A Polynomial-Time Algorithm for Deciding True Concurrency Equivalences of Basic Parallel Processes

Sławomir Lasota*

Institute of Informatics, Warsaw University, Poland,
sl@mimuw.edu.pl

Abstract. A polynomial-time algorithm is presented to decide distributed bisimilarity of Basic Parallel Processes. As a direct conclusion, several other non-interleaving semantic equivalences are also decidable in polynomial time for this class of process, since they coincide with distributed bisimilarity.

1 Introduction

One important problem in the verification of concurrent systems is to check whether two given systems P and Q are equivalent under a chosen notion of equivalence. For process algebras generating infinite-state systems the equivalence checking problem cannot be decidable in general, therefore restricted classes of processes have been defined and investigated. We study here the class of Basic Parallel Processes [9] (BPP), an extension of recursively defined finite-state systems by parallel composition. Strong bisimilarity [25] is a well accepted behavioural equivalence, which often remains decidable for infinite-state systems. An elegant proof of decidability of bisimilarity for BPP and even for BPP_τ, extension of BPP by communication, was given in [10]. The PSPACE lower bound has been recently proved in [26], followed by the PSPACE-completeness result of Jančar [18]. On the other hand, all other equivalences in van Glabbeek's spectrum are undecidable [17].

BPP is the natural class of processes to investigate non-interleaving equivalences, intended to capture true concurrent computations of a system. One of the bisimulation-like non-interleaving equivalences is distributed bisimilarity [6], taking into account spatial distribution of a process. Already in [8] distributed bisimilarity was shown to be decidable on BPP_τ by means of a sound and complete tableau proof system. Concerning complexity, the tableau depth was only bounded exponentially.

In this paper we design a polynomial-time decision procedure for distributed bisimilarity. It strongly uses a polynomial-time algorithm for deciding strong bisimilarity on *normed* BPP processes proposed in [16]. Distributed bisimilarity is therefore very likely to be computationally more feasible than interleaving bisimilarity, in the light of the recent PSPACE lower bound for the latter one by Srba [26].

* A part of this work has been performed during the post-doc stay at Laboratoire Specification et Verification, ENS Cachan. Partially supported by the KBN grant 7 T11C 002 21 and the EC Research Training Network "Games and Automata for Synthesis and Validation" (GAMES).

Further interesting conclusions follow from the fact that many non-interleaving equivalences coincide on BPP. As mentioned in [12], Kiehn proved [21] that *location equivalence* [7], *causal equivalence* [11] and distributed bisimilarity all coincide on CPP, a sublanguage of BPP_τ without explicit τ, hence also on BPP. Furthermore, causal equivalence and *history preserving bisimilarity* [13] coincide on BPP by the result of Aceto [1]; moreover, Fröschle showed coincidence of distributed and history preserving bisimilarity on BPP [12]. The coincidence with *performance equivalence*, a timed bisimulation equivalence proposed by [14], has been shown in [23], to complete the picture. [1] As a direct conclusion from all these results, all the mentioned equivalences can be also decided in polynomial time on BPP. Related results are [15] decision procedures for causal equivalence, location equivalence and ST-bisimulation equivalence of BPP_τ as well as for their weak versions on a subset of BPP_τ. However, complexity issues were not addressed there. Furthermore, polynomial-time complexity of performance equivalence extends the result of [5] shown only for timed BPP in a full standard form [9]. Surprisingly, polynomial-time complexity of history preserving bisimilarity on BPP can be contrasted with the EXPTIME-completeness on finite-state systems (finite 1-safe nets) [19]. Similarly, decidability of hereditary history preserving bisimilarity [4] on BPP, proved in [12], can be contrasted with undecidability on finite 1-safe nets shown by Jurdziński and Nielsen in [20].

We start by Section 2 containing definitions and some basic facts and then we outline our algorithm in Section 3. The algorithm works for BPP processes in *standard form*, similarly as in [9]. A polynomial-time preprocessing procedure transforming a process into standard form can be found in the full version of the paper [24].

