{1}$$

Claim 2. There is ξ such that $d'(\xi\alpha_1) \neq d'(\xi\alpha_2)$, and either $d'(\xi\alpha_1) < |\xi| + d'(\alpha_1)$ or $d'(\xi\alpha_2) < |\xi| + d'(\alpha_2)$.

Proof of Claim 2. Suppose that none of the γ and γ' from Equation (1) satisfies the claim. Since we cannot have that $|\gamma| + d'(\alpha_1) + z \neq |\gamma'| + d'(\alpha_1)$ and $|\gamma| + d'(\alpha_2) + z = |\gamma'| + d'(\alpha_2)$, it is sufficient to consider only the following cases:
(a) $d'(\gamma\alpha_1) = d'(\gamma\alpha_2)$ and $d'(\gamma'\alpha_1) \neq d'(\gamma'\alpha_2)$;
(b) $d'(\gamma\alpha_1) \neq d'(\gamma\alpha_2)$ and $d'(\gamma'\alpha_1) = d'(\gamma'\alpha_2)$.
For case (a): $d'(\gamma'\alpha_1) = |\gamma'| + d'(\alpha_1) \neq |\gamma'| + d'(\alpha_2) = d'(\gamma'\alpha_2)$ and $d'(\gamma\alpha_1) = d'(\gamma\alpha_2) = d'(\gamma'\alpha_2) - z$ (note that $z < \omega$). By Proposition 12(c) we get that $d'(\gamma'\alpha_1)$ and $d'(\gamma'\alpha_2)$ can differ by at most $\text{MAXSTEP}+1$, which is a contradiction.
For case (b): $d'(\gamma\alpha_1) = |\gamma| + d'(\alpha_1) \neq |\gamma| + d'(\alpha_2) = d'(\gamma\alpha_2)$. Proposition 12(c) implies that we cannot have $d'(\gamma'\alpha_1) = d'(\gamma'\alpha_2)$ unless the step $\gamma \to \gamma'$ is due to a rule $X \to \beta$ with $|\beta| = \omega$. But C was chosen bigger than $d'(\beta)$. (□)

Finally we show that the existence of a (d', C)-diff-large pair α_1 and α_2, together with a ξ satisfying Claim 2 contradicts the assumption that d' is prefix-encoded above C; this will finish the proof of the Lemma.

Without loss of generality, assume $d'(\xi\alpha_1) < d'(\xi\alpha_2)$. If $d'(\xi\alpha_1) < |\xi| + d'(\alpha_1)$ then there is ξ' such that $0 = d'(\xi'\alpha_1) < d'(\xi'\alpha_2)$, meaning α_1 and α_2 are d'-bad, which is a contradiction.

It remains to consider $d'(\xi\alpha_1) = |\xi| + d'(\alpha_1) < d'(\xi\alpha_2) < |\xi| + d'(\alpha_2)$. Since necessarily $d'(\alpha_1) < d'(\alpha_2)$, we have $d'(\xi\alpha_2) > |\xi| + C$, so by Proposition 12(c), $d'(\xi\alpha_2) = |\xi| + d'(\alpha_2)$, which again is a contradiction. □

4 Bisimilarity is decidable on the union of BPA and BPP

In this section we show that we can decide whether a given BPP process M_0 satisfies a necessary condition for being bisimilar to some unspecified BPA process. In the positive case we can (effectively) construct an OCR process which is bisimilar to M_0. So the decidability of the question whether $M_0 \sim \alpha_0$ (where α_0 is a BPA process) follows from the results of [17, 22].

We first recall some useful results from [11] which clarify the "bisimilarity state space" for BPP processes; Firstly, by inspection of [11] we can confirm the following.

Lemma 14. *For each BPP net $\mathcal{N}$ we can effectively construct a sequence $Q_1, \ldots, Q_m$ of sets of places which are* important *in the sense that their norms capture bisimilarity:*

$$\forall M, M' : M \sim M' \quad \textit{iff} \quad \forall i : \text{NORM}(Q_i)(M) = \text{NORM}(Q_i)(M').$$

In fact, the collection of all $\text{NORM}(Q_i)$, $1 \leq i \leq m$, is exactly the set of all DD-functions over the state-space of $\mathcal{N}$. More precisely, for every DD-function d there is some Q_i such that $d(M) = \text{NORM}(Q_i)(M)$ for every marking M of $\mathcal{N}$. Conversely, to every Q_i one can associate a DD-function d_i so that all elements of $\mathcal{D}(d_i)$ are among the functions associated to $Q_1, \ldots, Q_{i-1}$.

