Decomposition and Complexity of Hereditary History Preserving Bisimulation on BPP [*]

Sibylle Fröschle and Sławomir Lasota [**]

Institute of Informatics, Warsaw University
02–097 Warszawa, Banacha 2, Poland
{sib,sl}@mimuw.edu.pl

Abstract. We propose a polynomial-time decision procedure for hereditary history preserving bisimilarity (hhp-b) on Basic Parallel Processes (BPP). Furthermore, we give a sound and complete equational axiomatization for the equivalence. Both results are derived from a decomposition property of hhp-b, which is the main technical contribution of the paper. Altogether, our results complement previous work on complexity and decomposition of classical and history-preserving bisimilarity on BPP.

1 Introduction

The success of automatic verification in the finite-state world is contrasted by the reality that in practice most processes have either an infinite or an extremely large state space. Thus, it is important to clarify: how far can the automatic methods of the finite-state world be extended to infinite-state processes? It is folklore that full process calculi such as CCS are too expressive to allow for a decidable theory. However, there is now a standard hierarchy of restricted processes, the *Process Rewrite Systems* (*PRS*) hierarchy, along which the borderlines of decidability and complexity with respect to the major verification problems are well-investigated [19]. One central category of the PRS-hierarchy is *Basic Parallel Processes* (*BPP*): it can be seen as an extension of finite automata by a parallel composition operator. One of the major verification problems is to check whether two processes are equivalent under a given bisimulation equivalence.

With the recent addition of two more results our understanding of the computational power of bisimulation equivalences on BPP is now almost complete. On the one hand, the complexity of classical bisimilarity on BPP has finally been settled to be PSPACE-complete [18, 11]. On the other hand, [16] has established that truly-concurrent bisimulation equivalences, such as *history preserving bisimilarity* (*hp-b*), are P-complete for this class; in [12] the upper bound has been improved to $O(n^3)$, building on the technique of [11]. Together, these results indicate the following trend: while in the finite-state world truly-concurrent verification problems are at least as hard as their interleaving counterparts (e.g. [13, 15]), in the infinite-state world this effect seems reversed. The same trend has also been revealed in model-checking [4], and linear-time equivalence checking [20].

[*] This work is supported by the European Community Research Training Network GAMES.
[**] Partially supported by Polish KBN grant No. 4 T11C 042 25.

One gap remains in our understanding of bisimilarities on BPP: the complexity of *hereditary history preserving bisimilarity* (*hhp-b*) [2, 14]. Hhp-b is known to coincide with hp-b for *simple BPP* (*SBPP*) [6], and is thus polynomial-time decidable here. For full BPP it was shown to be decidable [5], but the proof left the complexity open. This paper fills this gap: we establish that hhp-b is polynomial-time decidable on BPP. Thereby we settle that hhp-b conforms to the positive trend for true-concurrency in the infinite-state world. This is particularly interesting since hhp-b takes a special position among bisimilarities: it is often considered to be *the* bisimulation equivalence for true-concurrency [14, 8]. Unlike all the other equivalences it is undecidable for finite-state systems [15]; only a few positive results could be achieved for restricted classes [7].

The reason behind the positive trend for true-concurrency in the infinite-state world seems to be the following: BPP processes have natural decomposition characteristics; these may translate into decomposition results for truly-concurrent equivalences, and allow us to decide the respective concept by a 'divide and conquer' approach. There are two kinds of decomposition results that one can consider. The classical question is [17]: given a process class and an equivalence, is each process term uniquely, up to the equivalence, represented as a parallel composition of prime processes? A process is prime if it cannot be expressed, up to the equivalence, as a non-trivial parallel composition. This type of decomposition stands behind the polynomial-time algorithm for bisimilarity on normed BPP by Hirshfeld et. al. [10]. Unique decomposition has also been shown for BPP with respect to distributed bisimilarity [3] (which coincides with hp-b for BPP).

As recently advocated in [6], in a truly-concurrent framework one can also consider whether a given equivalence is decomposable with respect to the independent components of the processes to be compared. If two processes P and Q are equivalent then we ask whether there is a one-to-one correspondence between the components of P and those of Q such that related components are equivalent. This kind of decomposition stands behind the coincidence of hhp-b with hp-b on SBPP: decomposition was proved for hp-b and hhp-b for a class that subsumes SBPP (and incomparable to BPP) [6]. Hp-b is not decomposable in general, and, as we will see later, neither is it for BPP.

As our core result, we will resolve that, modulo hhp-bisimilar choices (a concept to be explained later), hhp-b on BPP is indeed decomposable in the second sense. We will also show that an analogue for the choice operator holds. Building on our decomposition theory we will design a decision procedure for hhp-b on BPP, running in $O(n^2 \log n)$ time. Further, we will give a complete equational theory for hhp-b. The latter connects to work of Christensen, who presented equational theories for classical and distributed bisimilarity for BPP [3]. We proceed as follows. Section 2 contains the necessary definitions. In particular, we define hhp-b in terms of a *step game*. In Section 3 we prove our decomposition results. In Section 4 we present the algorithm and in Section 5 the equational theory. In Section 6 we discuss the consequences of our results, and highlight some further directions. Some proofs, missing here, can be found in [9].

2 Preliminaries

BPP. In the following assume a countably infinite set of actions $Act = \{a, b, \ldots\}$ and a countably infinite set of process variables $Vars = \{X, X_1, \ldots\}$. *BPP expressions* are given by the following grammar:

$$E ::= \mathbf{0} \mid X \mid a.E \mid E{+}E \mid E||E,$$

where $\mathbf{0}$ is the empty process, X is a process variable, $a.X$ is action prefix, $E{+}E$ denotes nondeterministic choice, and $E||E$ parallel composition. We usually consider BPP expressions modulo associativity and commutativity of choice and parallel composition, and $\mathbf{0}$ as unit for these operators. A *BPP definition* Δ is a finite family of recursive equations $X \stackrel{\text{def}}{=} E_X$, where the X are distinct variables, and each E_X is a BPP expression that only contains variables defined by Δ and where each variable occurrence is guarded, i.e., within the scope of action prefix. (This ensures that recursive definitions yield unique solutions.) The set of variables occurring in Δ is denoted by Vars_Δ. A *BPP process* is a pair (Δ, E), where Δ is a BPP definition, and E is an expression that only contains variables of Vars_Δ. If Δ is clear from the context, we denote (Δ, E) simply by E.

