Strong Bisimilarity on Basic Parallel Processes is PSPACE-complete

Petr Jančar
Techn. Univ. Ostrava
Dept of Computer Science
17. listopadu 15, CZ-708 33 Ostrava, Czech Republic
Petr.Jancar@vsb.cz

Abstract

The paper shows an algorithm which, given a Basic Parallel Processes (BPP) system, constructs a set of linear mappings which characterize the (strong) bisimulation equivalence on the system. Though the number of the constructed mappings can be exponential, they can be generated in polynomial space; this shows that the problem of deciding bisimulation equivalence on BPP is in PSPACE. Combining with the PSPACE-hardness result by Srba, PSPACE-completeness is thus established.

1. Introduction

Bisimulation equivalence is a fundamental behavioural equivalence in concurrency theory, having also natural connections with (modal) logics used in computer science (see, e.g., [10]). In the areas related to (automated) verification, the questions concerning complexity of decision problems for bisimulation equivalence on various classes of systems are natural subjects of research. One specific topic of this research deals with some basic models of infinite state labelled transition systems (LTS); surveys of relevant results can be found, e.g., in [2] or [14].

We can recall that finite state LTSs can be viewed as nondeterministic finite automata (NFA). While language equivalence is well known to be PSPACE-complete for NFA, bisimulation equivalence (which is finer) is solvable by fast algorithms [11, 9].

Context-free grammars, in Greibach normal form, can be viewed as generators of a class of infinite state LTSs, in two natural ways. In both cases, finite sequences of nonterminals are viewed as states. One way then views such a sequence as a sequential composition of nonterminals: only *the leftmost nonterminal*, say X, can be rewritten according to a rule $X \xrightarrow{a} Y_1Y_2 \dots Y_n$ (i.e., X is replaced by $Y_1Y_2 \dots Y_n$ while action a is 'emitted'). This way yields the class of so called BPA-processes (which can also be read as 'basic sequential processes'). The other way views a sequence as a parallel composition: *any nonterminal* in a sequence can be rewritten. This way yields the class of 'basic parallel processes' (BPP). While language equivalence is well-known to be undecidable for context-free grammars, bisimulation equivalence is decidable: decidability for BPA was shown in [5] (which extended the previous result [1]), and decidability for BPP was shown in [4]. But while we know a (superexponential) elementary upper bound in the case of BPA [3], no primitive recursive bound has been derived for BPP. (Polynomial algorithms are known in the subcases of normed BPA [6] and normed BPP [7]; speaking in language theory terminology, *normed* means that each nonterminal can derive a terminal word.)

We can further note that bisimulation equivalence is decidable even for (the richer class of) 'pushdown processes'; these can be viewed as BPA-processes extended with finite control state units. The (abstract of the) first proof was published in [12]; it is a very involved proof, an extension of the famous DPDA-result [13]. A more manageable proof is given in [17] (which extends the previous paper [16]). For similarly 'state-extended' BPP, bisimulation equivalence is undecidable (see, e.g., [8]).

In this paper, we will show that the problem Bis-BPP (bisimilarity on BPP) belongs to PSPACE. We use a novel approach, which also enables to show that a semilinear (or Presburger arithmetic) representation of bisimulation equivalence on a BPP system can be constructed effectively. Combining with the PSPACE-hardness result by Srba [15], we thus establish PSPACE-completeness of Bis-BPP. In Section 6, we briefly discuss possible applications to other related problems. In particular, we mention the problem of *weak* bisimilarity on BPP whose decidability is still open.

Basic definitions are given in Section 2. Sections 3 and 4 explain the used approach on more general levels, and give thus a background for the main technical proof in Section 5.

2. Preliminaries

A *labelled transition system* (LTS) is a tuple

$$(S, A, \{\xrightarrow{a}\}_{a\in A})$$

where S is a (possibly infinite) set of *states*, A a (finite) set of *actions*, or (transition) *labels*, and $\xrightarrow{a} \subseteq S \times S$ for each $a \in A$. We use infix notation $r \xrightarrow{a} r'$, and write $r \longrightarrow r'$ when $r \xrightarrow{a} r'$ for some a. We also use the natural generalization $r \xrightarrow{w} r'$ for finite sequences w of actions ($w \in A^*$).

A relation R on S (of an LTS) is a *bisimulation* iff for each $(r_1, r'_1) \in R$ the following conditions hold:

$$\forall a, r_2 : r_1 \xrightarrow{a} r_2 \Rightarrow (\exists r'_2 : r'_1 \xrightarrow{a} r'_2 \wedge (r_2, r'_2) \in R),$$

$$\forall a, r'_2 : r'_1 \xrightarrow{a} r'_2 \Rightarrow (\exists r_2 : r_1 \xrightarrow{a} r_2 \wedge (r_2, r'_2) \in R).$$

Bisimulation equivalence (bisimilarity), on an assumed LTS, is the maximal bisimulation (i.e., union of all bisimulations); we denote it by $\sim$.

A *BPP system* can be (traditionally) viewed as a context-free grammar in Greibach normal form, with rules written

$$X \xrightarrow{a} Y_1 Y_2 \dots Y_n .$$

To each sequence α of variables (i.e., nonterminals), we define $\text{M-SET}(\alpha)$ as the multiset of variables apearing in α (i.e., the Parikh image of α). The associated LTS has multisets of variables (ranged over by M, M', ...) as states and

$$M \xrightarrow{a} M'$$

iff $M \ni X$ and $M' = (M \setminus \{X\}) \cup \text{M-SET}(Y_1 Y_2 \dots Y_n)$ for a rule $X \xrightarrow{a} Y_1 Y_2 \dots Y_n$.

We will prefer an equivalent setting which views a BPP system as a (special) labelled Petri net (BPP-net)

$$\Sigma = (P, Tr, \text{PRE}, F, \ell)$$

where $\ell : Tr \rightarrow A$ for some label set A. Here $P = \{ p_1, p_2, \dots, p_k \}$ is a set of *places*, $Tr = \{ t_1, t_2, \dots, t_m \}$ a set of *transitions*;

$$\text{PRE} : Tr \rightarrow P$$

attaches the (only) input place to each transition, while the function

$$F : (T \times P) \rightarrow \mathcal{N}$$

determines the (multiset of) output places of each transition. $\mathcal{N}$ denotes the set of nonnegative integers.

We attach an (infinite state) LTS to Σ, with markings $(M, M', \dots)$ as states, where a marking $M : P \rightarrow \mathcal{N}$ provides the number of *tokens* to each place. We often view M as a vector

$$M = (x_1, x_2, \dots, x_k),$$

where $x_i \in \mathcal{N}$. We define $M \xrightarrow{a} M'$ iff $M \xrightarrow{t} M'$ with $\ell(t) = a$; here $M \xrightarrow{t} M'$ iff $M(\text{PRE}(t)) \geq 1$ and

$$M'(p) = \begin{cases} M(p) - 1 + F(t,p) & \dots \quad \text{if } p = \text{PRE}(t) \\ M(p) + F(t,p) & \dots \quad \text{otherwise} \end{cases}$$

Notation $\text{PRE}(t)$ is generalized to $\text{PRE}(T)$ for sets $T \subseteq Tr$: $\text{PRE}(T) = \{\text{PRE}(t) \mid t \in T\}$.