2 Basic Definitions and Facts

Let Act be a finite set of *actions*, ranged over by a, b, etc. and let $Const$ be a finite set of *process constants*, ranged over by X, Y, etc. The set of BPP process expressions [9] over Act and $Const$ is given by:

$$P \ ::= \ \mathbf{0} \ | \ X \ | \ a.P \ | \ \sum\nolimits_{i \in I} P_i \ | \ P \| P \ | \ P \lfloor\!\lfloor P \tag{1}$$

where $\mathbf{0}$ denotes the *empty* process, $a.\_$ is an action prefix, $\sum_{i \in I} P_i$ is a finite nondeterministic choice for a finite nonempty set I and $\_\|\_$ stands for a parallel composition. The only operator not present in CCS [25] is the left merge $\_\lfloor\!\lfloor\_$, which differs from the parallel composition only in that the very first action must be performed in the left argument. The purpose of considering $\lfloor\!\lfloor$ here is the standard form (3) below.

A BPP *process definition* Δ consists of a finite set $Act(\Delta)$ of actions, a finite set $Const(\Delta)$ of constants and a finite number of recursive process equations

$$X \stackrel{\text{def}}{=} P,$$

[1] Related results are [2], where Aceto proved that distributed bisimilarity coincides with *timed bisimilarity* on BPP without recursion, and [22], where decidability of strong bisimilarity for timed BPP was shown, that generalizes the result of [10].

one for each constant $X \in Const(\Delta)$, where P is a process expression over $Act(\Delta)$ and $Const(\Delta)$. Sets $Const(\Delta)$ and $Act(\Delta)$ are often even not mentioned explicitly, as they can be deduced from process equations. In the sequel we shall assume that a process definitions that our algorithm inputs is *guarded*, i.e., that each occurrence of a constant on the right-hand side is within the scope of an action prefix. For instance, $X \stackrel{\text{def}}{=} a.Y \,\|\!\_\, X + b.Z$ is *not* guarded. By a BPP *process* we mean a pair (P, Δ), where Δ is a BPP process definition and P is a BPP process expression over $Act(\Delta)$ and $Const(\Delta)$. When Δ is evident from the context, P itself is called a *process* too.

Distributed bisimilarity was introduced in [6], but here we follow [9]. Given a BPP process definition Δ, consider the following SOS transition rules:

$$
\begin{array}{cc}
\dfrac{(X \stackrel{\text{def}}{=} P) \in \Delta \qquad P \stackrel{a}{\to} [P', P'']}{X \stackrel{a}{\to} [P', P'']} & \dfrac{}{a.P \stackrel{a}{\to} [P, \mathbf{0}]} \\[2ex]
\dfrac{P_j \stackrel{a}{\to} [P', P''] \quad \text{for some } \ j \in I}{\sum_{i \in I} P_i \stackrel{a}{\to} [P', P'']} & \dfrac{P \stackrel{a}{\to} [P', P'']}{P \,\|\!\_\, Q \stackrel{a}{\to} [P', P'' \| Q]} \\[2ex]
\dfrac{P \stackrel{a}{\to} [P', P'']}{P \| Q \stackrel{a}{\to} [P', P'' \| Q]} & \dfrac{Q \stackrel{a}{\to} [Q', Q'']}{P \| Q \stackrel{a}{\to} [Q', P \| Q'']}
\end{array}
\tag{2}
$$

We write $P \stackrel{a}{\to} [P', P'']$ if this transition can be derived from the above rules. The rules reflect a view on a process as distributed in space. Each transition $P \stackrel{a}{\to} [P', P'']$ gives rise to a *local* derivative P', which intuitively records a *location* at which the action is observed, and a *concurrent* derivative P'', recording the part of the process separated from the local component.

BPP processes (P_1, Δ_1) and (P_2, Δ_2) are *distributed bisimilar*, denoted $(P_1, \Delta_1) \sim (P_2, \Delta_2)$ if they are related by some *distributed bisimulation* R, i.e., a binary relation over BPP process expressions such that whenever $(P, Q) \in R$, for each $a \in Act$,

- if $P \stackrel{a}{\to} [P', P'']$ then $Q \stackrel{a}{\to} [Q', Q'']$ for some Q', Q'' such that $(P', Q') \in R$ and $(P'', Q'') \in R$,
- if $Q \stackrel{a}{\to} [Q', Q'']$ then $P \stackrel{a}{\to} [P', P'']$ for some P', P'' such that $(P', Q') \in R$ and $(P'', Q'') \in R$.

