Techniques for Decidability and Undecidability of Bisimilarity

Petr Jančar[1⋆] and Faron Moller[2⋆⋆]

[1] Technical University of Ostrava, Czech Republic
Petr.Jancar@vsb.cz
[2] Uppsala University, Sweden
fm@csd.uu.se

Abstract. In this tutorial we describe general approaches to deciding bisimilarity between vertices of (infinite) directed edge-labelled graphs. The approaches are based on a systematic search following the definition of bisimilarity. We outline (in decreasing levels of detail) how the search is modified to solve the problem for finite graphs, BPP graphs, BPA graphs, normed PA graphs, and normed PDA graphs. We complete this by showing the technique used in the case of graphs generated by one-counter machines. Finally, we demonstrate a general reduction strategy for proving undecidability, which we apply in the case of graphs generated by state-extended BPP (a restricted form of labelled Petri nets).

1 Bisimulation Equivalence

The ***bisimulation game*** is played on a "board" which consists of a (generally infinite) directed edge-labelled multigraph (several edges can lead between two vertices), simply called a ***graph*** in the following. We assume that this graph is labelled from a *finite* set of labels, and that it is ***finite-branching***: that every vertex has finite out-degree. In particular, it is ***image-finite***: for every vertex E and every label a, the set $succ_a(E) = \{F : E \stackrel{a}{\to} F\}$ is finite. We also asume that these successor sets are effectively constructible.

A game is defined by two vertices E_0 and F_0 of a graph, as well as a predetermined time limit $n \in \mathbb{N} \cup \{\omega\}$ (where $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$); we specify such a game by $G_n(E_0, F_0)$. We have two ***players*** competing in the game whom we refer to as **Alice** (the *"Attacker"*) and **Bob** (the *"Bisimulator"*). Their individual goals are as follows:

1. **Alice** wants to show that E_0 and F_0 are in some sense "different".
2. **Bob** wants to show that E_0 and F_0 are in the same sense "the same".

⋆ This paper was written during a visit by the first author to Uppsala University supported by a grant from the Swedish STINT Fellowship Programme. He is also partially supported by the Grant Agency of the Czech Republic, Grant No. 201/97/0456.

⋆⋆ The second author is supported by Swedish TFR grants No. 221-98-103 *"Verification of Infinite State Automata"* and 221-97-275 *"Games for Processes"*.

The sense in which two vertices are deemed to be the same is given by the rules for playing the game. A ***play*** of the game is a sequence of pairs $(E_0, F_0)\,(E_1, F_1)\cdots$ of length $\leq(1{+}n)$, with the next pair in the sequence after (E_i, F_i) arising as follows:

1. **Alice** chooses an edge $E_i \xrightarrow{a} E_{i+1}$ or $F_i \xrightarrow{a} F_{i+1}$;
2. **Bob** chooses a matching edge $F_i \xrightarrow{a} F_{i+1}$ or $E_i \xrightarrow{a} E_{i+1}$.

Alice is thus acting as an attacker, trying to choose an edge leading out of one of the vertices which she believes cannot be matched by any edge (with the same label) leading out of the other vertex; **Bob** on the other hand is defending his thesis that the vertices are equal, that any edge leading out of either of the vertices has a matching edge leading out of the other vertex. **Alice** wins a play of the game if **Bob** ever gets stuck (that is, if he cannot respond to a move by **Alice**); and **Bob** wins any play of length n (that is, any "timed-out" play in which the players have exchanged n moves) as well as any play in which **Alice** finds herself with no move possible (that is, if there are no edges leading out of either of the two specified vertices).

We are interested in knowing if **Bob** has a winning (i.e., defending) strategy for the game $G_n(E_0, F_0)$, that is, if he is able to win *any* play of the game regardless of the moves made by **Alice**. To this end, we make the following definitions. For $n{\in}\mathbb{N}$, we say that E_0 and F_0 are ***n-game equivalent*** (written $E_0 \sim_n F_0$) iff **Bob** has a winning strategy for the game $G_n(E_0, F_0)$; and we say that E_0 and F_0 are ***game equivalent*** (written $E_0 \sim F_0$) iff **Bob** has a winning strategy for the game $G_\omega(E_0, F_0)$. The relation $\sim$ is referred to as ***bisimulation equivalence***, or ***bisimilarity***. Before proceeding, it is worth recording the following straightforward facts.