Further conventions are:

$|A|$ denotes the cardinality of a set A.

$A \subset B$ denotes that A is a *proper* subset of B; $A \subseteq B$ allows equality.

$f|_D$ denotes restriction of mapping f to (sub)domain D.

0 denotes the function which gives 0 for each element of a domain clear from the context.

Partition $\mathcal{U}$ of a set U is a set $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$ of disjoint *clasess* whose union is U. By the *intersection of two partitions* $\mathcal{U}_1$, $\mathcal{U}_2$ we mean the partition in which two elements of U are in the same class iff they are in the same class for both $\mathcal{U}_1$ and $\mathcal{U}_2$.

$\mathcal{Z}$ denotes the set of integers, and $\mathcal{N}_\omega = \mathcal{N} \cup \{\omega\}$. Here ω is just a symbol for infinite amount; we stipulate $\forall z \in \mathcal{Z} : z < \omega, \forall z \in \mathcal{Z} : \omega + z = z + \omega = \omega, \omega + \omega = \omega - \omega = -\omega + \omega = \omega, \omega \cdot 0 = 0 \cdot \omega = 0$, and $\forall n \geq 1 : \omega \cdot n = n \cdot \omega = \omega$.

3. General strategy

Here we sketch one of possible general strategies for (describing and) deciding bisimilarity. It is the strategy used in the later presented algorithm for BPP.

Let us assume a fixed (general) LTS $(S, A, \{\xrightarrow{a}\}_{a\in A})$. One approach to (describe and) decide the bisimulation equivalence $\sim$ on S tries to construct effectively a sequence of (descriptions of) decidable equivalences $\equiv_0$, $\equiv_1$, $\equiv_2$, $\dots$ where each $\equiv_i$ is guaranteed to subsume bisimilarity ($\equiv_i \supseteq \sim$) and is refined by $\equiv_{i+1}$ ($\equiv_i \supseteq \equiv_{i+1}$). If this process is guaranteed to reach and recognize a 'fixpoint' $\equiv_i = \equiv_{i+1}$ and this equality is guaranteed to imply $\equiv_i = \sim$ then the goal is satisfied.

A possibility to define an equivalence on S is to use a (suitable) mapping $\mathcal{C} : S \rightarrow \mathcal{D}$, which attaches a certain 'characteristic' from domain $\mathcal{D}$ to each state. Mapping $\mathcal{C}$ induces equivalence $\equiv_\mathcal{C}$ where

$$r \equiv_\mathcal{C} r' \Leftrightarrow \mathcal{C}(r) = \mathcal{C}(r')$$

A natural tool for refining equivalences is combining different mappings. For

$$\mathcal{C}_1 : S \rightarrow \mathcal{D}_1, \ \mathcal{C}_2 : S \rightarrow \mathcal{D}_2$$

we define the *pair of mappings* $\mathcal{C}_1, \mathcal{C}_2$ as the mapping

$$(\mathcal{C}_1, \mathcal{C}_2) : S \rightarrow \mathcal{D}_1 \times \mathcal{D}_2$$

where

$$(\mathcal{C}_1, \mathcal{C}_2)(r) = (\,\mathcal{C}_1(r), \mathcal{C}_2(r)\,)$$

Hence $\equiv_{(\mathcal{C}_1,\mathcal{C}_2)} = \;(\equiv_{\mathcal{C}_1} \cap \equiv_{\mathcal{C}_2})$.
The definitions naturally generalize to yield *tuples of mappings*

$$(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n)\,.$$

We say that a mapping $\mathcal{C}$ is *bis-necessary* iff $\sim \subseteq \equiv_{\mathcal{C}}$. (Equality of the $\mathcal{C}$-value is a necessary condition for bisimilarity of two states.)
We observe:

Claim 3.1 *If $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ are bis-necessary then also $(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n)$ is bis-necessary.*

In fact, the strategy of our later algorithm for BPP will start with some bis-necessary mapping $\mathcal{C}_0$ and define

$$\mathcal{C}_{i+1} = (\,\mathcal{C}_i, der(\mathcal{C}_i)\,)$$

for a function der which, given a bis-necessary mapping, 'derives' another bis-necessary mapping.

We also consider predicates on S. The notion of a *bis-necessary predicate* $\mathcal{P}$ is induced by viewing $\mathcal{P}$ as a mapping

$$\mathcal{P} : S \rightarrow \{0,1\}\,.$$

Note that if $\mathcal{P}$ is bis-necessary then $\neg\mathcal{P}$ is bis-necessary.

One of useful predicates is $\mathrm{DISABLED}_a$ ($a \in A$), which is defined as follows:

$$\mathrm{DISABLED}_a(r) \Leftrightarrow \neg\exists r' : r \xrightarrow{a} r'$$

We denote $\neg\mathrm{DISABLED}_a$ by $\mathrm{ENABLED}_a$ and we observe that both $\mathrm{DISABLED}_a$ and $\mathrm{ENABLED}_a$ are bis-necessary.

In further mappings $\mathcal{C} : S \rightarrow \mathcal{D}$, we assume a binary operation $+$ on $\mathcal{D}$ such that

$\mathcal{C}$ is *safe*, i.e.:

for any $r \longrightarrow r'$ there is (at least one) $\delta \in \mathcal{D}$ such that $\mathcal{C}(r) + \delta = \mathcal{C}(r')$.

Remark. If we do not mention the respective operation $+$ explicitly, e.g. at some mappings derived from $\mathcal{C}$, it either does not matter or can be naturally deduced from the context; this might also require a natural extension of the (co)domain.

We now define the following predicate $\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)$, for a transition label a and a 'change' $\delta \in \mathcal{D}$:

$$\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)(r) \Leftrightarrow \neg\exists r' : r \xrightarrow{a} r' \wedge \mathcal{C}(r) + \delta = \mathcal{C}(r')$$

So $\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)(r)$ holds iff the 'label-change pair' (a,δ) is disabled in r. Note that $\mathrm{DISABLED}_a(r)$ implies $\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)(r)$. (In fact, we can view $\mathrm{DISABLED}_a$ as $\mathrm{DISABLED}_{\mathcal{C}}(\,a, ()\,)$ for $\mathcal{C}$ being the 0-tuple, i.e., for $\mathcal{C}$ being a constant mapping.) We also denote

$$\neg\mathrm{DISABLED}_{\mathcal{C}}(a,\delta) \;\text{ by }\; \mathrm{ENABLED}_{\mathcal{C}}(a,\delta)\,.$$

We observe:

Claim 3.2 *If $\mathcal{C}$ is a bis-necessary mapping then $\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)$ is a bis-necessary predicate.*

Proof: Consider a bis-necessary mapping $\mathcal{C}$. For sake of contradiction, let us assume $r_1 \sim r_1'$, $\mathrm{ENABLED}_{\mathcal{C}}(a,\delta)(r_1)$ and $\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)(r_1')$. Then $\mathcal{C}(r_1) = \mathcal{C}(r_1')$ and there is a move $r_1 \xrightarrow{a} r_2$ where $\mathcal{C}(r_1) + \delta = \mathcal{C}(r_2)$. There must be a move $r_1' \xrightarrow{a} r_2'$ where $r_2 \sim r_2'$; but necessarily $\mathcal{C}(r_2) \neq \mathcal{C}(r_2')$ – a contradiction. □

For $\mathcal{C} : S \rightarrow \mathcal{D}$ we introduce mapping $\text{D-LC}_{\mathcal{C}}$; the value $\text{D-LC}_{\mathcal{C}}(r)$ shows which label-change pairs are disabled (and which are enabled) in the state r:

$$\text{D-LC}_{\mathcal{C}} : S \rightarrow (\{0,1\})^{A \times \mathcal{D}}$$

where

$$\text{D-LC}_{\mathcal{C}}(r) = \lambda(a,\delta)\,.\,\mathrm{DISABLED}_{\mathcal{C}}(a,\delta)(r)$$

We note that if $\mathcal{C}$ is bis-necessary then $\text{D-LC}_{\mathcal{C}}$ is bis-necessary as well.