In the next section we prove polynomial-time complexity of the problem of checking distributed bisimilarity for a given pair of constants. We do not lose generality, as checking $P \sim Q$ for arbitrary P, Q is equivalent to checking $X_P \sim X_Q$, where X_P and X_Q are new fresh constants with defining equations: $X_P \stackrel{\text{def}}{=} a.P$ and $X_Q \stackrel{\text{def}}{=} a.Q$, for arbitrary a. Moreover, w.l.o.g. we assume that both constants share a process definition.

Problem:	DISTRIBUTED BISIMILARITY FOR BPP
Instance:	A BPP process definition Δ and $X, Y \in Const(\Delta)$
Question:	$(X, \Delta) \sim (Y, \Delta)$?

Christensen presented in [8] a sound and complete tableau proof system for $\sim$ on BPP_τ and proved an exponential upper bound for the depth of a tableau.

Theorem 1 ([8, 9]). *Distributed bisimilarity is decidable on* BPP.

Christensen [9] showed also that each BPP process definition Δ can be effectively transformed into an equivalent process definition Δ' in *standard form*, i.e., consisting exclusively of process equations in the restricted form

$$X \stackrel{\text{def}}{=} \sum_{i \in I} (a_i.P_i) \,⌊⌊\, Q_i, \tag{3}$$

where all P_i and Q_i are merely a parallel composition of constants, i.e., of the form

$$X_1 \| X_2 \| \ldots \| X_n, \quad \text{for } n > 0 \text{ and } X_1, \ldots, X_n \in Const(\Delta'). \tag{4}$$

Note that (3) is not guarded in general. We omit brackets in (4) as _‖_ is associative and commutative w.r.t. $\sim$ (and w.r.t. any other known semantical equivalence, in fact). A parallel composition of constants (4) is called *basic process (expression)* in the sequel. Observe that processes P_i in (3) are precisely local derivatives of X and processes Q_i are precisely its concurrent derivatives. Hence the left merge operator allows here to syntactically separate both derivatives. Consequently, in the next section we will only consider basic processes since both local and concurrent derivatives of a basic process are basic again. Since Δ is guarded, the process definition Δ' produced by Christensen's transformation is *bounded* in the following sense:

Definition 1. *Define a binary relation over $Const(\Delta')$ as follows: $Y \prec_1 X$ iff Y appears in some concurrent derivative of X. We say that Δ' is* bounded *if the transitive closure $\prec_1^+$ of $\prec_1$ is irreflexive, i.e., no constant satisfies $X \prec_1^+ X$.*

Christensen's transformation produces Δ' that contains only basic processes consisting of *at most* three constants. The price for this constant bound is the exponential size of Δ' w.r.t. the size of Δ, defined as the number of process equations in Δ plus the sum of lengths of right-hand sides. This is why we proved (Theorem 2 below) that the transformation to the standard form (3) can be done in polynomial time.

Theorem 2. *There exists a polynomial-time algorithm that transforms a guarded process definition Δ into a process definition Δ' such that:*

1. *Δ' is in standard form (3),*
2. *Δ' is bounded,*
3. *$Const(\Delta) \subseteq Const(\Delta')$ and for each $X \in Const(\Delta)$, $(X, \Delta) \sim (X, \Delta')$.*

(The proof is given in the full version of this paper [24].)

Strong bisimilarity is defined w.r.t. single-derivative transitions $P \xrightarrow{a} P'$ obtained by the rules (2) when the distribution of a process is ignored – for details we refer eg. to [25]. A remarkable result is that strong bisimilarity can be decided for BPP processes in polynomial time [16], but only when all the constants are *normed*. A constant, or generally a process P, is *normed* if an inactive process is reachable from P, i.e., if there is a finite sequence of transitions $P \xrightarrow{a_1} P_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} P_n$, $n \geq 0$, such that there are no further transition from P_n. The *norm* of P is the length of the shortest such sequence.

Strong bisimilarity is less restrictive than distributed bisimilarity. Hence a process definition Δ can be transformed into a strong bisimilarity equivalent process definition Δ' in a more restrictive than (3) *full standard form* [9]. Full standard form admits exclusively process equations in the form: $X \stackrel{\text{def}}{=} \sum_{i \in I} a_i.P_i$, where all P_i are basic again.

Theorem 3 ([16]). *There exists a polynomial-time algorithm to decide strong bisimilarity on normed* BPP *in full standard form.*

3 Algorithm

Throughout this section we fix Δ, and hence also sets of constants and actions. Δ is assumed to be in standard form and bounded.