We now explore further related technical notions. Let $\mathcal{N}$ be a labelled Petri net with initial marking M_0. Given $c \in \mathbb{N}_\omega$, we say that places p and q of $\mathcal{N}$ are *c-dependent* (for M_0) if for every reachable marking M we have that if $c < M(p)$ and $c < M(q)$, then $M(p) = M(q)$. Note that p and q are trivially ω-dependent (for every M_0). The *dependence level* of p, q (for M_0) is the least $c \in \mathbb{N}_\omega$ such that p and q are c-dependent for M_0.

Lemma 15. *Let $\mathcal{N}$ be a Petri net, M_0 a marking of $\mathcal{N}$, and p, q places of $\mathcal{N}$. The dependence level of p, q for M_0 is effectively computable.*

Proof. The dependence level of p, q can be computed, e.g., by employing a slightly modified version of the algorithm for constructing the coverability tree for M_0 [14]. We briefly sketch the construction, emphasizing the difference from the standard algorithm.

An *extended marking* is a function $M : P \to \mathbb{N}_\omega$. All notions introduced for "ordinary" markings also apply to extended markings by employing the standard ω-conventions introduced in Section 2.1. The goal is to compute a finite tree where nodes are labelled by extended markings such that the dependence level of p, q for M_0 can be "read" from the tree. It is also possible that the algorithm terminates earlier (without constructing the whole tree) and outputs ω. This happens if the part of the tree constructed so far exhibits a "pumpable" sequence of transitions witnessing the infinity of the dependence level. To simplify our notation, we introduce the following notion: Let n, n' be nodes of the tree labelled by M, M' such that n' is a descendant of n. We say that a place s is *pumped* at n' from n by k, where $0 < k < \omega$, iff $M' \geq M$ and $M'(s) - M(s) = k$. Furthermore, s is *pumpable* at n' iff s is pumped at n' from some predecessor of n' by some (positive) value.

Initially, we put M_0 to be the (label of the) root of the tree. Then, for every node n labelled by M which has not yet been processed we do the following: If the tree contains a processed node with the same label, then the node n is immediately declared as processed. Otherwise, for every transition t which is enabled at M we do the following:

- If $M(p) = M(q) = \omega$ and $F(t,p) - F(p,t) \neq F(t,q) - F(q,t)$, then the algorithm halts and outputs ω. Otherwise, a new successor n' of n with a temporary label M' (where $M \xrightarrow{t} M'$) is created.
- We check whether the following two conditions hold for every predecessor n'' of n'. If not, the algorithm halts and outputs ω.
  - If p is pumped at n' from n'' by k, then $M'(q) \neq \omega$ and q is either not pumpable at n', or it is pumped at n' from n'' by the same k.
  - If q is pumped at n' from n'' by k, then $M'(p) \neq \omega$ and p is either not pumpable at n', or it is pumped at n' from n'' by the same k.
- If the algorithm does not terminate in the previous point, we redefine $M'(r) = \omega$ for every place r pumpable at n'.

If the algorithm terminates by processing all nodes, it outputs the maximal finite value c for which there is a node n labelled by M in the constructed tree such that $M(p) \neq M(q)$, and $M(p) = c$ or $M(q) = c$. $\square$

Now let $\mathcal{N}$ be a BPP net with initial marking M_0, and let Q and Q' be important sets of places of $\mathcal{N}$. We say that Q and Q' are *c-dependent* for a given $c \in \mathbb{N}_\omega$ if for every reachable marking M we have that if $c < \textsc{norm}(Q)(M) < \omega$ and $c < \textsc{norm}(Q')(M) < \omega$, then $\textsc{norm}(Q)(M) = \textsc{norm}(Q')(M)$. The *dependence level* of Q, Q' is the least $c \in \mathbb{N}_\omega$ such that Q and Q' are c-dependent.