Execution Normal Form. We will mainly work with BPP in *Execution Normal Form* (*ENF*). *BPP expressions in ENF* (*ENF expressions*) are defined by:

$$E ::= \mathbf{0} \mid a.X \mid E{+}E \mid E||E.$$

Each BPP process (E, Δ) can easily be transformed into a process in ENF, $\mathtt{enf}(E)$. During the transformation, always work modulo $\mathbf{0}$ as unit for '$+$' and '$||$': remove all superfluous occurrences of $\mathbf{0}$ in the expressions. Translate E and all defining expressions of Δ into ENF expressions by replacing each subexpression $a.E'$ by $a.X_{E'}$. Add new equations $X_{E'} \stackrel{\text{def}}{=} E'$ to Δ. E' is possibly unguarded. Therefore, replace each unguarded occurrence of a variable Y by E_Y. Treat such newly created defining expressions as the original ones, until finally all defining expressions will be in ENF. Note that this transformation only makes use of operations such as unfolding of variables and introduction of new variables for subexpressions, which will be respected by any behavioural equivalence.

Transition-based ENF. For our definition of hhp-b, given a BPP E, we need to be able to uniquely identify each occurrence of an action prefix within E. A convenient way to do so is to work with labelled transitions rather than actions. In the following, for each $a \in Act$, assume a countably infinite set of transitions labelled by a, $T_a = \{t^a, t_1^a, \ldots\}$. Let $T = T_a \cup T_b \cup \ldots$ be the set of all transitions. Let $t, t_1, \ldots$ range over T, and set $l(t) = a$ if $t \in T_a$. *Transition-based ENF* (*T-ENF*) *expressions* are defined as follows:

$$E ::= \mathbf{0} \mid t.X \mid E{+}E \mid E||E,$$

where $t.X$ is transition prefix. We denote the set of transitions occurring in E by T_E. We only consider T-ENF expressions E that are *transition-genuine* in that every $t \in T_E$ appears syntactically only once in E. Given $t \in T_E$, there will be exactly one X such that '$t.X$' is a subexpression of E; denote X by X_t. Given a definition Δ, by $T\text{-}ENF_\Delta$ we denote all T-ENF expressions E such that E only contains variables of Vars_Δ.

Proviso. In the following, we mainly work with T-ENF processes. We allow us to assume that all defining expressions in a definition Δ are in T-ENF, and that $\texttt{enf}(E)$ is in T-ENF as well. Clearly, whatever we state for T-ENF processes can be carried over to ENF processes obtained by replacing all transitions with their labels.

Steps of T-ENF Processes. Rather than providing an operational semantics for T-ENF processes we prefer to capture the *concurrent steps* of a T-ENF expression E, i.e., the sequences of pairwise concurrent transitions initially enabled at E. This will be sufficient for our definition of hhp-b.

We say a transition t is enabled at E, written $E \xrightarrow{t}$, iff $t \in T_E$. If $E \xrightarrow{t}$ then the *parallel remainder* of E wrt. t, written $pR(E, t)$, is inductively defined as follows, where we work modulo $\mathbf{0}$ as unit for '$+$' and '$||$':

$$\begin{aligned} pR(t.X, t) &= \mathbf{0}, \\ pR(E{+}F, t) &= \texttt{if } t \in T_E \texttt{ then } pR(E, t) \texttt{ else } pR(F, t), \\ pR(E||F, t) &= \texttt{if } t \in T_E \texttt{ then } pR(E, t)||F \texttt{ else } E||pR(F, t). \end{aligned}$$

We say $r = t_1 \dots t_n \in T_E^*$ is a *concurrent step* of E, denoted by $r \in \mathit{steps}(E)$, iff there is a sequence $E_1, \dots, E_n$ such that $E = E_1$, and $\forall i \in [1, n]$, $E_i \xrightarrow{t_i}$ and $E_{i+1} = pR(E_i, t_i)$. We generalize $pR(E, t)$ to steps in the obvious way. Given $r \in \mathit{steps}(E)$ and $t \in T_E$, we say t is enabled at r, written $r \xrightarrow{t}$, iff $E' \xrightarrow{t}$, where $E' = pR(E, r)$. E' is a parallel remainder of E, written $E' \in pR(E)$, iff $E' = pR(E, r)$ for some $r \in \mathit{steps}(E)$.

Step Game and Hhp-b. The usual way to define hhp-b for BPP would be to proceed as follows: first, give the standard definition of hhp-b for, say, 1-safe Petri nets; second, define true-concurrency semantics for BPP so that each BPP is interpreted as a (typically infinite) 1-safe Petri net; and third, define two BPP to be hhp-bisimilar iff their interpretations as 1-safe Petri nets are hhp-bisimilar ([5]). To avoid the bulk of definitions this would require, we define hhp-b in a non-standard way, making use of a characterization of [5]: two T-ENF processes E and F are hhp-bisimilar iff: (1) Duplicator has a winning strategy $\mathcal{H}$ in a bisimulation game with backtracking, which is *only* played in the scope of the concurrent steps of E and F; and (2) whenever two transitions t_E and t_F are related by $\mathcal{H}$ then X_{t_E} and X_{t_F} are hhp-bisimilar.

Let E be a T-ENF expression, and $r = t_1 t_2 \dots t_n \in \mathit{steps}(E)$. Write $|r|$ for the length of r, that is $|r| = n$. Given $k \in [1, |r|]$, we define $\delta(r, k)$ to be the result of backtracking the kth transition in r, that is $\delta(r, k) = t_1 \dots t_{k-1} t_{k+1} \dots t_n$. Observe that we have $\delta(r, k) \in \mathit{steps}(E)$. Given $\mathcal{H} \subseteq \mathit{steps}(E) \times \mathit{steps}(F)$, we define $\mathit{Matches}(\mathcal{H})$ to be the set $\{(t_E, t_F) \mid (r_E t_E r'_E, r_F t_F r'_F) \in \mathcal{H},\ \text{where } |r_E| = |r_F|\}$.

Let E and F be T-ENF expressions. The $(E, F)_{\mathit{step}}$*-game* between Spoiler and Duplicator is played as follows. Configurations are pairs $(r_E, r_F) \in \mathit{steps}(E) \times \mathit{steps}(F)$ with $|r_E| = |r_F|$. The initial configuration is $(\varepsilon, \varepsilon)$. A play proceeds from (r_E, r_F) by the following rules:

1. Spoiler chooses one of E or F, say E, and picks a transition $t_E \in T_E$ that is enabled at r_E. Duplicator has to respond by executing a transition t_F in F that is enabled at r_F and satisfies $l(t_E) = l(t_F)$. Play continues at $(r_E t_E, r_F t_F)$.

2. Alternatively, Spoiler chooses one of E or F, say E; he picks $k \in [1, |r_E|]$, and backtracks the kth transition in r_E. Duplicator has to backtrack the corresponding transition in r_F. Play resumes at $(\delta(r_E, k), \delta(r_F, k))$.
3. The play continues like this forever, in which case Duplicator wins, or until either Spoiler or Duplicator is unable to move, in which case the other participant wins.

Note that a play can continue indefinitely only because of repeated backward and forward steps which may undo each other.