We also use a further technical notion. We say that $\mathcal{C}$ is *lc-closed* iff

$$\mathcal{C}(r) = \mathcal{C}(r') \text{ implies } \text{D-LC}_{\mathcal{C}}(r) = \text{D-LC}_{\mathcal{C}}(r')$$

(In such a case, the value $\mathcal{C}(r)$ determines $\text{D-LC}_{\mathcal{C}}(r)$.) We observe

Claim 3.3 *If $\mathcal{C}$ is (safe and) lc-closed then $\equiv_{\mathcal{C}}$ is a bisimulation. (So $\equiv_{\mathcal{C}} \subseteq \sim$.)*

Proof: Consider a (safe and) lc-closed mapping $\mathcal{C}$. Assume $\mathcal{C}(r_1) = \mathcal{C}(r_1')$ and $r_1 \xrightarrow{a} r_2$. Since $\mathcal{C}$ is safe, there is δ such that $\mathcal{C}(r_1) + \delta = \mathcal{C}(r_2)$. So we have $\mathrm{ENABLED}_{\mathcal{C}}(a,\delta)(r_1)$ and since $\text{D-LC}_{\mathcal{C}}(r_1) = \text{D-LC}_{\mathcal{C}}(r_1')$ we also have $\mathrm{ENABLED}_{\mathcal{C}}(a,\delta)(r_1')$. Then there is r_2' such that $r_1' \xrightarrow{a} r_2'$ and $\mathcal{C}(r_1') + \delta = \mathcal{C}(r_2')$; this means $\mathcal{C}(r_2) = \mathcal{C}(r_2')$. □

So we can derive

Proposition 3.4 *If $\mathcal{C} : S \rightarrow \mathcal{D}$ is bis-necessary and lc-closed then $\equiv_{\mathcal{C}}$ coincides with the bisimulation equivalence $\sim$ on S.*

We still look more closely at a refining 'tool' we will use. Given a predicate $\mathcal{P}$ on S, we define the mapping ('distance to $\mathcal{P}$')

$$\mathrm{DIST}(\mathcal{P}) : S \rightarrow \mathcal{N}_\omega$$

where $\text{DIST}(\mathcal{P})(r)$ is the length of a shortest w such that $r \xrightarrow{w} r'$ with $\mathcal{P}(r')$; it is ω if there is no such w.

Note that $\text{DIST}(\mathcal{P})$ is a safe mapping if we deem the (co)domain $\mathcal{N}_\omega$ extended to $\mathcal{Z} \cup \{\omega\}$ (in fact, $\{-1\} \cup \mathcal{N}_\omega$ would be sufficient):

for any $r \longrightarrow r'$ there is δ such that

$$\text{DIST}(\mathcal{P})(r) + \delta = \text{DIST}(\mathcal{P})(r')\,;$$

(in other words, if $\text{DIST}(\mathcal{P})(r) = \omega$ then $\text{DIST}(\mathcal{P})(r') = \omega$).

Claim 3.5 *If $\mathcal{P}$ is bis-necessary then $\text{DIST}(\mathcal{P})$ is bis-necessary.*

Proof: Assume $r_1 \sim r'_1$ and $\text{DIST}(\mathcal{P})(r_1) < \text{DIST}(\mathcal{P})(r'_1)$. Then we have a shortest $w \in A^*$ such that $r_1 \xrightarrow{w} r_2$ with $\mathcal{P}(r_2)$. Since $r_1 \sim r'_1$, there is r'_2 such that $r'_1 \xrightarrow{w} r'_2$ and $r_2 \sim r'_2$. But we necessarily have $\neg\mathcal{P}(r'_2)$, which means that $\mathcal{P}$ is not bis-necessary. □

We can now introduce the mapping which provides the 'distances to disablings of label-change pairs':

$$\text{DD-LC}_{\mathcal{C}} : S \rightarrow (\mathcal{N}_\omega)^{A\times\mathcal{D}}$$

where

$$\text{DD-LC}_{\mathcal{C}}(r) = \lambda(a,\delta)\,.\,\text{DIST}(\,\text{DISABLED}_{\mathcal{C}}(a,\delta)\,)\,(r)$$

Note that $\text{DD-LC}_{\mathcal{C}}$ is safe (wrt the naturally defined $+$ on the extended (co)domain). Also note that $\text{DD-LC}_{\mathcal{C}}(r) = \text{DD-LC}_{\mathcal{C}}(r')$ implies $\text{D-LC}_{\mathcal{C}}(r) = \text{D-LC}_{\mathcal{C}}(r')$. We also observe:

Claim 3.6 *If $\mathcal{C}$ is bis-necessary then $(\mathcal{C}, \text{DD-LC}_{\mathcal{C}})$ is bis-necessary.*

Our strategy, used later for BPP, can be viewed as starting with $\mathcal{C}_0 = ()$ (the 0-tuple; hence $\equiv_{\mathcal{C}_0} = S \times S$), and defining

$$\mathcal{C}_{i+1} = (\mathcal{C}_i, \text{DD-LC}_{\mathcal{C}_i})$$

We can note that $r \equiv_{\mathcal{C}_1} r'$ means that r, r' have the same distance to disabling a, for every label a.

In our particular case of BPP, the described process will finish with some lc-closed $\mathcal{C}_i$; since $\mathcal{C}_i$ is also bis-necessary, we will have $\equiv_{\mathcal{C}_i} = \sim$.

4. General strategy applied to BPP

We now assume a fixed BPP system $\Sigma = (P, Tr, \text{PRE}, F, \ell)$ with $\ell : Tr \rightarrow A$; we also assume $P = \{\, p_1, p_2, \ldots, p_k \,\}$.

We consider the respective (infinite-state) LTS_Σ with the state set

$$\mathcal{S}_\Sigma = \{\, (x_1, x_2, \ldots, x_k) \mid x_1 \geq 0, x_2 \geq 0, \ldots, x_k \geq 0 \,\}$$

and with the respective relations $\xrightarrow{a}$ on $\mathcal{S}_\Sigma$, for labels $a \in A$. The state set $\mathcal{S}_\Sigma$ will be ranged over by variables (markings) $M, M', \ldots$.

We shall apply the previously described general strategy, which will lead us to Theorem 4.7. This shows that a 'semilinear description' (which can be viewed as a Presburger arithmetic formula) of the bisimulation equivalence $\sim$ on $\mathcal{S}_\Sigma$ can be effectively constructed.