Lemma 16. *Let Q and Q' be important sets of places of a BPP net $\mathcal{N}$, and let M_0 be a marking of $\mathcal{N}$. The dependence level of Q, Q' is effectively computable.*

Proof. First we extend the net $\mathcal{N}$ by two fresh places p and q. Then we remove all transitions which put a token to $\textsc{maxtrap}(Q)$ or to $\textsc{maxtrap}(Q')$, and modify the other transitions so that for every reachable marking M we have that $\textsc{norm}(Q)(M) = M(p)$ and $\textsc{norm}(Q')(M) = M(q)$ (i.e., we "count" $\textsc{norm}(Q)$ in p and $\textsc{norm}(Q')$ in q). This is easy because $\textsc{norm}(Q)(M)$ and $\textsc{norm}(Q')(M)$ are just weighted sums of tokens in M (cf. the remarks after Definition 6). Note that the resulting Petri net is not necessarily a BPP net. Initially, p and q contain $\textsc{norm}(Q)(M_0)$ and $\textsc{norm}(Q')(M_0)$ tokens, respectively. Obviously, the dependence level of Q, Q' in $\mathcal{N}$ equals to the dependence level of p, q in the modified net, and thus it is effectively computable by Lemma 15. □

The usefulness of the above explorations now becomes apparent.

Lemma 17. *Let M_0 be a marking of a BPP net $\mathcal{N}$. If Q and Q' are important sets of places with dependence level ω, then M_0 is not bisimilar to any BPA process.*

For example, the sets $\{P\}$ and $\{Q\}$ are important for the BPP net of Figure 1; the associated DD-functions (referring to Lemma 14) are dd_a and dd_b, respectively. As these sets have a dependence level of ω, this Lemma demonstrates that there is no BPA process which is bisimilar to P.

Proof. Let $Q_1, \ldots, Q_m$ be the important sets of places associated to the BPP net $\mathcal{N}$, and let $d_1, \ldots, d_m$ be the associated DD-functions in the sense of Lemma 14. Now suppose that there are important sets Q and Q' whose dependence level for M_0 equals ω, and that there is a BPA process α_0 such that $M_0 \sim \alpha_0$. By Lemma 13, $d_1, \ldots, d_m$ are prefix-encoded on (any) BPA. Let

$$C = \max\{C_{d_i} : 1 \leq i \leq m\}$$

where C_{d_i} is the constant of Definition 11 chosen for d_i and the underlying BPA process of α_0. We also let k be the number of places of $\mathcal{N}$.

Since the dependence level of Q, Q' for M_0 equals ω, there is a reachable marking M such that

$$C + k < \textsc{norm}(Q)(M) < \omega \quad \text{and} \quad C + k < \textsc{norm}(Q')(M) < \omega,$$

and $\textsc{norm}(Q)(M) \neq \textsc{norm}(Q')(M)$. Let d and d' be the DD-functions associated to Q and Q', respectively. Since M is reachable, $M \sim \alpha$ for some α reachable from α_0. As DD-functions are bisimulation invariant, we get that

$$C + k < d(M) = d(\alpha) < \omega \quad \text{and} \quad C + k < d'(M) = d'(\alpha) < \omega\,.$$

Due to the choice of C, we know that for every sequence of (at most) k transitions, if each transition is d-reducing then each is also d'-reducing, and vice versa.

Certainly $Q \neq Q'$ as otherwise we could not have $\textsc{norm}(Q)(M) \neq \textsc{norm}(Q')(M)$. Hence, there is some $p \in (Q \setminus Q') \cup (Q' \setminus Q)$. Suppose, e.g., $p \in (Q \setminus Q')$. If $M(p) \geq 1$, we are done immediately, as then there is a d-reducing transition $M \xrightarrow{a} M'$, where $d = \text{NORM}(Q)$, which takes the token away from p, and therefore it does not decrease $\text{NORM}(Q') = d'$. The bisimilar BPA process α cannot match this transition.