A winning strategy for Duplicator in the $(E, F)_{step}$-game is a set of configurations $\mathcal{H}$ such that $(\varepsilon, \varepsilon) \in \mathcal{H}$ and whenever Spoiler has a move at some $(r_E, r_F) \in \mathcal{H}$ then Duplicator has a response and the accordingly updated configuration is in $\mathcal{H}$.

Let Δ be a BPP definition in T-ENF. We map a relation $\sim \subseteq T\text{-}ENF_\Delta \times T\text{-}ENF_\Delta$ to a relation $\dot{\sim} \subseteq T\text{-}ENF_\Delta \times T\text{-}ENF_\Delta$ as follows: $E \mathrel{\dot{\sim}} F$ iff Duplicator has a winning strategy $\mathcal{H}$ in the $(E, F)_{step}$-game such that for all $(t_E, t_F) \in Matches(\mathcal{H})$, $X_{t_E} \sim X_{t_F}$ (by convention, for variables X and Y we write $X \sim Y$ if $E_X \sim E_Y$). In [9] it is proved that the standard definition of hhp-b on BPP (e.g. [5]) is equivalent to:

Definition 1. *Hhp-b, denoted by $\sim_{hhp}$, is the greatest relation $\sim$ such that $\sim \, = \, \dot{\sim}$. We carry over $\sim_{hhp}$ to all BPP processes: $E \sim_{hhp} F$ iff* $\mathtt{enf}(E) \sim_{hhp} \mathtt{enf}(F)$.

3 Decomposition

Let Δ be a BPP definition in T-ENF. All processes that appear in this section are assumed to be in $T\text{-}ENF_\Delta$. We define the *summands* and *factors* of a process inductively as follows:

$$\begin{array}{rlrl}
\mathtt{summands}(E_1{+}E_2) &= \mathtt{summands}(E_1) \cup \mathtt{summands}(E_2) & \mathtt{summands}(t.X) &= \{t.X\} \\
\mathtt{summands}(E_1||E_2) &= \{E_1||E_2\} & \mathtt{summands}(\mathbf{0}) &= \emptyset \\
\mathtt{factors}(E_1{+}E_2) &= \{E_1{+}E_2\} & \mathtt{factors}(t.X) &= \{t.X\} \\
\mathtt{factors}(E_1||E_2) &= \mathtt{factors}(E_1) \cup \mathtt{factors}(E_2) & \mathtt{factors}(\mathbf{0}) &= \emptyset.
\end{array}$$

We investigate whether hhp-b is decomposable wrt. parallel composition in the following sense: whenever E and F are hhp-bisimilar is there a bijection between the factors of E and those of F such that related factors are hhp-bisimilar? We also ask whether hhp-b is decomposable wrt. choice in the analogous sense. In view of Section 4 we prove our decomposition results in a more general formulation: we work with $\dot{\sim}$ rather than $\sim_{hhp}$, where we assume $\sim \subseteq T\text{-}ENF_\Delta \times T\text{-}ENF_\Delta$ to be an arbitrary equivalence.

A first observation is that we will have to work modulo choices that are trivial wrt. $\dot{\sim}$: let $P = E||F$ and $Q = P'{+}P''$ such that $P \mathrel{\dot{\sim}} P' \mathrel{\dot{\sim}} P''$; clearly P is equivalent to Q under any reasonable behavioural equivalence, but there is no bijection between the factors of P and those of Q. Formally, we capture trivial choices as follows.

Definition 2. *We say that E* contains a trivial choice wrt. $\dot{\sim}$ *if it contains, up to associativity and commutativity of $+$, a subexpression $E_1{+}E_2$ with $E_1 \mathrel{\dot{\sim}} E_2$. When $\dot{\sim} \, = \, \sim_{hhp}$, we say that E* contains a hhp-bisimilar choice.

We will prove that, modulo trivial choices, $\dot\sim$ is indeed decomposable wrt. both operators. The proof of decomposition wrt. parallel composition relies on three lemmas. The first is a cancellation lemma, which holds in general.

Lemma 1. $F||E \dot\sim G||E \Longrightarrow F \dot\sim G$.

Proof. For shorter notation we set $L = F||E$ and $R = G||E$. Assume a winning strategy $\mathcal{H}$ for Duplicator in the $(L, R)_{step}$-game such that for all $(t_L, t_R) \in Matches(\mathcal{H})$, $X_{t_L} \sim X_{t_R}$.

Based on $\mathcal{H}$ we exhibit a winning strategy $\mathcal{H}'$ for Duplicator in the $(F, G)_{step}$-game. The idea behind the construction of $\mathcal{H}'$ is as follows. Assume, in the $(F, G)_{step}$-game, Spoiler picks a transition in F, say t_F, as his first move. This move can be copied to the $(L, R)_{step}$-game. According to $\mathcal{H}$, Duplicator has a reply, say t_m, either in E or in G. If the latter holds then Duplicator can copy t_m straight to the (F, G)-game. But what to do if t_m is in E? Then Duplicator can obtain her answer to t_F by the following 'zig-zag'-strategy. Spoiler can choose t_m in L as his next move in the $(L, R)_{step}$-game. If, according to $\mathcal{H}$, Duplicator's answer, say t'_m, is in G, take t'_m to be her reply to t_F in the $(F, G)_{step}$-game. Otherwise, let Spoiler perform t'_m in L as his next move in the $(L, R)_{step}$-game, and check whether this time Duplicator's answer is in G. We repeat this procedure, until, finally, we hit a match in G. In this manner, we will exhibit answers for Duplicator not only to Spoiler's first moves but to all of his moves.

We will make use of the following two observations, where $r_L \in steps(L)$, $r_R \in steps(R)$, $r_F \in steps(F)$, and $r_G \in steps(G)$. We use a notation $r \upharpoonright T$ for projection of a concurrent step r on a set of transitions T, i.e., $r \upharpoonright T$ is a concurrent step obtained by dropping all transitions of r that are not in T.

1. If $r_L \upharpoonright T_E = r_R \upharpoonright T_E$ then $\forall t_E \in T_E, r_L \stackrel{t_E}{\to} \iff r_R \stackrel{t_E}{\to}$.
2. If $r_L \upharpoonright T_F = r_F$ then $\forall t_F \in T_F, r_L \stackrel{t_F}{\to} \iff r_F \stackrel{t_F}{\to}$. And, in analogy:
   If $r_R \upharpoonright T_G = r_G$ then $\forall t_G \in T_G, r_R \stackrel{t_G}{\to} \iff r_G \stackrel{t_G}{\to}$.