So we start with $\mathcal{C}_0 = ()$ ($M \equiv_{\mathcal{C}_0} M'$ for all M, M'), and explore how $\mathcal{C}_1 = \text{DD-LC}_{\mathcal{C}_0}$ looks like.

Let us put

$$T_a = \{t \in Tr \mid \ell(t) = a\}$$

and define L-PART as the partition of the transition set Tr according to labels:

$$\text{L-PART} = \{\, T_a \,\}_{a\in A}$$

We observe that a is disabled in M iff $M|_{\text{PRE}(T_a)} = \mathbf{0}$; and $\text{DIST}(\text{DISABLED}_a)\,(M)$ is the length of a shortest transition sequence from M leading to 'unmarking' $\text{PRE}(T_a)$. These considerations suggest to introduce, for a set of places $Q \subseteq P$, the predicate

$$\text{ZERO}_Q(M) \Leftrightarrow M|_Q = \mathbf{0}$$

and the function

$$\text{NORM}(Q) = \text{DIST}(\text{ZERO}_Q)$$

So $\text{DD-LC}_{\mathcal{C}_0}$ can be naturally viewed as the tuple of all functions $\text{NORM}(\text{PRE}(T))$ where T is a class of L-PART.

The next subsection explores functions $\text{NORM}(Q)$ in more detail. The main result is then contained in the following subsection.

Norms of sets of places and linear functions

We recall a useful notion from Petri net theory:

A set of places $R \subseteq P$ is a *trap* iff

$$\forall t : \text{PRE}(t) \in R \Rightarrow (\exists p \in R : F(t,p) \geq 1)$$

Note that a 'marked' trap, i.e., a trap with at least one token, can not get unmarked. Hence for a trap R,

$$\text{NORM}(R)(M) \text{ is either } 0 \text{ or } \omega\,.$$

We also note that $\emptyset$ is a trap ($\text{NORM}(\emptyset)(M) = 0$) and that union of traps is a trap; so each set of places Q contains the maximal trap, denoted $\text{MAXTRAP}(Q)$. And we observe

$$\text{NORM}(Q)(M) = \omega \text{ iff } M|_{\text{MAXTRAP}(Q)} \neq \mathbf{0}$$

(i.e., iff $\text{MAXTRAP}(Q)$ is marked in M).

By a *linear function* we mean a function $L : \mathcal{S}_\Sigma \to \mathcal{N}_\omega$ given by

$$L(x_1, x_2, \ldots, x_k) = c_1x_1 + c_2x_2 + \ldots + c_kx_k \qquad (1)$$

with coefficients c_j from $\mathcal{N}_\omega$. We define

$$\Omega\text{-CARR}(L) = \{ p_i \mid c_i = \omega \}$$

Note that $L(M)$ is finite iff $M|_{\Omega\text{-CARR}(L)} = \mathbf{0}$ (i.e., if $\text{ZERO}_{\Omega\text{-CARR}(L)}(M)$).

A (linear) *function* L is *safe* (wrt LTS_Σ) iff

$$L(M_1) = \omega \text{ and } M_1 \longrightarrow M_2 \text{ implies } L(M_2) = \omega.$$

We note that L is safe iff $\Omega\text{-CARR}(L)$ is a trap.

Claim 4.1 $\text{NORM}(Q)$ *is a safe linear function.*

Proof: We recall the assumption $P = \{ p_1, p_2, \ldots, p_k \}$. Informally we can say that $\text{NORM}(Q)$ can be given in form (1) where

- for $p_i \in Q$, the coefficient c_i is the length of a shortest transition sequence which removes a token from p_i while leaving the values of other places in Q unchanged; $c_i = \omega$ if there is no such sequence
- for $p_i \notin Q$, $c_i = 0$

More formally, we can sketch the following algorithm computing the coefficients c_i :

for each $p \in P \setminus Q$ do $c_p := 0$;
for each $p \in Q$ do $c_p := \omega$;
$Q' := Q$;
while $Q' \neq \emptyset$ do

  for each $p \in Q'$ do

    for each t s.t. $\text{PRE}(t) = p$ compute

$$d_t = \sum_{i=1}^{k} c_{p_i} \cdot F(t, p_i) \, ;$$

    let d_p be the minimum from $\{\omega\} \cup \{d_t \mid \text{PRE}(t) = p\}$;

  $d := min\{d_p \mid p \in Q'\}$;
  for each $p \in Q'$ s.t. $d_p = d$ do

    $c_p := 1 + d$; $Q' := Q' \setminus \{p\}$

It is easily verifiable that $\text{NORM}(Q)$ can be given in form (1) where $c_i = c_{p_i}$ as computed by the above algorithm.
($c_p = \omega$ precisely for all $p \in \text{MAXTRAP}(Q)$.)

Safeness of $\text{NORM}(Q)$ is obvious. □

For future complexity considerations, we note the following claims which can be easily derived from the above algorithm.

Claim 4.2 *The finite coefficients of the linear function* $\text{NORM}(Q)$ *are bounded by* $(kW)^k$ *where* k *is the number of places and* $W = max\{ F(t,p) \mid t \in T, p \in P \}$.

Proof: Let us consider all moments when the algorithm attaches the final value c_p to a place p. Let $q_1, q_2, \ldots, q_k$ be the ordering of places corresponding to the ordering of these moments in time. Then obviously $c_{q_1} \leq c_{q_2} \leq \ldots \leq c_{q_k}$.

By induction we can easily show that for a finite $c_{q_{j+1}}$ we have $c_{q_{j+1}} \leq (kW)^j$. □

The next claim assumes a standard presentation of a BPP system Σ where the numbers $F(t,p)$ can be given in binary.

Claim 4.3 *(Given a BPP system* Σ *and a set* Q *of places,) the coefficients of* $\text{NORM}(Q)$ *can be computed in polynomial time and presented (in binary) in polynomial space.*

We can also note that, given a linear function L and M (in binary), the value $L(M)$ can be computed in polynomial time.

Effective description of bisimilarity on BPP

Now we proceed with our strategy. After clarifying $\mathcal{C}_0, \mathcal{C}_1$, we want to explore $\text{DD-LC}_{\mathcal{C}_1}$, etc. Speaking slightly more generally, we shall be interested in exploring how to represent $\text{DD-LC}_{\mathcal{F}}$ where

$$\mathcal{F} = (L_1, L_2, \ldots, L_h)$$

is a tuple of safe linear functions.

Note that $\mathcal{F}(M)$ is *finite*, meaning finite in each component, iff $M|_{\Omega\text{-CARR}(\mathcal{F})} = \mathbf{0}$, where

$$\Omega\text{-CARR}(\mathcal{F}) = \bigcup_{1 \leq i \leq h} \Omega\text{-CARR}(L_i)$$

Remark. In our case, $L_i = \text{NORM}(Q_i)$ for some $Q_i \subseteq P$ ($i = 1, 2, \ldots, h$). Then $\Omega\text{-CARR}(\mathcal{F}) = R_1 \cup R_2 \cup \ldots R_h$ where $R_i = \text{MAXTRAP}(Q_i)$.