A more succinct representation is possible for Δ in standard form. Namely, due to associativity and commutativity of $\_\|\_$, it is sufficient to remember the number of occurrences of each constant in each basic expression P_i and Q_i in the right-hand side of all equations (3) in Δ, encoded in binary. In this section, complexity is measured w.r.t. the size of Δ defined as the number of process equations plus the sum of lengths of the succinct representations of the right-hand sides. Theorem 3 is still valid for this definition of size.

The reachability relation over process expressions is defined as the smallest transitive relation such that each P is related to all its local and concurrent derivatives, i.e, whenever $P \xrightarrow{a} [P', P'']$, the relation contains pairs (P, P') and (P, P''). We say that Q is *reachable* from P if the pair (P, Q) is in the reachability relation.

Let us denote by $\mathbb{P}$ the set of all process expressions reachable from the constants. As observed previously, all $P \in \mathbb{P}$ are basic. Unless stated otherwise, all the relations mentioned below are binary relations over $\mathbb{P}$. This includes also $\sim$, which is restricted now to $\mathbb{P}$ only. Unless explicitly stated otherwise, P, Q, etc., range over $\mathbb{P}$.

An exponential-time algorithm can be easily derived as follows. First define two monotonic operators as follows. The operator $\mathcal{B}_2$ acts on a pair of relations as follows: $(P, Q) \in \mathcal{B}_2(R', R'')$ iff for each $a \in Act$,

- if $P \xrightarrow{a} [P', P'']$ then $Q \xrightarrow{a} [Q', Q'']$ for some Q', Q'' such that $(P', Q') \in R'$ and $(P'', Q'') \in R''$,
- if $Q \xrightarrow{a} [Q', Q'']$ then $P \xrightarrow{a} [P', P'']$ for some P', P'' such that $(P', Q') \in R'$ and $(P'', Q'') \in R''$.

Then the operator $\mathcal{B}_1$ is defined by $\mathcal{B}_1(R) := \mathcal{B}_2(R, R)$. Now, define the approximating equivalences $\sim_i$ as follows:

- $P \sim_0 Q$ for all P and Q,
- $\sim_{i+1} \; := \; \mathcal{B}_1(\sim_i)$, for $i \geq 0$.

Distributed bisimulations R are exactly the post-fixed points of $\mathcal{B}_1$, i.e., are defined by $R \subseteq \mathcal{B}_1(R)$. Hence $\sim$ being the union of all distributed bisimulations is then the greatest fixed point of $\mathcal{B}_1$, by the Knaster-Tarski theorem. Recall that BPP is image-finite, that is to say each process expression has only finitely many local and concurrent derivatives. Thus by a standard argument the decreasing chain $\{\sim_i\}$ converges to $\sim$:

$$\sim \; = \bigcap_{i \in \mathbb{N}} \sim_i. \tag{5}$$

Furthermore, observe that each local derivative of a basic process $X_1\|\ldots\|X_n$ is a local derivative of some X_j, i.e., is equal to some process P_i appearing in a process equation (3) in Δ. In consequence, the number of local derivatives of all basic processes is polynomial. Let us denote the set of all those by $\mathbb{L}$. Moreover, there are only exponentially many processes reachable from each $P \in \mathbb{L}$ – this follows easily from boundedness of Δ. Consequently, the cardinality N of the whole $\mathbb{P}$ is exponential. Hence $\sim$ can be computed over $\mathbb{P}$ in exponential time, e.g., as the limit of the sequence $\{\sim_i\}$ of equivalences, since the sequence stabilizes after at most $N-1$ steps.

We have not focused on details of the exponential-time algorithm, as in the rest of this section we argue that one can do better: the problem can be solved in polynomial time. Essentially, this is possible due to a "quicker" convergence to the greatest fixed point, as explained in Lemma 2 below and thereafter. Then, in the crucial Lemma 4 we reduce distributed bisimilarity to strong bisimilarity between normed processes. To this aim we incorporate local derivatives into actions and obtain single-derivative transitions. We start by a couple of definitions and simple facts.