Formally, we construct $\mathcal{H}'$ inductively from the initial configuration while preserving the following property:

Property P. Let $(r_F, r_G) \in \mathcal{H}'$; (r_F, r_G) is of the form $(t_F^1 \dots t_F^m, t_G^1 \dots t_G^m)$, where $m \geq 0$ and $\forall i \in [1, m]$, $t_F^i \in T_F$, $t_G^i \in T_G$. Then there is $(r_L, r_R) \in \mathcal{H}$ such that $r_L = w_L^1 \dots w_L^m$, $r_R = w_R^1 \dots w_R^m$, and $\forall i \in [1, m]$, w_L^i and w_R^i are of the form

$$\begin{aligned} w_L^i &= t_F^i t_E^1 \dots t_E^n, \\ w_R^i &= t_E^1 \dots t_E^n t_G^i, \end{aligned} \qquad \text{or} \qquad \begin{aligned} w_L^i &= t_E^1 \dots t_E^n t_F^i, \\ w_R^i &= t_G^i t_E^1 \dots t_E^n, \end{aligned}$$

where $n \geq 0$ and $\forall j \in [1, n]$, $t_E^j \in T_E$.

Base case. We start with $(\varepsilon, \varepsilon) \in \mathcal{H}'$. Property (P) trivially holds since $(\varepsilon, \varepsilon) \in \mathcal{H}$.

Inductive case. Let $(r_F, r_G) \in \mathcal{H}'$. Spoiler chooses his next move according to rule (1) or (2) of the game. Assume $(r_L, r_R) \in \mathcal{H}$ as given by (P). In either case, we construct a response for Duplicator such that (P) is preserved.

(1) Spoiler chooses one of F or G, say F, and performs a transition t_F of F that is enabled at r_F. Consider the $(L,R)_{step}$-game. Let Spoiler perform t_F at (r_L, r_R); this is possible by Observation (2). Say Duplicator's response according to $\mathcal{H}$ is t_m. We obtain a match for t_F in the $(F,G)_{step}$-game by the following 'zig-zag' algorithm:

$r_L := r_L t_F; r_R := r_R t_m;$ -- update the configuration
`while` $t_m \notin T_G$ `do`
  let Spoiler perform t_m in L;
  set t'_m to be Duplicator's response according to $\mathcal{H}$;
  $r_L := r_L t_m; r_R := r_R t'_m;$ -- update the configuration
  $t_m := t'_m;$ -- update the match
`return` t_m;

The following is an invariant of the while-loop: let r'_R be given by r_R minus Duplicator's last match; (a) $r_L \restriction T_E = r'_R \restriction T_E$, and (b) $r'_R \restriction T_G = r_G$. By (a) and Observation (1), the first instruction of the while-loop is indeed a valid move in the $(L,R)_{step}$-game. The algorithm clearly terminates: there is only a finite number of transitions in T_E. We take t_m to be Duplicator's response to t_F in the $(F,G)_{step}$-game; by (b) and Observation (2) this is a legal move. Thus, we extend $\mathcal{H}'$ by $(r_F t_F, r_G t_m)$. Property (P) will be preserved: at the last stage of the algorithm (r_L, r_R) is a configuration as required.

(2) Spoiler chooses one of F or G, say F; he picks $k \in [1, |r_F|]$, and backtracks the kth transition in r_F. Duplicator must backtrack the kth transition in r_G. We add $(\delta(r_F,k), \delta(r_G,k))$ to $\mathcal{H}'$. Property (P) will be preserved by this addition. In the $(L,R)_{step}$-game, at (r_L, r_R), let Spoiler backtrack all the w_L^k-transitions. Then $(w_L^1 \dots w_L^{k-1} w_L^{k+1} \dots w_L^m, w_R^1 \dots w_R^{k-1} w_R^{k+1} \dots w_R^m) \in \mathcal{H}$; but this is exactly a configuration as required.

It remains to check whether for all $(t_F, t_G) \in \mathit{Matches}(\mathcal{H}')$, $X_{t_F} \sim X_{t_G}$. Let $(t_F, t_G) \in \mathit{Matches}(\mathcal{H}')$. If $(t_F, t_G) \in \mathit{Matches}(\mathcal{H})$ then $X_{t_F} \sim X_{t_G}$ is immediate. Otherwise, wlog. assume (P) gives us (t_F, t_E^1), (t_E^1, t_E^2), …, $(t_E^n, t_G) \in \mathit{Matches}(\mathcal{H})$, where $n > 0$, and $\forall i \in [1,n]$, $t_E^i \in T_E$. We know that $X_{t_F} \sim X_{t_E^1}$, $\forall i \in [1, n-1]$, $X_{t_E^i} \sim X_{t_E^{i+1}}$, and $X_{t_E^n} \sim X_{t_G}$. But then $X_{t_F} \sim X_{t_G}$ follows by transitivity of $\sim$. □

Relation $\dot\sim$ is a congruence with respect to parallel composition; hence we also obtain:

Corollary 1. *$(E \dot\sim E')$ & $(F||E \dot\sim F'||E') \Longrightarrow F \dot\sim F'$.*

The second lemma implies: if a choice and a parallel composition are related by $\dot\sim$ then the choice must be trivial wrt. $\dot\sim$.

Lemma 2. *If $E \dot\sim F$ and $|\mathtt{factors}(F)| \geq 2$ then for each $G \in \mathtt{summands}(E)$, $G \dot\sim F$.*

Proof. Let E and F be given as above. If $|\mathtt{summands}(E)| < 2$ then the lemma is immediate. Otherwise, let $\mathcal{H}$ be a winning strategy for Duplicator in the $(E,F)_{step}$-game such that for all $(t_E, t_F) \in \mathit{Matches}(\mathcal{H})$, $X_{t_E} \sim X_{t_F}$. Choose any $G \in \mathtt{summands}(E)$. We will exhibit a winning strategy $\mathcal{H}'$ for Duplicator in the $(G,F)_{step}$-game such that

$(t_G, t_F) \in \mathit{Matches}(\mathcal{H}')$ only if $(t_G, t_F) \in \mathit{Matches}(\mathcal{H})$. This will clearly yield $G \dot{\sim} F$.

If Spoiler picks a transition in G as his first move then Duplicator can copy her response and all subsequent moves straight from $\mathcal{H}$. This is so because: once we have decided for G, the other E-summands become disabled, and, from this point onwards, the $(E, F)_{step}$-game corresponds exactly to the $(G, F)_{step}$-game. Similarly, if Spoiler picks a transition in F as his first move, and, according to $\mathcal{H}$, Duplicator responds with a G-transition then she can copy this response and all subsequent moves from $\mathcal{H}$. The difficult case is when Spoiler performs his first move in F, say he executes t_F, and $\mathcal{H}$ prescribes a match in a E-summand other than G. We show that, in this case, Duplicator has an alternative match in G: we exhibit $t_E \in G$ such that $(t_E, t_F) \in \mathcal{H}$.

Consider the $(E, F)_{step}$-game. At $(\varepsilon, \varepsilon)$, let Spoiler perform a transition in G, say t_G; this is clearly possible. Assume, according to $\mathcal{H}$, Duplicator answers this move by t'_F. There are three cases:

(a) $t'_F = t_F$, (b) t'_F and t_F are concurrent in F, (c) t'_F and t_F are in conflict in F.