An attempt to represent $\text{DD-LC}_{\mathcal{F}}$ leads to defining the following 'change' δ^t_L, which is caused by transition $t \in Tr$ on a safe function L (in form (1)):

$$\delta^t_L = -c_i + \sum_{1 \leq j \leq k} c_j \cdot F(t, p_j)$$

where $p_i = \text{PRE}(t)$. (The definition is sound due to safeness of L.) We note

Claim 4.4 *Suppose a safe L, and $M_1 \xrightarrow{t} M_2$. Then $L(M_1) + \delta^t_L = L(M_2)$. Moreover, if $L(M_1)$ is finite and $\delta \neq \delta^t_L$ then $L(M_1) + \delta \neq L(M_2)$.*

(We can note that if $L(M_1) = \omega$ then $L(M_1) + \delta = L(M_2)$ for any δ.)

This leads us to defining the partition

$$\text{LC-PART}_{\mathcal{F}}$$

of the set Tr, according to 'label-change pairs', where a change is a vector from $(\mathcal{Z} \cup \{\omega\})^h$ (in fact, from $(\{-1\} \cup \mathcal{N}_\omega)^h$ for norms):

t, t' are in the same class of $\text{LC-PART}_{\mathcal{F}}$ iff $\ell(t) = \ell(t')$ and $\delta^t_{L_i} = \delta^{t'}_{L_i}$ for $i = 1, 2, \ldots, h$.

Supposing $\mathcal{F}$ bis-necessary, we might be tempted to think that, for a class T of $\text{LC-PART}_{\mathcal{F}}$, the predicate $\text{ZERO}_{\text{PRE}(T)}$ (seemingly disabling a particular label-change pair) is bis-necessary. (This would imply that $\text{NORM}(\text{PRE}(T))$ is also bis-necessary.) But it is not the case because of the possibility of infinite components in $\mathcal{F}(M)$. Nevertheless we can safely claim a relativized version (Claim 4.5), using other useful notions and notation:

For a trap R and a partition $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ of the set Tr, we define the mapping

$$\text{NORMS}_{R,\mathcal{T}} : \mathcal{S}_\Sigma \to (\mathcal{N}_\omega)^n$$

as the tuple of functions

$$\text{NORMS}_{R,\mathcal{T}} = (L_1, L_2, \ldots, L_n)$$

where

$$L_i = \text{NORM}(\, R \cup \text{PRE}(T_i)\,)\,.$$

Note that if $\neg\text{ZERO}_R(M)$ then $\text{NORMS}_{R,\mathcal{T}}(M)$ has ω in each component. We can also note that

$$\text{DD-LC}_{\mathcal{C}_0} = \text{NORMS}_{\emptyset,\text{L-PART}}\,.$$

We say that a *trap* R is *important* if ZERO_R is bis-necessary; i.e., $M \sim M'$ implies $M|_R = \mathbf{0} \iff M'|_R = \mathbf{0}$.

Note that if $\mathcal{F}$ is bis-necessary then $\Omega\text{-CARR}(\mathcal{F})$ is an important trap.

Claim 4.5 *If $\mathcal{F}$ is bis-necessary and $R \supseteq \Omega\text{-CARR}(\mathcal{F})$ is an important trap then $\text{NORMS}_{R,\text{LC-PART}_{\mathcal{F}}}$ is bis-necessary.*

Proof: (Assuming the assumption of the Claim,) it is sufficient to prove that for each class T of $\text{LC-PART}_{\mathcal{F}}$ we have that $\text{ZERO}_{(R\cup\text{PRE}(T))}$ is bis-necessary.

For the sake of contradiction, assume $M_1 \sim M'_1$, $\neg\text{ZERO}_{(R\cup\text{PRE}(T))}(M_1)$ and $\text{ZERO}_{(R\cup\text{PRE}(T))}(M'_1)$.

Since R is important, and $\text{ZERO}_R(M'_1)$, we also have $\text{ZERO}_R(M_1)$. This means that

$\mathcal{F}(M_1) = \mathcal{F}(M'_1)$ is finite.

Now, since necessarily $\neg\text{ZERO}_{\text{PRE}(T)}(M_1)$, we have $t \in T$ such that $M_1 \xrightarrow{t} M_2$.

There must be t' such that $M'_1 \xrightarrow{t'} M'_2$ where $\ell(t) = \ell(t')$ and $M_2 \sim M'_2$. Since $\text{ZERO}_{\text{PRE}(T)}(M'_1)$, transitions t, t' are necessarily from different classes of $\text{LC-PART}_{\mathcal{F}}$ (so $\delta^t_L \neq \delta^{t'}_L$ for some component L of the tuple $\mathcal{F}$).

But then $\mathcal{F}(M_2) \neq \mathcal{F}(M'_2)$ (recall Claim 4.4), which is a contradiction with $M_2 \sim M'_2$. □

Remark. To achieve the desired result (Theorem 4.7), it is not necessary to argue that the following construction $\mathcal{C}_{i+1}$ really implements the strategy $\mathcal{C}_{i+1} = (\mathcal{C}_i, \text{DD-LC}_{\mathcal{C}_i})$. So we avoid the technicalities which would be related with claiming and proving this.

Having $\mathcal{C}_i = (L_1, L_2, \ldots, L_n)$ as a bis-necessary tuple of norms (i.e., each L_i is $\text{NORM}(Q)$ for some $Q \subseteq P$), we define

$$\mathcal{C}_{i+1}$$

as the following enlarging of $\mathcal{C}_i$:

for each subtuple $\mathcal{F} = (L_{i_1}, L_{i_2}, \ldots, L_{i_h})$ of $\mathcal{C}_i$ (there are 2^n such subtuples) we add (the components of)

$$\text{NORMS}_{R,\text{LC-PART}_{\mathcal{F}}}$$

where $R = \Omega\text{-CARR}(\mathcal{F})$. (Note that R is an important trap.) In fact, we assume the natural optimalization: if $\text{NORM}(Q)$ (for some $Q \subseteq P$) would thus appear more than once as a component of $\mathcal{C}_{i+1}$, we leave it only once there by removing the excessive occurrences.

The mapping $\mathcal{C}_{i+1}$ is certainly bis-necessary and has at most 2^k components ($k = |P|$). The refining process (constructing $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \ldots$) thus surely finishes, with a certain

$$\mathcal{C}_\Sigma = \mathcal{C}_i = \mathcal{C}_{i+1} \text{ (for some } i < 2^k).$$

A crucial fact is:

Claim 4.6 *Relation $\equiv_{\mathcal{C}_\Sigma}$ is a bisimulation ($\mathcal{C}_\Sigma$ is lc-closed). ($\mathcal{C}_\Sigma$ is also bis-necessary, hence $\equiv_{\mathcal{C}_\Sigma} = \sim$.)*

Proof: Suppose $\mathcal{C}_\Sigma(M_1) = \mathcal{C}_\Sigma(M'_1)$, $M_1 \xrightarrow{t} M_2$ (and $\mathcal{C}_\Sigma(M_1) + \delta = \mathcal{C}_\Sigma(M_2)$).