Definition 2. *Given a binary relation S, a* distributed bisimulation w.r.t. S *is any binary relation R such that $R \subseteq \mathcal{B}_2(S, R)$. P and Q are* distributed bisimilar w.r.t. S, *denoted $P {\sim^S} Q$, if they are related by some distributed bisimulation w.r.t. S.*

Definition 3. *We say that a relation R is a* distributed bisimilarity w.r.t. itself *if $R = {\sim^R}$. Let $\approx$ denote the greatest distributed bisimilarity w.r.t. itself.*

A relation is a distributed bisimilarity w.r.t. itself precisely if it is a fixed point of the monotonic mapping $R \mapsto {\sim^R}$. Hence the greatest distributed bisimilarity w.r.t. itself always exists.

Lemma 1. *$\sim$ and $\approx$ coincide.*

Proof. For one inclusion, recall that $\sim$ is the union of all distributed bisimulations while ${\sim^\sim}$ is the union of all distributed bisimulations w.r.t. $\sim$. Since each distributed bisimulation is a distributed bisimulation w.r.t. $\sim$, we have ${\sim} \subseteq {\sim^\sim}$, i.e., $\sim$ is a post-fixed point of the mapping $R \mapsto {\sim^R}$. As $\approx$ is the greatest fixed point of that mapping, we obtain ${\sim} \subseteq {\approx}$.

For the other inclusion, assume a relation S is a distributed bisimilarity w.r.t. itself, $S = {\sim^S}$. Since ${\sim^S}$ is the union of all distributed bisimulations w.r.t. S, it is the (greatest) fixed point of the monotonic mapping $R \mapsto \mathcal{B}_2(S, R)$, i.e., ${\sim^S} = \mathcal{B}_2(S, {\sim^S})$. Substituting S in place of ${\sim^S}$ we get $S = \mathcal{B}_2(S, S)$, i.e., S is a fixed point of $\mathcal{B}_1$. Hence $S \subseteq {\sim}$. As S was chosen arbitrary, we showed that each distributed bisimilarity w.r.t. itself is included in $\sim$. Hence also $\approx$ does, i.e., ${\approx} \subseteq {\sim}$. □

We proved that $\approx$ is just another formulation of $\sim$. But $\approx$ gives rise to another sequence of approximating equivalences $\{\approx_i\}$ that converges more rapidly than $\{\sim_i\}$, namely after a polynomial number of iterations:

Lemma 2. *${\approx} = \bigcap_{i\in\mathbb{N}} {\approx_i}$, where the sequence $\{\approx_i\}$ is defined by:*

- *$P \approx_0 Q$ for all P and Q,*
- *${\approx_{i+1}} := {\sim^{\approx_i}}$.*

Proof. Obviously $\approx\ \subseteq \bigcap_{i\in\mathbb{N}} \approx_i$, so we only need to show the opposite inclusion. Similarly as $\sim$, $\sim^S$ is the greatest fixed point of the monotonic mapping $R \mapsto \mathcal{B}_2(S, R)$, for any fixed S. So $\sim^S$ is also a limit of a decreasing sequence of approximations:

$$\sim^S = \bigcap_{i\in\mathbb{N}} \sim_i^S, \tag{6}$$

where relations $\sim_i^S$ are defined by:

- $P \sim_0^S Q$ for all P and Q,
- $\sim_{i+1}^S \ := \ \mathcal{B}_2(S, \sim_i^S)$.

Having this, by an easy induction we show $\approx_i\ \subseteq\ \sim_i$, for all $i \geq 0$. As the induction assumption suppose $\approx_i\ \subseteq\ \sim_i$, for a fixed $i \geq 0$. Now substitute $\approx_i$ in place of S in (6) and in the definition of $\sim_j^S$, $j \geq 0$. Due to monotonicity of $\mathcal{B}_2$ we derive by another easy induction on $j \leq i$ that $\sim_{j+1}^{\approx_i} \ = \ \mathcal{B}_2(\approx_i, \sim_j^{\approx_i}) \ \subseteq \ \mathcal{B}_2(\sim_j, \sim_j) \ = \ \sim_{j+1}$, since $\approx_i\ \subseteq\ \sim_i\ \subseteq\ \sim_j$ by the induction assumption and by $j \leq i$. Hence $\sim_{i+1}^{\approx_i}\ \subseteq\ \sim_{i+1}$. By (6) we know that $\approx_{i+1}\ =\ \sim^{\approx_i}\ \subseteq\ \sim_{i+1}^{\approx_i}$, hence we conclude $\approx_{i+1}\ \subseteq\ \sim_{i+1}$. This completes the induction step.