(Given an expression E, two distinct transitions $t, t' \in T_E$ are in conflict in E if there is a subexpression $E_1 + E_2$ of E with $t \in T_{E_1}$ and $t' \in T_{E_2}$; otherwise t, t' are concurrent in E.)

If (a) holds then t_G is a match as required. In case (b) Spoiler can perform t_F as his next move. Duplicator must match t_F by a transition in G, say t'_G. Let Spoiler backtrack t'_F. Duplicator must backtrack t_G. We arrive at $(t'_G, t_F) \in \mathcal{H}$, and thus t'_G is a match as required. Finally, assume (c) holds. t_F and t'_F must belong to the same factor of F, say H. There must be a further factor of F, say H'. Spoiler can perform a transition in H', say t''_F, as his next move. Duplicator must match t''_F by a G-transition, say t'_G. Let Spoiler backtrack t'_F. Duplicator must backtrack t_G. We arrive at $(t'_G, t''_F) \in \mathcal{H}$; but from here we can proceed exactly as in (b). $\square$

With the help of the previous lemma we will show: given $E \dot{\sim} F$, where E and F are non-zero and contain no trivial choice, we can always find a factor G of E and a factor H of F such that $G \dot{\sim} H$. This will ensure that we can apply Corollary 1 consecutively to obtain our decomposition result.

Lemma 3. *Assume that E and F contain no trivial choice wrt. $\dot{\sim}$, and $E \neq \mathbf{0}$ or $F \neq \mathbf{0}$. If $E \dot{\sim} F$ then there exist $G \in$ `factors`(E) and $H \in$ `factors`(F) such that $G \dot{\sim} H$.*

Proof. Wlog. assume $E \neq \mathbf{0}$. Let $\mathcal{H}$ be a winning strategy for Duplicator in the $(E, F)_{step}$-game such that for all $(t_E, t_F) \in \mathit{Matches}(\mathcal{H})$, $X_{t_E} \sim X_{t_F}$. Choose any $G \in$ `factors`(E), and consider $r_E \in \mathit{steps}(E)$ such that $pR(E, r_E) = G$; this is clearly possible. There must be $r_F \in \mathit{steps}(F)$ such that $(r_E, r_F) \in \mathcal{H}$. Set $F' = pR(F, r_F)$. It is straightforward to derive $G \dot{\sim} F'$. One of the following three cases will hold:

1. $F' \in$ `factors`(F).
2. $F' \in pR(H)$ and $F' \neq H$ for some $H \in$ `factors`(F).

3. $F' = H'_1||\ldots||H'_n$, where $n \geq 2$ and $\forall i \in [1,n]$, $H'_i \neq \mathbf{0}$ and $H'_i \in pR(H)$ for some $H \in \mathtt{factors}(F)$.

If (1) holds then G and F' are factors as required. If (2) applies, at (r_E, r_F), let Spoiler backtrack all the H-transitions in r_F. Duplicator must backtrack the corresponding transitions. The new configuration, say (r'_E, r'_F), satisfies: $pR(F, r'_F) = H$ and $pR(E, r'_E) = G||G'_1||\ldots||G'_n$, where $n \geq 1$ and $\forall i \in [1,n]$, $G'_i \neq \mathbf{0}$ and $G'_i \in pR(G')$ for some $G' \in \mathtt{factors}(E)$. But this means (2) reduces to (3): wlog. we can exchange G by H. Finally, assume (3) holds. G cannot be of the form $t.X$: we have $G \mathrel{\dot\sim} F'$ but there are at least two concurrent transitions in F' for Duplicator to match. Since G cannot be a parallel composition either, we conclude $|\mathtt{summands}(G)| \geq 2$. But then we can apply Lemma 2 to obtain a contradiction with our assumption that E does not contain any trivial choice wrt. $\dot\sim$. □

Now, we are ready to prove decomposition wrt. parallel composition.

Theorem 1. *Assume E and F contain no trivial choice wrt. $\dot\sim$. If $E \mathrel{\dot\sim} F$ then there exists a bijection $\beta : \mathtt{factors}(E) \to \mathtt{factors}(F)$ such that $G \mathrel{\dot\sim} \beta(G)$ for each $G \in \mathtt{factors}(E)$.*

Proof. Set $m = |\mathtt{factors}(E)|$. The proof is by induction on m. If $m = 0$ then we must also have $|\mathtt{factors}(F)| = 0$, and a bijection as required is trivially given. If $m > 0$, we can apply Lemma 3 to obtain $G \in \mathtt{factors}(E)$ and $H \in \mathtt{factors}(F)$ such that $G \mathrel{\dot\sim} H$. Let E' be given by $E = E'||G$, and F' by $F = F'||H$. Corollary 1 gives us $E' \mathrel{\dot\sim} F'$, and, applying the induction hypothesis, we easily obtain a bijection as required. □

Decomposition wrt. choice is not as involved to prove. It is a consequence of the following theorem.

Theorem 2. *If $E \mathrel{\dot\sim} F$ then*

- *for each $G \in \mathtt{summands}(E)$ there is $H \in \mathtt{summands}(F)$ such that $G \mathrel{\dot\sim} H$,*
- *for each $H \in \mathtt{summands}(F)$ there is $G \in \mathtt{summands}(E)$ such that $G \mathrel{\dot\sim} H$.*

Corollary 2. *Assume E and F contain no trivial choice wrt. $\dot\sim$. If $E \mathrel{\dot\sim} F$ then there exists a bijection $\beta : \mathtt{summands}(E) \to \mathtt{summands}(F)$ such that $G \mathrel{\dot\sim} \beta(G)$ for each $G \in \mathtt{summands}(E)$.*

Although we will build on Theorem 1 and Corollary 2 it is worth spelling them out for the special case $\sim \; = \; \sim_{hhp}$. By definition of hhp-b all the previous results carry over, and we obtain decomposition of hhp-b wrt. all BPP operators.

Corollary 3. *Assume E and F contain no hhp-bisimilar choice. If $E \sim_{hhp} F$ then*

- *there exists a bijection $\beta : \mathtt{factors}(E) \to \mathtt{factors}(F)$ such that $G \sim_{hhp} \beta(G)$ for each $G \in \mathtt{factors}(E)$;*
- *there exists a bijection $\beta : \mathtt{summands}(E) \to \mathtt{summands}(F)$ such that $G \sim_{hhp} \beta(G)$ for each $G \in \mathtt{summands}(E)$.*

Note that $\dot\sim$, including $\sim_{hhp}$, is clearly compositional, i.e., preserved by '$+$' and '$||$'; hence the opposite directions of Theorems 1, 2 and Corollaries 2, 3 hold as well.