Denote $\mathcal{F}$ the maximal subtuple of $\mathcal{C}_\Sigma$ for which $\mathcal{F}(M_1) = \mathcal{F}(M_1')$ is finite; denote $R = \Omega\text{-CARR}(\mathcal{F})$. Let T be the class of partition $\text{LC-PART}_{\mathcal{F}}$ which contains t.

We have $\text{ZERO}_R(M_1)$ and $\neg\text{ZERO}_{(R\cup\text{PRE}(T))}(M_1)$. We also have $\text{ZERO}_R(M_1')$, and $\neg\text{ZERO}_{(R\cup\text{PRE}(T))}(M_1')$ (since $\text{NORMS}_{R,\text{LC-PART}_{\mathcal{F}}}(M_1) = \text{NORMS}_{R,\text{LC-PART}_{\mathcal{F}}}(M_1')$).

So there is $t' \in T$ such that $M_1' \xrightarrow{t'} M_2'$; since both t, t' are in T, necessarily $\ell(t) = \ell(t')$ and $\mathcal{C}_\Sigma(M_2') = \mathcal{C}_\Sigma(M_2)$ (so $\mathcal{C}_\Sigma(M_1') + \delta = \mathcal{C}_\Sigma(M_2')$). □

Thus we have shown that we can effectively construct a semilinear (i.e., Presburger arithmetic) description of the bisimulation equivalence in the case of BPP:

Theorem 4.7 *There is an algorithm which, given a BPP system Σ, constructs a tuple $\mathcal{C}_\Sigma$ of linear functions (in fact, norms for certain sets of places) such that*

$$\forall M, M' \in \mathcal{S}_\Sigma : M \sim M' \iff \mathcal{C}_\Sigma(M) = \mathcal{C}_\Sigma(M')$$

As we have noted, $\mathcal{C}_\Sigma$ has at most 2^k components, k being the number of places.

Remark. The author conjectures that in the worst case the number of functions in such $\mathcal{C}_\Sigma$ is exponential. But this is not pursued here.

5. Polynomial space algorithm

We consider complexity of the following problem:

> Problem Bis-BPP
> *Instance*: a BPP system Σ, states $M, M' \in \mathcal{S}_\Sigma$
> *Question*: Is $M \sim M'$?

As mentioned in the Introduction, [4] shows decidability of Bis-BPP (without any primitive recursive upper bound) and [15] shows PSPACE-hardness (also if markings M, M', and numbers $F(t,p)$ are given in unary).

The previous section has already shown that Bis-BPP is solvable in exponential space. But looking in more detail at the described strategy will allow us to derive that the problem Bis-BPP belongs to PSPACE (even with presenting numbers in binary). The proof presented here is rather technical but it should be fairly understandable due to the intuition gained from the previous sections.

We again fix a BPP system $\Sigma = (P, Tr, \text{PRE}, F, \ell)$ with $\ell : Tr \to A$ and we start by defining certain refinings of partitions of Tr ; $\text{REFINE}(\mathcal{T}, R_1, R_2)$ defined below will be a crucial 'procedure' in our algorithm.

Let us recall (the change) δ_L^t, for a safe linear function L, defined before Claim 4.4. We now relativize the change wrt a trap R ; we put

$$\delta_L^{t,R} = \begin{cases} \delta_L^t & \dots \text{ if } \Omega\text{-CARR}(L) \subseteq R \\ \omega & \dots \text{ if } \Omega\text{-CARR}(L) \not\subseteq R \end{cases}$$

This can be naturally generalized to (vectors) $\delta_{\mathcal{F}}^{t,R}$ for tuples $\mathcal{F}$ (of safe linear functions). By

$$\text{PART}_{\mathcal{F}}^{R}$$

we denote the partition of the set Tr according to these relativized changes:

> t, t' are in the same class of $\text{PART}_{\mathcal{F}}^{R}$
> iff $\delta_{\mathcal{F}}^{t,R} = \delta_{\mathcal{F}}^{t',R}$.

We can note that $\text{PART}_{\mathcal{F}}^{R'}$ refines $\text{PART}_{\mathcal{F}}^{R}$ when $R' \supseteq R$.

The above definition of $\text{PART}_{\mathcal{F}}^{R}$ has been chosen so that we can guarantee:

Claim 5.1 *If t, t' are in* different classes *of* $\text{PART}_{\mathcal{F}}^{R}$
and $\mathcal{F}(M_1) = \mathcal{F}(M_1')$ and $M_1|_R = M_1'|_R = \mathbf{0}$
and $M_1 \xrightarrow{t} M_2$, $M_1' \xrightarrow{t'} M_2'$
then $\mathcal{F}(M_2) \neq \mathcal{F}(M_2')$.

Proof: Since t, t' are in different classes of $\text{PART}_{\mathcal{F}}^{R}$, there is a component L of $\mathcal{F}$ such that $\delta_L^{t,R} \neq \delta_L^{t',R}$, which also implies $\Omega\text{-CARR}(L) \subseteq R$; hence $\delta_L^{t,R} = \delta_L^t$, $\delta_L^{t',R} = \delta_L^{t'}$ and $L(M_1) = L(M_1')$ is finite. Therefore $L(M_2) = L(M_1) + \delta_L^t \neq L(M_1') + \delta_L^{t'} = L(M_2')$. □

Claim 5.2 *If t, t' are in* the same class *of* $\text{PART}_{\mathcal{F}}^{R}$
and $\mathcal{F}(M_1) = \mathcal{F}(M_1')$ and $L(M_1) = L(M_1') = \omega$ for each L in $\mathcal{F}$ such that $\Omega\text{-CARR}(L) \not\subseteq R$
and $M_1 \xrightarrow{t} M_2$, $M_1' \xrightarrow{t'} M_2'$
then $\mathcal{F}(M_2) = \mathcal{F}(M_2')$.

Proof: For each component L of $\mathcal{F}$ such that $L(M_1) = L(M_1')$ is finite we have $\Omega\text{-CARR}(L) \subseteq R$, and so $\delta_L^t = \delta_L^{t'}$; hence $L(M_2) = L(M_2')$. (If $L(M_1) = L(M_1') = \omega$ then $L(M_2) = L(M_2') = \omega$ due to safeness.) □

Given a partition $\mathcal{T}$ of the set Tr and traps $R_1 \subseteq R_2$, we define the partition

$$\text{REFINE}(\mathcal{T}, R_1, R_2)$$

as the intersection of $\mathcal{T}$ with $\text{PART}_{\mathcal{F}}^{R_2}$ where $\mathcal{F} = \text{NORMS}_{R_1,\mathcal{T}}$.

(I.e., t, t' belong to the same class of $\text{REFINE}(\mathcal{T}, R_1, R_2)$ iff they belong to the same class of $\mathcal{T}$ and $\delta_L^{t,R_2} = \delta_L^{t',R_2}$ for each $L = \text{NORM}(R_1 \cup \text{PRE}(T))$ where T is a class of $\mathcal{T}$.)

A particular case is $R_1 = R_2$. We also note that if $R_2 \subseteq R_2'$ then the partition $\text{REFINE}(\mathcal{T}, R_1, R_2')$ is a refinement of $\text{REFINE}(\mathcal{T}, R_1, R_2)$.