Having $\approx_i\ \subseteq\ \sim_i$, for all $i \geq 0$, we apply Lemma 1 and (5). $\square$

Equipped with Lemmas 1 and 2, we are ready to describe the polynomial-time algorithm. Recall that $\mathbb{L}$ denotes the set of all local derivatives. The algorithm consists of two phases as outlined in the figure below. By $R \cap (\mathbb{L} \times \mathbb{L})$ we mean here the restriction of a relation R to pairs from $\mathbb{L}$.

PHASE 1:

let $\bumpeq_0 := \mathbb{L} \times \mathbb{L}$
REPEAT FOR $n = 0, 1, \ldots$
  compute $\bumpeq_{n+1}\ \subseteq \mathbb{L} \times \mathbb{L}$ as follows: $\bumpeq_{n+1}\ :=\ \sim^{\bumpeq_n} \cap (\mathbb{L} \times \mathbb{L})$
UNTIL $\bumpeq_n\ =\ \bumpeq_{n+1}$

PHASE 2:

decide whether $X \sim^{\bumpeq_n} Y$

The first phase is a crucial one, but amounts simply to computing an initial part of $\{\approx_i\}$ up to the position n where it eventually stabilizes. The trick is that $\approx_i$ is computed only for local derivatives. Then, in the second phase we only need to check whether the input pair (X, Y) belongs to $\sim^{\approx_n}$. Assuming that the first phase of the algorithm terminates, the outcome $\sim^{\bumpeq_n}$ coincides with $\approx$.

Lemma 3. *If* $\bumpeq_n\ =\ \bumpeq_{n+1}$ *then* $\sim^{\bumpeq_n}\ =\ \approx$.

Proof. Assuming $\bumpeq_n\ =\ \bumpeq_{n+1}$, we will show $\sim^{\bumpeq_n}\ \subseteq\ \approx$ and $\sim^{\bumpeq_n}\ \supseteq\ \approx$. For both inclusions we will silently use an obvious fact: when two relations S_1 and S_2 coincide on $\mathbb{L} \times \mathbb{L}$, i.e., $S_1 \cap (\mathbb{L} \times \mathbb{L}) = S_2 \cap (\mathbb{L} \times \mathbb{L})$, then $\sim^{S_1}\ =\ \sim^{S_2}$.

For $\sim^{\bumpeq_n}\ \subseteq\ \approx$, it is sufficient to show that $\sim^{\bumpeq_n}$ is a distributed bisimilarity w.r.t. itself. Indeed, $\sim^{\sim^{\bumpeq_n}}\ =\ \sim^{\sim^{\bumpeq_n} \cap (\mathbb{L} \times \mathbb{L})}\ =\ \sim^{\bumpeq_{n+1}}\ =\ \sim^{\bumpeq_n}$.

For $\sim^{\bumpeq_n}\ \supseteq\ \approx$, we show by induction that for all $i \geq 0$,

(a) $\doteq_i = \approx_i \cap (\mathbb{L} \times \mathbb{L})$,
(b) $\sim^{\doteq_i} = \approx_{i+1}$.

For $i = 0$ it is obvious, so assume $i > 0$ and (a) and (b) hold for $i-1$. We prove (a) first: $\doteq_i \;=\; \sim^{\doteq_{i-1}} \cap (\mathbb{L} \times \mathbb{L}) \;\overset{(b)}{=}\; \approx_i \cap (\mathbb{L} \times \mathbb{L})$. Then (b) follows easily from (a): $\sim^{\doteq_i} \;\overset{(a)}{=}\; \sim^{\approx_i \cap (\mathbb{L}\times\mathbb{L})} \;=\; \sim^{\approx_i} \;=\; \approx_{i+1}$.

Now $\sim^{\doteq_n} \supseteq \approx$ follows from (b) since $\approx_{n+1} \supseteq \approx$, by Lemma 2. ☐

Termination of the first phase of the algorithm after a polynomial number of iterations of the main loop is guaranteed as the sequence $\{\doteq_i\}$ is non-increasing: $\doteq_0 \supseteq \doteq_1 \supseteq \ldots$, and each $\doteq_i$ contains only polynomially many pairs. What we still need to show is that the single iteration of the loop body, i.e., computation of $\doteq_{i+1}$ from $\doteq_i$, can be done in polynomial time. To this aim we will prove the following Lemma 4. In the proof we will profit from Theorem 3.