If for every $p \in (Q \setminus Q') \cup (Q' \setminus Q)$ we have that $M(p) = 0$, we argue as follows: Let us assume that, e.g., $\text{NORM}(Q)(M) < \text{NORM}(Q')(M)$. Then there must be some $q \in Q \cap Q'$ such that $M(q) \geq 1$, and each sequence of d-reducing transitions, where $d = \text{NORM}(Q)$, which removes the token in q out of Q (that is, the total effect of the sequence on places in Q is that the token in q disappears) must temporarily mark some place p in Q' which is not in Q. Otherwise, we would immediately obtain that $\text{NORM}(Q')(M) \leq \text{NORM}(Q)(M)$. Now we can change the order of these transitions so that p is marked after at most $(k-1)$ d-reducing transitions. (Here we rely on a folklore result about BPP processes. Also note that any performable permutation of a sequence of d-reducing transitions also consists of d-reducing transitions). Now we can use the same argument as in the previous paragraph. □

Lemma 18. *Let M_0 be a marking of a BPP net $\mathcal{N}$. If for all important sets of places Q and Q' we have that the dependence level of Q, Q' is finite, then we can effectively construct an OCR process which is bisimilar to M_0.*

Proof. Let $Q_1, \ldots, Q_m$ be the important sets of places constructed for the BPP net $\mathcal{N}$, and let $\mathcal{F} = (d_1, \ldots, d_m)$ be the (tuple of the) associated DD-functions as in Lemma 14. Furthermore, let

$$C = \max\{dl(Q_i, Q_j) : 1 \leq i, j \leq m\}$$

where $dl(Q_i, Q_j)$ is the dependence level of Q_i, Q_j. Note that C is effectively computable due to Lemma 16. For every reachable marking M, all norms $\text{NORM}(Q_i)(M)$ which are finite and larger than C keep to coincide. More precisely, if

$$C < \text{NORM}(Q_i)(M) < \omega \quad \text{and} \quad C < \text{NORM}(Q_j)(M) < \omega$$

where $1 \leq i, j \leq m$, then $\text{NORM}(Q_i)(M) = \text{NORM}(Q_j)(M)$.

So we can construct an OCR $\mathcal{P}$, with the initial state bisimilar to M_0, which mimics the behaviour of markings of $\mathcal{N}$ in the following way: Instead of (the current marking) M, the OCR $\mathcal{P}$ records in the finite control state unit:

– for which sets Q_i the value $\text{NORM}(Q_i)(M)$ equals ω (now and forever);
– for which sets Q_i the value $\text{NORM}(Q_i)(M)$ is “small”, i.e., no greater than C; the precise values of these norms are also recorded in the finite control; and
– for which sets Q_i the value $\text{NORM}(Q_i)(M)$ is larger than C.

Since the values $\text{NORM}(Q_i)(M)$ in the last collection must be the same, i.e., equal to some v, $\mathcal{P}$ can record $(v-C)$ in the counter.

Note that the configuration of $\mathcal{P}$ associated to M does not contain the “full” information about M in the sense that the exact distribution of tokens in M cannot be reconstructed from the recorded norms. It can happen that different markings M, M' have the property $\text{NORM}(Q_i)(M) = \text{NORM}(Q_j)(M')$ for every $1 \leq i \leq m$, and then (and only then) they are represented by the same configuration of $\mathcal{P}$. However, if M

and M' coincide on every $\textsc{norm}(Q_i)$, then (and only then) they are bisimilar—see Lemma 14.

It remains to explain how $\mathcal{P}$ performs its transitions. First, for every marking M we define the set

$$\mathit{fire}(M) = \{(a, \delta) : \exists M \xrightarrow{a} M' \text{ such that } \delta^{\mathcal{F}}(M \xrightarrow{a} M') = \delta\}.$$

Note that if $M \sim M'$, then $\mathit{fire}(M) = \mathit{fire}(M')$. Hence, for every configuration $p\alpha$ of $\mathcal{P}$ we can define $\mathit{fire}(p\alpha) = \mathit{fire}(M)$ where M is a marking to which $p\alpha$ is associated. (If there is no such M, $\mathit{fire}(p\alpha)$ is undefined.) Next we show that for every $p\alpha$ for which $\mathit{fire}(p\alpha)$ is defined, the set $\mathit{fire}(p\alpha)$ is effectively computable just from the information stored in the control state p (hence, we can denote $\mathit{fire}(p\alpha)$ just by $\mathit{fire}(p)$). It clearly suffices for our purposes, because then we can compute the sets $\mathit{fire}(q)$ for all control states q and "implement" each pair (a, δ) in a given $\mathit{fire}(q)$ in the straightforward way; in particular, if all "large" norms are set to ω, then the control state is updated and the counter is reset to zero. One can also readily verify that every marking M is bisimilar to its associated configuration of $\mathcal{P}$.