Recalling Claim 4.3, we can easily establish the following claim concerning complexity:

Claim 5.3 *There is an algorithm and a polynomial pol such that, given a BPP system Σ, sets R_1, R_2 and a partition $\mathcal{T}$, the algorithm computes $\text{REFINE}(\mathcal{T}, R_1, R_2)$ in time bounded by $pol(size(\Sigma))$.*

Our (general) algorithm will compute certain (d-important) traps, and certain partition $\mathcal{T}_R$ for each such trap R.

Remark. We use the word *d-important* ('defined as important') since we do not need to show that the notion coincides with *important* from the previous section.

To define $\mathcal{T}_R$, we will first define certain $\mathcal{T}_R^0$ and put

$$\mathcal{T}_R^{i+1} = \text{REFINE}(\mathcal{T}_R^i, R, R)\,.$$

It is clear that there is a fixpoint $\mathcal{T}_R^i = \mathcal{T}_R^{i+1}$ for some $i < |Tr|$. So we can soundly define

$$\mathcal{T}_R = \mathcal{T}_R^i \text{ for which } \mathcal{T}_R^i = \text{REFINE}(\mathcal{T}_R^i, R, R)$$

We now define d-important traps R, and partitions $\mathcal{T}_R^0$, inductively according to cardinality of R, denoted $|R|$:

- The (unique) trap with cardinality 0 (i.e., $R = \emptyset$) is d-important.
- $\mathcal{T}_\emptyset^0$ is the partition L-PART induced by transition labels.
- A trap R with cardinality $n+1$ is d-important iff we have some of the following cases:
  - $R = \text{MAXTRAP}(R' \cup \text{PRE}(T))$ for some d-important trap $R' \subset R$ (hence $|R'| \le n$) and a class T of partition $\mathcal{T}_{R'}$; or
  - $R = R_1 \cup R_2$ for some d-important traps R_1, R_2 with $|R_1| \le n$, $|R_2| \le n$.
- For a d-important trap $R \neq \emptyset$ we define $\mathcal{T}_R^0$ as

  the intersection of all $\text{REFINE}(\mathcal{T}_{R'}, R', R)$

  for d-important traps $R' \subset R$ (hence $|R'| < |R|$). (Note that we refine wrt R, not R'.)

We first note

Proposition 5.4 *There is a polynomial space algorithm which, given a BPP system $\Sigma = (P, Tr, \text{PRE}, F, \ell)$ and a set of places $R \subseteq P$, decides if R is a d-important trap, and in the positive case constructs partition $\mathcal{T}_R$.*

Proof: Roughly speaking, the recursive algorithm induced by the definition of d-important traps and respective partitions can be realized as a traversal of a 'tree' of depth at most $k = |P|$ while each node of the tree is of polynomial size.

More specifically, we can sketch two interrelated recursive procedures: $\text{DIMP}(R)$ decides if R is a d-important trap, $\text{PART}(R)$ constructs partition $\mathcal{T}_R$.

$\text{DIMP}(R)$:

```
if R = ∅ then RETURN TRUE ;
for all R1, R2 ⊂ R s.t. R1 ∪ R2 = R do
    if DIMP(R1) and DIMP(R2)
    then RETURN TRUE ;
for each R' ⊂ R s.t. DIMP(R') do
    T := PART(R') ;
    for each class T of T do
        if R = MAXTRAP(R' ∪ PRE(T))
        then RETURN TRUE ;
RETURN FALSE
```

$\text{PART}(R)$:

```
if DIMP(R) = FALSE then FAIL ;
T := L-PART ;
for each R' ⊂ R s.t. DIMP(R') do
    T := T ∩ REFINE(PART(R'), R', R) ;
for i := 1 to |Tr| do
    T := REFINE(T, R, R) ;
RETURN T
```

We can easily check that the depth of recursion is $|P|$ at most and that the procedures can thus run in polynomial space. $\square$

We now define a special mapping $\mathcal{C}$ by

$$\mathcal{C}(M) = \lambda R\,.\,\text{NORMS}_{R,\mathcal{T}_R}(M)$$

where R ranges over d-important traps.

We say that M, M' *agree for* R iff

$$\text{NORMS}_{R,\mathcal{T}_R}(M) = \text{NORMS}_{R,\mathcal{T}_R}(M')$$

Thus $M \equiv_{\mathcal{C}} M'$ iff M, M' agree for each d-important R.

From Proposition 5.4 (also recalling Claim 4.3 and the note afterwards) we easily derive

Lemma 5.5 *There is a polynomial space algorithm which, given a BPP system Σ and markings M, M', decides whether or not $M \equiv_{\mathcal{C}} M'$.*

We now show that bisimilarity $\sim$ coincides with $\equiv_{\mathcal{C}}$ (and thus $\equiv_{\mathcal{C}} = \equiv_{\mathcal{C}_\Sigma}$). This will follow from Lemmas 5.9 and 5.11. We first introduce some auxiliary notions and claims.

In the rest of the section, R, R' always denote d-important traps. We say that a pair

(R', j) is *smaller than* (R, i)

iff $R' \subset R$ or $R' = R$ and $j < i$.

We now extend the above definition of 'agreeing'. We say that M, M' *agree for* (R, i) iff

$$\text{NORMS}_{R,\mathcal{T}_R^i}(M) = \text{NORMS}_{R,\mathcal{T}_R^i}(M') .$$

We say that M, M' *agree below* (R, i) iff M, M' agree for each (R', j) smaller than (R, i).

Claim 5.6 *Assume that M_1, M_1' agree below (R, i), $M_1|_R = M_1'|_R = \mathbf{0}$ and $M_1 \xrightarrow{t} M_2$, $M_1' \xrightarrow{t'} M_2'$, where $\ell(t) = \ell(t')$ but t, t' are in* different classes *of $\mathcal{T}_R^i$. Then M_2, M_2' do not agree for some pair (R', j) smaller than (R, i).*

Proof: There must be a pair (R', j) smaller than (R, i) such that t, t' are in the same class of $\mathcal{T}_{R'}^j$ and in different classes of $\text{REFINE}(\mathcal{T}_{R'}^j, R', R)$. So t, t' are in different classes of $\text{PART}_{\mathcal{F}}^R$ where $\mathcal{F} = \text{NORMS}_{R',\mathcal{T}_{R'}^j}$. And we apply Claim 5.1. $\square$

Claim 5.7 *If R is a d-important trap and $\neg\text{ZERO}_R(M)$, $\text{ZERO}_R(M')$ then there is some (d-important) $R' \subset R$ for which M, M' do not agree.*

Proof: W.l.o.g. assume that R is minimal (d-important trap) such that $\neg\text{ZERO}_R(M)$, $\text{ZERO}_R(M')$. Necessarily, $R = \text{MAXTRAP}(R' \cup \text{PRE}(T))$ for some d-important trap $R' \subset R$ and a class T of partition $\mathcal{T}_{R'}$.