Lemma 4. *Let $S \subseteq \mathbb{L} \times \mathbb{L}$ be an equivalence such that there exists a polynomial-time algorithm (w.r.t. the size of Δ) to decide whether $(P, Q) \in S$, for given $P, Q \in \mathbb{L}$. Then there exists a polynomial-time algorithm to decide $P \sim^S Q$, for given P, $Q \in \mathbb{P}$.*

Proof. As the first stage, we construct from Δ a new process definition Δ' in the full standard form, equivalent to Δ in the following sense: $P, Q \in \mathbb{P}$,

$$(P, \Delta) \sim^S (Q, \Delta) \quad \text{iff} \quad (P, \Delta') \text{ is strongly bisimilar to } (Q, \Delta'). \tag{7}$$

The construction of Δ' is as follows:

$$Const(\Delta') := Const(\Delta) \qquad\qquad Act(\Delta') := Act(\Delta) \times \mathbb{L}/S,$$

where $\mathbb{L}/S$ denotes the set of equivalence classes of S and can be computed in polynomial time. Furthermore, whenever Δ contains a process equation

$$X \overset{\text{def}}{=} \sum\nolimits_{i \in I} (a_i.P_i) \lfloor\!\lfloor Q_i, \tag{8}$$

Δ' contains

$$X \overset{\text{def}}{=} \sum\nolimits_{i \in I} (a_i, [P_i]_S).Q_i, \tag{9}$$

where $[P_i]_S$ denotes the equivalence class of P_i in S. Having this, (7) is clear, by the very definitions of the two bisimilarities involved. Now, the crucial point is that Δ' is always normed, since Δ is bounded. We have $Y \prec_1 X$ (cf. Section 2) iff Y appears in some Q_i on the right-hand side of process equation (8) defining X. As the transitive closure $\prec_1^+$ is irreflexive, the following equations (10) and (11) are well-defined and give the norm of all the constants. First, for each constant X defined by (9) in Δ',

$$\text{norm}(X) = 1 + \min_{i \in I} \{\text{norm}(Q_i)\}. \tag{10}$$

Second, the norm is additive w.r.t. parallel composition:

$$\text{norm}(P \| Q) = \text{norm}(P) + \text{norm}(Q), \tag{11}$$

and this implies that the norm of each concurrent derivative Q_i in equation (10) is the sum of norms of all parallel components.

Now we apply Theorem 3 to Δ' and by (7) get a polynomial-time procedure to decide $\sim^S$. □

Evidently each $\doteq_i$ is an equivalence, hence the lemma applies and we conclude that the body of the main loop of the first phase requires only polynomial time: it amounts to invoking the decision procedure for strong bisimilarity on normed BPP polynomially many times, since the set $\mathbb{L}\times\mathbb{L}$ has polynomial cardinality. By Lemma 4 the second phase can be computed in polynomial time as well. Correctness of the algorithm follows by Lemmas 3 and 1. This completes the proof of the following:

Theorem 4. *There exists a polynomial-time algorithm to decide distributed bisimilarity for* BPP *processes.*

4 Final Remarks

We have proposed a polynomial-time decision procedure for distributed bisimilarity on BPP. As mentioned in the introduction, many non-interleaving equivalences coincide on BPP. Therefore, we directly conclude from Theorem 4:

Corollary 1. *There exists a polynomial-time algorithm to decide the following equivalences for* BPP *processes: location equivalence, causal equivalence, history preserving equivalence, performance equivalence.*

Consider BPP_τ, an extension of BPP with communication between parallel components expressed by one additional rule:

$$\frac{P \xrightarrow{a} [P', P''] \qquad Q \xrightarrow{\bar{a}} [Q', Q'']}{P \| Q \xrightarrow{\tau} [P' \| Q', P'' \| Q'']}. \tag{12}$$

A local derivative of a τ-transition can be composed of two local derivatives of parallel components. Hence local derivatives cannot be encoded directly into actions, and the reduction of distributed bisimilarity to strong bisimilarity in the proof of Lemma 4 fails.

A crucial ingredient of our decision procedure is the polynomial-time transformation of a process definition to the standard form, described in the full version of this paper [24]. It is different from the transformation proposed by Christensen in [9], since the process definition in standard form yielded by the latter is of exponential size.