Let t be a transition of $\mathcal{N}$. In each marking M where t is enabled, the (firing of) t causes some change of $\mathcal{F}$ (i.e., of $(\textsc{norm}(Q_1), \cdots, \textsc{norm}(Q_m))$); we define

$$\delta_t(M) = \delta^{\mathcal{F}}(M \xrightarrow{t} M').$$

In general, the same transition can cause different changes in different markings $M, \bar{M}$. The (only) reason why this can happen is that some $\textsc{norm}(Q_i)$ which is finite for M can be ω for $\bar{M}$, and vice versa (note that if $\textsc{norm}(Q_i)(M) = \omega$, then $\delta_t(M)_i = \omega$). So, $\delta_t(M)$ is (completely and effectively) determined by the structure of $\mathcal{N}$ and the *mode* of M, i.e., the set of all Q_i's for which $\textsc{norm}(Q_i)(M) < \omega$. Hence, for a given a mode $\mathcal{M}$ (i.e., a subset of $\{Q_1, \cdots, Q_m\}$ whose norms are to be finite) we can effectively partition the transitions of $\mathcal{N}$ into classes $T_1, \cdots, T_k$ so that transitions in the same T_i have the same label and the same change at every marking with mode $\mathcal{M}$. Thus, to each T_i we can associate a pair (a_i, δ_i). Note that

$$\mathit{fire}(M) \subseteq \big\{(a_1, \delta_1), \cdots, (a_k, \delta_k)\big\}$$

for every marking M with mode $\mathcal{M}$. Now we show that for each T_i one can effectively construct an important set Q_{j_i} such that for every marking M with mode $\mathcal{M}$ we have that

$$(a_i, \delta_i) \in \mathit{fire}(M) \quad \text{iff} \quad \textsc{norm}(Q_{j_i})(M) > 0. \tag{2}$$

Actually, it suffices to put

$$Q_{j_i} = \bigcup_{t \in T_i} \mathit{Pre}(t) \cup \bigcup_{Q \in \mathcal{M}} \textsc{maxtrap}(Q).$$

Clearly, $\textsc{norm}(Q_{j_i})$ is a bisimulation-invariant norm; and it follows from the construction of the important sets $Q_1, \cdots, Q_m$ presented in [11] that Q_{j_i} appears in the sequence $Q_1, \cdots, Q_m$. It remains to verify that Equation (2) indeed holds. So, let M be a marking with mode $\mathcal{M}$. If $(a_i, \delta_i) \in \mathit{fire}(M)$, some of the transitions in T_i are enabled at M, and hence $\textsc{norm}(Q_{j_i})(M) > 0$. Conversely, if $\textsc{norm}(Q_{j_i})(M) > 0$, the places of $\bigcup_{Q \in \mathcal{M}} \textsc{maxtrap}(Q)$ are surely not marked (otherwise, we would have that $\textsc{norm}(Q)(M) = \omega$ for some Q in $\mathcal{M}$ which is a contradiction). Hence, some $Pre(t)$, where $t \in T_i$, must be marked at M which means that t is enabled at M.