4 Algorithm

Let Δ be a BPP definition in T-ENF. By convention, let E_X denote the defining expression of a variable X. Let n be the size of Δ, i.e., the sum of lengths of all E_X. We will concentrate on relations $\sim \subseteq \mathit{Vars}_\Delta \times \mathit{Vars}_\Delta$ between variables in this section. Hence, in the following, symbol $\sim_{hhp}$ is used to denote hhp-b restricted to variables. We will show that $\sim_{hhp}$ can be computed in time polynomial wrt. n.

Define an operator $\mathcal{F}$ that given $\sim \subseteq \mathit{Vars}_\Delta \times \mathit{Vars}_\Delta$ yields a relation $\mathcal{F}(\sim) \subseteq \mathit{Vars}_\Delta \times \mathit{Vars}_\Delta$, defined by: $\langle X, Y \rangle \in \mathcal{F}(\sim)$ iff $E_X \;\dot{\sim}\; E_Y$. $\mathcal{F}$ can be seen as the restriction of the mapping $\sim \mapsto \dot{\sim}$ to variables. In particular, $\mathcal{F}$ is monotonic: if $\sim_1 \subseteq \sim_2$ then $\mathcal{F}(\sim_1) \subseteq \mathcal{F}(\sim_2)$. By Definition 1 we get:

Proposition 1. *$\sim_{hhp}$ is the greatest fixed point of $\mathcal{F}$.*

Hence, $\sim_{hhp}$ is the limit of the following sequence of approximants, where $\sim_0 = \mathit{Vars}_\Delta \times \mathit{Vars}_\Delta$:

$$\sim_0 \supseteq \mathcal{F}(\sim_0) \supseteq \mathcal{F}^2(\sim_0) \supseteq \ldots .$$

In other words, $\sim_{hhp}$ equals the first $\mathcal{F}^i(\sim_0)$ with $\mathcal{F}^i(\sim_0) = \mathcal{F}^{i+1}(\sim_0)$. It can easily be shown, by induction on i, that all the approximants are equivalence relations; hence the number of iterations is not greater than the number of variables. We only need to show that computing $\mathcal{F}^{i+1}(\sim_0)$ from $\mathcal{F}^i(\sim_0)$ can be done in polynomial time. We will prove:

Lemma 4. *Given an equivalence $\sim \subseteq \mathit{Vars}_\Delta \times \mathit{Vars}_\Delta$, relation $\mathcal{F}(\sim)$ can be computed in time $\mathcal{O}(n \log n)$.*

We will also show that checking whether the limit has been reached can be done without any extra cost. Thus, altogether we obtain:

Theorem 3. *Relation $\sim_{hhp}$ can be computed in time $\mathcal{O}(n^2 \log n)$.*

In the rest of this section we describe the algorithm announced in Lemma 4. It is inspired by the standard algorithm solving tree isomorphism (e.g. [1]). Our algorithm assigns an integer $i(v)$ to each node v of the syntactic trees corresponding to the defining expressions of Δ such that for any two nodes v_1, v_2 we have: $i(v_1) = i(v_2)$ iff the expressions represented by v_1 and v_2 are related by $\dot{\sim}$.

The nodes of the syntactic trees are of three types: prefix, '$+$' and '$||$'; the leaves are precisely the nodes of type prefix. We assume that the trees are constructed up to associativity and commutativity of '$+$' and '$||$'. In particular, if a node has type '$+$', its parent has type '$||$', and vice versa. The trees can be constructed in time $\mathcal{O}(n)$.

The algorithm works in bottom-up manner, visiting each of the nodes once. It starts in the leaves and each non-leaf is processed after all its children have been visited. In each node v, a sorted list l_v is computed. To start off with, l_v exactly contains the child nodes of v. To notionally remove trivial choices from the trees, l_v is processed such that: (1) if v is of type '$+$' then, for any integer j, l_v will contain at most one node v' with $i(v') = j$; (2) if v is of type '$||$' and some child v' of v has been identified as a trivial choice, i.e., v' is of type '$+$' and $l_{v'}$ contains only one node, say v'', then the nodes of

$l_{v''}$ will be inserted into l_v in place of v'. For convenience, we assume $l_v = \{v\}$ at each leaf v. A table T is used to store triples (j, x, t), where j is an integer assigned to some non-leaf node v, x is the type of v, and $t = i(l_v)$ is the sorted tuple of integers assigned to the nodes of l_v. The table T is initially empty.

Assign integers to all leaves such that two leaves $t.X$ and $t'.X'$ have the same integer iff $l(t) = l(t')$ and $X \sim X'$. This can clearly be done in time $\mathcal{O}(n)$. The processing of a non-leaf v depends on its type. First, we do the following:

let l_v be a tuple containing all child nodes of v
`if` v is of type '$+$'
  sort l_v wrt. the integers assigned to the nodes
  remove duplicates from l_v: $(*)$
    as long as $v_1, v_2 \in l_v$, $v_1 \neq v_2$ and $i(v_1) = i(v_2)$, remove one of v_1, v_2
  if l_v contains only one node, mark node v 'trivial choice'
`else` $--$ i.e., v is of type '$||$'
  `for each` node $v' \in l_v$ marked 'trivial choice'
    replace v' by the elements of $l_{v''}$, where $\{v''\} = l_{v'}$ $(**)$
  sort l_v wrt. the integers assigned to the nodes.

If v is marked 'trivial choice' then we assign to v the integer that has been given to the unique element of l_v. Otherwise, we perform a look-up in T. If a triple (j, x, t) is found with $t = i(l_v)$ and x the type of v, assign j to v. Otherwise, assign to v a fresh number j' and update T by inserting $(j', x', i(l_v))$ into T, where x' is the type of v.

After all nodes have been processed we assign to each variable X of Δ the integer assigned to the root of the tree that represents E_X. This yields a representation of $\mathcal{F}(\sim)$.

The correctness of the algorithm follows from Lemma 5 below. For its formulation and proof we adopt some conventions. Given an expression E, define the *children* of E, denoted by $\texttt{children}(E)$, as follows: if E is a choice then set $\texttt{children}(E) = \texttt{summands}(E)$, otherwise define $\texttt{children}(E) = \texttt{factors}(E)$. (If E is a prefix this implies $\texttt{children}(E) = \{E\}$.) Given a processed node v, let l_v^r be the 'real' tuple of v: if v is marked 'trivial choice' set l_v^r to be $l_{v'}$ where $\{v'\} = l_v$, otherwise set l_v^r to be l_v. We carry over $\dot{\sim}$ to nodes in the obvious way: e.g., given a node v, we write $E \mathrel{\dot{\sim}} v$ iff E and the expression represented by v are related by $\dot{\sim}$.