Our algorithm needs $\Theta(n^2)$ calls to the polynomial-time algorithm of [16] in each iteration in the first phase, where n stands for the size of Δ. At most n iterations are needed, since all $\doteq_i$ are equivalences, and therefore total cost is $\Theta(n^3)$ calls to the procedure of [16]. On the other hand, P-completeness of the problem follows easily since it subsumes strong bisimilarity for finite-state systems and the latter is P-complete [3]. An interesting continuation of the work would be to develop a more efficient direct algorithm, not referring to the procedure of [16].

Acknowledgements The author is very grateful to Philippe Schnoebelen for many fruitful discussions.

References

1. L. Aceto. History preserving, causal and mixed-ordering equivalence over stable event structures. *Fundamenta Informaticae*, 17:319–331, 1992.
2. L. Aceto. Relating distributed, temporal and causal observations of simple processes. *Fundamenta Informaticae*, 17:369–397, 1992.
3. J. Balcázar, J. Gabarró, and M. Sántha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, (6A):638–648, 1992.
4. M. Bednarczyk. Hereditary history preserving bisimulation or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences, Gdańsk, 1991.
5. B. Bérard, A. Labroue, and P. Schnoebelen. Verifying performance equivalence for timed Basic Parallel Processes. In *Proc. FOSSACS'00, LNCS 1784*, pages 35–47, 2000.
6. I. Castellani. *Bisimulations for Concurrency*. PhD thesis, University of Edinburg, 1988.
7. I. Castellani. Process algebras with localities. In *J. Bergstra, A. Ponse, S. Smolka, eds., Handbook of Process Algebra, chapter 15*, pages 945–1046, 2001.
8. S. Christensen. Distributed bisimilarity is decidable for a class of infinite state systems. In *Proc. 3th Int. Conf. Concurrency Theory (CONCUR'92), LNCS 630*, pages 148–161, 1992.
9. S. Christensen. *Decidability and Decomposition in process algebras*. PhD thesis, Dept. of Computer Science, University of Edinburgh, UK, 1993.
10. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation equivalence is decidable for Basic Parallel Processes. In *Proc. CONCUR'93, LNCS 713*, pages 143–157, 1993.
11. P Darondeau and P. Degano. Causal trees. In *Proc. ICALP'89, LNCS 372*, pages 234–248, 1989.
12. S. Fröschle. Decidability of plain and hereditary history-preserving bisimulation for BPP. In *Proc. EXPRESS'99, volume 27 of ENTCS*, 1999.
13. R. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Proc. MFCS'89, LNCS 379*, pages 237–248, 1989.
14. R. Gorrieri, M. Roccetti, and E. Stancampiano. A theory of processes with durational actions. *Theoretical Computer Science*, 140(1):73–94, 1995.
15. M. Hennessy and A. Kiehn. On the decidability of non-interleaving process equivalences. In *Proc. 5th Int Conf. Concurrency Theory (CONCUR'94)*, pages 18–33, 1994.
16. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial time algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Mathematical Structures in Computer Science*, 6:251–259, 1996.
17. H. Hüttel. Undecidable equivalences for basic parallel processes. In *Proc. TACS'94, LNCS 789*, pages 454–464, 1994.
18. P. Jančar. Bisimilarity of basic parallel processes is PSPACE-complete. In *Proc. LICS'03, to appear*, 2003.
19. L Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154:107–143, 1996.
20. M. Jurdziński and M. Nielsen. Hereditary history preserving bisimilarity is undecidable. In *Proc. STACS'00, LNCS 1770*, pages 358–369, 2000.
21. A. Kiehn. A note on distributed bisimulations. Unpublished draft, 1999.
22. S. Lasota. Decidability of strong bisimilarity for timed BPP. In *Proc. 13th Int. Conf. on Concurrency Theory (CONCUR'02)*, LNCS 2421, pages 562–578. Springer-Verlag, 2002.
23. S. Lasota. On coincidence of distributed and performance equivalence for Basic Parallel Processes. `http://www.mimuw.edu.pl/~sl/papers/`, unpublished draft, 2002.
24. S. Lasota. A polynomial-time algorithm for deciding true concurrency equivalences of Basic Parallel Processes. Research Report LSV-02-13, LSV, ENS de Cachan, France, 2002.
25. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
26. J. Srba. Strong bisimilarity and regularity of Basic Parallel Processes is PSPACE-hard. In *Proc. STACS'02, LNCS 2285*, 2002.