Since control states of $\mathcal{P}$ carry the information about the current mode and currently positive $\textsc{norm}(Q_i)$'s, we are done. □

The previous lemmata allow us to conclude the following:

Theorem 19. *Bisimilarity between BPA and BPP processes is decidable.*

Proof. We first check if there are important sets Q and Q' whose dependence level equals ω (this is decidable by Lemmas 14 and 16). If this is the case, then M_0 cannot be bisimilar to any BPA process. Otherwise, we can effectively construct an OCR process $p\alpha$ which is bisimilar to M_0 (Lemma 18). We can then check bisimilarity between $p\alpha$ and the given BPA process (using, e.g., the algorithm of [17, 22]). □

As a final remark, we note that a more detailed analysis of the properties of BPP processes which are bisimilar to BPA processes would admit a refinement of the result of Lemma 18: the OCR must be special in the sense that the first transition which increments the counter (since the last reset) "selects" the control state q to which the machine must switch by the first decrement operation. The only possibility of how to change q to some other state is to perform a reset. We conjecture that such a behaviour can be matched by an effectively constructible BPA process. This being the case, we could then use the bisimilarity-checking algorithm for BPA processes [4] instead of the one for PDA processes, which would yield an elementary upper complexity bound for bisimilarity between BPA and BPP.

References

[1] J.C.M. Baeten, J.A. Bergstra and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the ACM* **40**(3):653–682, 1993.

[2] J. Blanco. Normed BPP and BPA. In *Proceedings of ACP'94*, pages 242–251, Springer 1995.

[3] O. Burkart, D. Caucal, F. Moller and B. Steffen. Verification on infinite structures. In J. Bergstra, A. Ponse and S. Smolka (editors), *Handbook of Process Algebra*, chapter 9, pages 545–623. Elsevier Science, 2001.

[4] O. Burkart, D. Caucal and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of MFCS'95, LNCS* 969, pages 423–433. Springer, 1995.

[5] I. Černá, M. Křetínský and A. Kučera. Comparing expressibility of normed BPA and normed BPP processes. *Acta Informatica* **36**:233–256, 1999.

[6] R. Glabbeek. The linear time – branching time spectrum I: The semantics of concrete sequential processes. In J. Bergstra, A. Ponse and S. Smolka (editors), *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier Science, 2001.

[7] Y. Hirshfeld and M. Jerrum. Bisimulation equivalence is decidable for normed process algebra. In *Proceedings of ICALP'99, LNCS* 1644, pages 412–421. Springer, 1999.

[8] Y. Hirshfeld, M. Jerrum and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science* **158**:143–159, 1996.

[9] Y. Hirshfeld, M. Jerrum and F. Moller. A polynomial-time algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Journal of Mathematical Structures in Computer Science* **6**:251–259, 1996.

[10] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science* **148**(2):281–301, 1995.

[11] P. Jančar. Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proceedings of LICS'03*, to appear, 2002.

[12] P. Jančar and A. Kučera. Equivalence-checking with infinite-state systems: Techniques and results. In *Proceedings of SOFSEM'02*, *LNCS* 2540, pages 41–73. Springer, 2002.

[13] P. Jančar and F. Moller. Techniques for decidability and undecidability of bisimilarity. In *Proceedings of CONCUR'99*, *LNCS* 1664, pages 30–45. Springer, 1999.

[14] R.M. Karp and R.E. Miller. Parallel Program Schemata *Journal of Computer and Systems Sciences* **3**:147–195, 1969.

[15] R. Milner and F. Moller. Unique decomposition of processes. *Theoretical Computer Science* **107**:357–363, 1993.

[16] F. Moller. Infinite results. In *Proceedings of CONCUR'96*, *LNCS* 1119, pages 195–216. Springer, 1996.

[17] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proceedings of FOCS'98*, pages 120–129. IEEE Computer Society Press, 2001.

[18] G. Sénizergues. L(A)=L(B)? decidability results from complete formal systems. *Theoretical Computer Science* **251**(1-2):1–166, 2001.

[19] J. Srba. Roadmap of Infinite Results. http://www.brics.dk/˜srba/roadmap.

[20] J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proceedings of ICALP'02*, *LNCS* 2380, pages 716–727. Springer, 2002.

[21] J. Srba. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard. In *Proceedings of STACS'02*, *LNCS* 2285, pages 535–546. Springer, 2002.

[22] C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Research Report No. EDI-INF-RR-0005, School of Informatics, Edinburgh University. January 2000.

[23] C. Stirling. Decidability of DPDA equivelance. *Theoretical Computer Science* **255**(1-2):1–31, 2001.