Lemma 5. *Let v, v_1, v_2 be nodes of the trees after termination of the algorithm.*

1. $v \mathrel{\dot{\sim}} E$ *for some process E such that*
   (a) *E does not contain any trivial choice;*
   (b) *there exists a bijection $\beta : l_v^r \to \texttt{children}(E)$ such that $v' \mathrel{\dot{\sim}} \beta(v')$ for each $v' \in l_v^r$;*
   (c) *if there is no entry for $i(v)$ in T then E is a prefix, otherwise E is of type x, where x is given by $(i(v), x, i(l_v^r)) \in T$.*
2. $v_1 \mathrel{\dot{\sim}} v_2$ *iff* $i(v_1) = i(v_2)$.

Proof (Sketch). The lemma follows by induction on the number of nodes that have already been processed. (1) If v is a prefix then take E to be v. Otherwise, for each $v_i \in l_v^r$ assume E_i such that $E_i \mathrel{\dot{\sim}} v_i$ as given by the induction hypothesis. Let x be defined by $(i(v), x, i(l_v^r)) \in T$. If $x =$ '$||$' then take E to be the parallel composition of the E_i, otherwise take E to be the choice of the E_i. Using the induction hypothesis of (2) it is routine to check that $E \stackrel{\bullet}{\sim} v$ and that conditions (a)–(c) are satisfied.

(2)($\Rightarrow$) Assume E_1 and E_2 such that $E_1 \stackrel{\bullet}{\sim} v_1$ and $E_2 \stackrel{\bullet}{\sim} v_2$ as given by (1). Since E_1 and E_2 do not contain any trivial choice we can apply Theorem 1 and Corollary 2 to obtain: E_1 and E_2 must be of the same type, and there is a bijection between the children of E_1 and those of E_2 such that related children are in $\stackrel{\bullet}{\sim}$. If E_1 and E_2 are of type prefix then $i(v_1) = i(v_2)$ can be derived immediately. Otherwise, using the induction hypothesis, we first obtain $i(l_{v_1}^r) = i(l_{v_2}^r)$, and then conclude $i(v_1) = i(v_2)$. ($\Leftarrow$) By a converse argument using congruence rather than decomposition. $\square$

Finally, we provide a cost estimation of the algorithm.

Claim. The algorithm runs in time $\mathcal{O}(L \cdot \log n)$, where $L = \sum_v |l_v|$ is the sum of lengths of all tuples l_v. (When v is a '$+$' node, we consider the length of l_v *before* removing duplicates in $(*)$.)

Indeed: sorting l_v requires $\mathcal{O}(|l_v| \cdot \log |l_v|)$ time; each look-up and update can be done in time $\mathcal{O}(|l_v| \cdot \log n)$ by bisection: T never contains more than n entries, and equality test for l_v requires at most time $|l_v|$ since all tuples are sorted. The crucial observation for the total cost estimation is the following:

Claim. L is $\mathcal{O}(n)$.

Each node w belongs to a tuple l_v of its parent v (before w may be removed from l_v during $(*)$). Moreover, a node can belong to several other tuples $l_{v'}$, due to the replacement $(**)$ in the algorithm. Obviously, L is equal to the total number of pairs $(w, v) : w \in l_v$. There are at most n such pairs with v being the parent of w. We will show that there are also at most n pairs with v *not* the parent of w. Concretely, we will injectively assign to each such pair a node in the tree.

Consider stage $(**)$ of the algorithm: assume a '$||$' node v with one of its children v' marked 'trivial choice'; let $l_{v'} = \{v''\}$, and assume $w \in l_{v''}$. Node v' has type '$+$' and v'' can either be of type prefix or of type '$||$'. We will assign to the pair (w, v) a node as follows. There must be a second child $\bar{v}''$ of v' which satisfies $i(\bar{v}'') = i(v'')$; $\bar{v}''$ must have been removed from $l_{v'}$ at an earlier stage of the algorithm. We have $i(l_{v''}) = i(l_{\bar{v}''})$. Assign to (w, v) a corresponding node $\bar{w}$ in $l_{\bar{v}''}$. (Note that $\bar{w}$ is not necessarily a child of $\bar{v}''$.) In total, all pairs (w, v) with $w \in l_{v''}$ can be assigned injectively to the nodes of $l_{\bar{v}''}$. The crucial observation is that a node from $l_{\bar{v}''}$ will not be assigned to any other pair again later, since $\bar{v}''$ has been removed from $l_{v'}$. This implies that the mapping is injective.

To complete the cost estimation, we note that checking whether $\sim\, = \mathcal{F}(\sim)$ can be done without any extra cost. It can be shown that, as long as the limit has not been reached, in each iteration of the algorithm the set of nodes marked 'trivial choice' is a strict subset of the nodes thus marked in the previous iteration. Hence, let the overall algorithm terminate when no node is 'unmarked' during the current iteration.

5 Equational Theory

In this section we work with general BPP expressions. We give a complete equational theory for hhp-b. That is to say, $E \sim_{hhp} F$ if and only if $\vdash E = F$ can be derived within the theory. Our approach is sequent-based (similarly to [3]), i.e., we provide a set of axioms of the form $\Gamma \vdash E = F$, to be read as "$E = F$ is provable under assumption Γ", where Γ is a finite set of equations. We write $\vdash E = F$ when Γ is empty. Interestingly, our axiomatization is essentially the same as that given by [3] for hp-b on SBPP, a subclass for which hhp-b and hp-b coincide [6]. We work relative to a BPP definition Δ.

Summation

(S1) $\Gamma \vdash E{+}F = F{+}E$

(S2) $\Gamma \vdash E{+}(F{+}G) = (E{+}F){+}G$

(S3) $\Gamma \vdash E{+}\mathbf{0} = E$

(S4) $\Gamma \vdash E{+}E = E$

Composition

(P1) $\Gamma \vdash E||F = F||E$

(P2) $\Gamma \vdash E||(F||G) = (E||F)||G$

(P3) $\Gamma \vdash E||\mathbf{0} = E$

Recursion

(R1) $\Gamma, E = F \vdash E = F$

(R2) $\dfrac{\Gamma, X = F \vdash E_X = F}{\Gamma \vdash X = F}\ (X \stackrel{\text{def}}{=} E_X) \in \Delta$

Axioms (S1)-(S3) and (P1)-(P3) are the commutative monoid laws for summation and parallel composition. Axiom (S4) is idempotency for summation. Rules (R1)-(R2) are laws for recursion and can be seen as an instance of fixed-point induction. In particular, Rule (R2) says that in order to prove a goal $X = F$ under assumption Γ, one is allowed to replace X by its defining expression. Moreover, the additional assumption $X = F$ is added to Γ, which guarantees immediate termination of the proof, by (R1), whenever a subgoal $X = F$ is to be proved again.