Moreover, $\text{ZERO}_{R'}(M)$ and $\text{ZERO}_{R'}(M')$. But then $\text{NORM}(R' \cup \text{PRE}(T))(M) = \omega$ and $\text{NORM}(R' \cup \text{PRE}(T))(M') < \omega$. $\square$

We introduce

$$\text{MUIT}(M)$$

as the maximal unmarked d-important trap in M (i.e., the union of all d-important traps which are unmarked in M). The previous claim shows that $M \equiv_{\mathcal{C}} M'$ implies $\text{ZERO}_R(M) \Leftrightarrow \text{ZERO}_R(M')$ for each d-important trap R; so we have:

Claim 5.8 *If $M \equiv_{\mathcal{C}} M'$ then $\text{MUIT}(M) = \text{MUIT}(M')$.*

Lemma 5.9 *Mapping $\mathcal{C}$ is bis-necessary. ($M \sim M' \Rightarrow \mathcal{C}(M) = \mathcal{C}(M')$.)*

Proof: Suppose the claim of the lemma does not hold. Then there is a minimal pair (R, i) such that there are $M_1 \sim M_1'$ which do not agree for (R, i). We can even suppose $\neg\text{ZERO}_{R\cup\text{PRE}(T)}(M_1)$ and $\text{ZERO}_{R\cup\text{PRE}(T)}(M_1')$ for a class T of $\mathcal{T}_R^i$ (cf. Claim 3.5).

Since necessarily $\text{ZERO}_R(M_1)$ (recall Claim 5.7), we have $\neg\text{ZERO}_{\text{PRE}(T)}(M_1)$.

So some $t \in T$ is enabled in M_1; let $M_1 \xrightarrow{t} M_2$. Then there must be t', $\ell(t') = \ell(t)$, such that $M_1' \xrightarrow{t'} M_2'$ and $M_2 \sim M_2'$. But since $\text{ZERO}_{\text{PRE}(T)}(M_1')$, necessarily $t' \notin T$. So Claim 5.6 shows that M_2, M_2' do not agree for a pair (R', j) smaller than (R, i) – which is a contradiction. $\square$

Claim 5.10 *Suppose $\text{MUIT}(M_1) = \text{MUIT}(M_1') = R$ and*

$$M_1 \xrightarrow{t} M_2,\ M_1' \xrightarrow{t'} M_2'$$

for t, t' being from the same class of $\mathcal{T}_R$. For each $R' \subseteq R$, if M_1, M_1' agree for R' then M_2, M_2' agree for R' as well.

Proof: Since t, t' are in the same class of $\mathcal{T}_R$, they are necessarily in the same class of $\text{REFINE}(\mathcal{T}_{R'}, R', R)$, hence in the same class of $\text{PART}_{\mathcal{F}}^R$ where $\mathcal{F} = \text{NORMS}_{R',\mathcal{T}_{R'}}$. And we apply Claim 5.2. $\square$

Lemma 5.11 *The equivalence $\equiv_{\mathcal{C}}$ is a bisimulation. (Mapping $\mathcal{C}$ is lc-closed.)*

Proof: Suppose $M_1 \equiv_{\mathcal{C}} M_1'$ and $M_1 \xrightarrow{t} M_2$. Let

$$R = \text{MUIT}(M_1) = \text{MUIT}(M_1')$$

(recall Claim 5.8), and let T be the class of $\mathcal{T}_R$ containing t.

We have $\text{ZERO}_R(M_1)$ and $\neg\text{ZERO}_{\text{PRE}(T)}(M_1)$; so $\text{NORM}(R \cup \text{PRE}(T))(M_1) \geq 1$.

Since $\text{ZERO}_R(M_1')$ and

$$\text{NORMS}_{R,\mathcal{T}_R}(M_1) = \text{NORMS}_{R,\mathcal{T}_R}(M_1')$$

we have $\neg\text{ZERO}_{\text{PRE}(T)}(M_1')$.

So there is $t' \in T$ which is enabled in M_1'; let $M_1' \xrightarrow{t'} M_2'$. We surely have $\ell(t) = \ell(t')$ and $M_2 \equiv_{\mathcal{C}} M_2'$ (cf. Claim 5.10). $\square$

From Lemmas 5.9, 5.11 and 5.5 we derive

Theorem 5.12 *Problem Bis-BPP is in PSPACE (and is thus PSPACE-complete).*

6. Additional remarks

Confirmation of $X^2 \sim X^3 \Rightarrow X \sim X^2$

Researchers studying problems for BPP have recognized the surprisingly 'puzzling' character of the question if $X^2 \sim X^3 \Rightarrow X \sim X^2$ (X denoting a BPP-variable [or nonterminal].) The question is now answered positively by the next claim.

Claim 6.1 *For a BPP system Σ, if $M \sim M + M'$ then $M \sim M''$ for each M'' where $M''(p) = M(p)$ for all p s.t. $M'(p) = 0$ and $M''(p) \geq 1$ for all p s.t. $M'(p) \geq 1$.*

Proof: We recall Theorem 4.7 and note that for each linear function L we have: if $L(M) = L(M + M')$ then either $L(M) = \omega$ or $L(M') = 0$; and we can easily verify that in both cases $L(M) = L(M'')$. $\square$

Weak bisimilarity

In the case of *weak bisimilarity*, we allow transitions to be labelled by silent actions τ and define $M_1 \overset{a}{\Longrightarrow} M_2$ iff $M_1 \xrightarrow{\tau^i a \tau^j} M_2$ for some $i, j \geq 0$; and $M_1 \overset{\tau}{\Longrightarrow} M_2$ iff $M_1 \xrightarrow{\tau^i} M_2$ for some i. Then we define the (weak) bisimulation equivalence $\approx$ wrt the relations $\overset{a}{\Longrightarrow}$. One indication that $\approx$ is more complicated than $\sim$ is shown by the next example where $X^2 \approx X^3$ but $X \not\approx X^2$:

$$X \xrightarrow{\tau} \varepsilon, X \xrightarrow{\tau} Y, X \xrightarrow{\tau} Z, Y \xrightarrow{a} Y, Z \xrightarrow{b} Z$$

Nevertheless, the author conjectures that the method introduced here for strong bisimilarity will also turn out useful for showing decidability of weak bisimilarity for BPP (which is an open problem so far).

Some other related problems

The results presented here surely open a way for clarifying complexity of other problems for BPP, like, e.g., the problem of deciding $M \sim f$ (f being a finite-state system) and the 'regularity' problem (given a BPP M, is there a finite-state system f such that $M \sim f$?). In particular, applying the described algorithm for normed BPP (where the only trap is the empty set) should readily provide a polynomial time bound of a 'reasonable' degree (like $O(n^6)$ or better); it seems worth to compare this with the algorithm in [7] (where no upper bound was stated explicitly).

Acknowledgements

The author has been supported by the Grant Agency of the Czech Republic, grant No. 201/03/1161.

References

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

[2] 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*, pages 545–623. Elsevier, 2001.

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

[4] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proc. CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer, 1993.

[5] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121(2):143–148, 1995.

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

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

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

[9] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

[10] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[11] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[12] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proc. FOCS'98*, pages 120–129. IEEE Computer Society Press, 1998.

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

[14] J. Srba. Roadmap of infinite results. *Bull. of EATCS*, 78:163–175, Oct. 2002. (See also an updated online version at http://www.brics.dk/~srba/roadmap/).

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

[16] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, 195(2):113–131, 1998.

[17] C. Stirling. Decidability of bisimulation equivalence for pushdown processes. http://www.dcs.ed.ac.uk/home/cps/, 2000.