In addition, we need standard equivalence rules (E1)-(E3) and substitutivity rules (C1)-(C3):

Equivalence

(E1) $\Gamma \vdash E = E$ (E2) $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash F = E}$ (E3) $\dfrac{\Gamma \vdash E = F \quad \Gamma \vdash F = G}{\Gamma \vdash E = G}$

Congruence

(C1) $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash a.E = a.F}$ (C2) $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash E||G = F||G}$ (C3) $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash E{+}G = F{+}G}$

A proof of $\Gamma \vdash E = F$ is in the form of a finite tree, whose root is labelled by $\Gamma \vdash E = F$, leaves are instances of axioms and the children of each non-leaf are determined by an instance of some rule (in fact, only (E3) admits more than one child). We write $\Gamma \vdash E = F$ when such a proof exists. (It would be more precise to write $\Gamma \vdash_\Delta E = F$; however, we assume that Δ is clear from the context.)

Soundness of the theory for hhp-b is intuitively clear: one would expect each rule to be respected by any behavioural equivalence. Completeness follows from the strong decomposition characteristics of hhp-b on BPP. A formal proof of soundness and completeness is provided in [9].

Theorem 4 (soundness, completeness). $\vdash E = F$ *if and only if* $E \sim_{hhp} F$.

6 Conclusions

We have provided a polynomial-time procedure (working in time $\mathcal{O}(n^2 \log n)$) to compute hhp-b on BPP. Our algorithm takes a BPP definition in T-ENF as input. Transformation to T-ENF can easily be done in time quadratic wrt. the size of the input; the size of the definition may also grow by that factor during the transformation. Furthermore, we have proposed a sound and complete equational axiomatization of the equivalence. The crucial insight behind both of these results is that, modulo hhp-bisimilar choices, hhp-b is decomposable wrt. parallel composition and choice. Our results highlight that, modulo trivial choices, hhp-b fully reflects the structure of BPP expressions. One could argue that this is what one would intuitively expect of a truly-concurrent bisimulation equivalence. In particular, it does not imply that hhp-b is trivial on BPP: hhp-bisimilar choices may be hidden deeply within the process definition.

One could ask whether hhp-b also satisfies the unique decomposition property usually investigated in the interleaving setting: is each BPP process uniquely, up to hhp-b, represented as a parallel composition of primes? A process is prime if it cannot be expressed, up to hhp-b, as a non-trivial parallel composition. Indeed, from our results it is straightforward to derive that hhp-b does satisfy unique decomposition in this sense: Lemma 2 ensures that there is a one-to-one correspondence between prime factors wrt. hhp-b and factors that do not contain any hhp-bisimilar choices.

As mentioned in the introduction, unique decomposition with respect to distributed bisimilarity, and hence with respect to hp-b, has been established for BPP [3]. It has also been proved that cancellation (c.f. Lemma 1) does hold for distributed bisimilarity [3]. However, the following example of [3] shows that hp-b is *not* decomposable wrt. $||$ or $+$, in the sense of Section 3.

$$E = (a.\mathbf{0} + b.\mathbf{0})||a.\mathbf{0} + a.\mathbf{0}||a.\mathbf{0} \qquad\qquad F = (a.\mathbf{0} + b.\mathbf{0})||a.\mathbf{0}.$$

Both E and F have no hp-bisimilar choices, and $E \sim_{hp} F$. But $\texttt{summands}(E)$ and $\texttt{factors}(F)$ have two elements while $\texttt{summands}(F)$ and $\texttt{factors}(E)$ are singletons. In particular, the example illustrates that Lemma 2 fails for hp-b. An interesting question that remains open is whether, modulo hhp-bisimilar choices, hhp-b is decomposable with respect to prime decompositions of labelled asynchronous transition systems (c.f. [6]).

Our algorithm is a natural complement of the polynomial-time procedures for hp-b on BPP [16, 12]. However, in the case of hhp-b the good complexity is due to its very strong decomposition properties; the technique of [11] seems not to be applicable here. Both algorithms can be carried over to CPP, an extension of BPP that allows for synchronization between processes in CCS style but disallows a silent action τ to appear explicitly inside expressions. It is not clear whether a polynomial-time procedure exists for hhp-b or hp-b on BPP_τ, which extends CPP by allowing explicit τ-actions. Preliminary investigations give hope that polynomial-time complexity of hhp-b on BPP_τ can indeed be achieved;—this issue will be treated in detail in a full version of this paper.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Co., 1974.
2. M. Bednarczyk. Hereditary history preserving bisimulation or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences, Gdansk, 1991.
3. S. Christensen. *Decidability and Decomposition in process algebras*. PhD thesis, Dept. of Computer Science, University of Edinburgh, UK, 1993.
4. J. Esparza and A. Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *CAV'95*, volume 939 of *LNCS*, pages 353–366. Springer-Verlag, 1995.
5. S. Fröschle. Decidability of plain and hereditary history-preserving bisimulation for BPP. In *Proc. EXPRESS'99, volume 27 of ENTCS*, 1999.
6. S. Fröschle. Composition and decomposition in true-concurrency. In *Proc. FOSSACS'05*, LNSC. Springer-Verlag, to appear, 2005.
7. S. Fröschle. The decidability border of hereditary history preserving bisimilarity. *Information Processing Letters*, to appear, 2005.
8. S. Fröschle and T. Hildebrandt. On plain and hereditary history-preserving bisimulation. In *MFCS'99*, volume 1672 of *LNCS*, pages 354–365. Springer-Verlag, 1999.
9. S. Fröschle and S. Lasota. Decomposition and complexity of hereditary history preserving bisimulation on BPP. Technical Report 280, Institute of Informatics, Warsaw University, Poland, 2005.
10. 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.
11. P. Jančar. Bisimilarity of basic parallel processes is PSPACE-complete. In *Proc. LICS'03*, pages 218–227, 2003.
12. P. Jančar and Z. Sawa. On distributed bisimilarity over Basic Parallel Processes. In *Proc. AVIS2'05*, 2005.
13. L Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154:107–143, 1996.
14. A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127:164–185, 1996.
15. Marcin Jurdziński, Mogens Nielsen, and J. Srba. Undecidability of domino games and hhp-bisimilarity. *Inform. and Comput.*, 184:343–368, 2003.
16. S. Lasota. A polynomial-time algorithm for deciding true concurrency equivalences of Basic Parallel Processes. In *Proc. MFCS'03*, LNCS 2747, pages 521–530. Springer-Verlag, 2003.
17. R. Milner and F. Moller. Unique decomposition of processes. *TCS*, 107(2):357–363, 1993.
18. J. Srba. Strong bisimilarity and regularity of Basic Parallel Processes is PSPACE-hard. In *Proc. STACS'02, LNCS 2285*, 2002.
19. J. Srba. *Roadmap of Infinite Results*, volume 2: Formal Models and Semantics. World Scientific Publishing Co., 2004.
20. K. Sunesen and M. Nielsen. Behavioural equivalence for infinite systems—partially decidable! In *ICATPN'96*, volume 1091 of *LNCS*, pages 460–479. Springer-Verlag, 1996.