Fact 1 *$\sim_n$ and $\sim$* are *equivalence (reflexive, symmetric, transitive) relations.*

Fact 2 $\sim_0 \;\supseteq\; \sim_1 \;\supseteq\; \sim_2 \;\supseteq\; \sim_3 \;\supseteq\; \cdots \;\supseteq\; \sim.$

The following fact gives us an *inductive* characterisation of the finite-game equivalences.

Fact 3 ("Stratified Bisimilarity")

1. $E \sim_0 F$ *for all* E, F.
2. $E \sim_{n+1} F$ *iff*
   (a) *if* $E \xrightarrow{a} E'$ *then* $F \xrightarrow{a} F'$ *with* $E' \sim_n F'$*;*
   (b) *if* $F \xrightarrow{a} F'$ *then* $E \xrightarrow{a} E'$ *with* $E' \sim_n F'$*.*

Fact 3 can be used as the basis of a recursive algorithm for determining if $E \sim_n F$, that is, if **Bob** has a winning strategy for $G_n(E, F)$. Less obvious is how to devise an algorithm for determining if $E \sim F$. As a start, the following fact gives us a *coinductive* characterisation of the infinite-game equivalence.

Fact 4 ("Bisimilarity")

$$\begin{array}{l} \sim \ \textit{is the } \textbf{\textit{largest}} \textit{ relation } \equiv \textit{ satisfying: if } E \equiv F \textit{ then} \\ \quad 1. \ \textit{if } E \xrightarrow{a} E' \textit{ then } F \xrightarrow{a} F' \textit{ with } E' \equiv F'; \\ \quad 2. \ \textit{if } F \xrightarrow{a} F' \textit{ then } E \xrightarrow{a} E' \textit{ with } E' \equiv F'. \end{array}$$

The characterisation given in Fact 4 does not in general give rise to any effective procedure for determining if $E \sim F$. However, given our assumption of image-finiteness, we can use the following fact to exploit the finite-game characterisation of Fact 3 to verify that $E \not\sim F$.

Fact 5 *For* ***image-finite*** *graphs,* $\sim \ = \ \bigcap_{n \in \mathbb{N}} \sim_n$.

Hence in the case of image-finite graphs, the non-equivalence problem $E \not\sim F$ is semi-decidable: we simply use Fact 3 to find the smallest n such that $E \not\sim_n F$.

We can be more explicit as to what constitutes a winning strategy for **Bob** for the game $G_\omega(E_0, F_0)$; this is a set $\mathcal{B}$ of pairs of vertices containing (E_0, F_0) which satisfies the following property:

$$\begin{array}{l} \text{For every pair } (E, F) \in \mathcal{B}, \\ \quad 1. \ \text{if } E \xrightarrow{a} E' \text{ then } F \xrightarrow{a} F' \text{ with } (E', F') \in \mathcal{B}; \\ \quad 2. \ \text{if } F \xrightarrow{a} F' \text{ then } E \xrightarrow{a} E' \text{ with } (E', F') \in \mathcal{B}. \end{array}$$

That such a set constitutes a winning strategy for **Bob** is clear: he merely uses this set to choose matching edges, maintaining the invariant that the pair offered to **Alice** is in the set $\mathcal{B}$. Furthermore, if **Bob** has a winning strategy for the game $G_\omega(E_0, F_0)$, then this strategy can be represented by such a set: we merely take the collection of all pairs which appear after every exchange of moves during any and all plays in which **Bob** uses this strategy. A set $\mathcal{B}$ which satisfies the above property is referred to as a ***bisimulation relation***. Thus $\sim$ is the maximal bisimulation and the following is apparent.

Fact 6 $\sim$ *is the union of all bisimulation relations (winning strategies for* ***Bob****).*

2 Decidability

Deciding whether or not $E_0 \sim F_0$ amounts to deciding whether or not there is a bisimulation relation containing the pair (E_0, F_0). A straightforward idea for tackling this problem is to employ some systematic search for such a potential bisimulation. In Section 2.1 we describe a general procedure which arises naturally from the definition of a bisimulation, and demonstrate its (modified) use in various contexts, particularly in Section 2.2 for classes of graphs determined by various process algebras. In Section 2.3 we outline a different approach, exemplified on the class of graphs generated by one-counter machines.

2.1 Expansion trees

Given two sets $B \neq \emptyset$ and A of pairs of vertices, A is called an ***expansion*** of B iff it is a minimal set (wrt inclusion) satisfying the following property:

For every pair $(E, F) \in B$,
1. if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ with $(E', F') \in A$;
2. if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ with $(E', F') \in A$.

Note that a nonempty set *fails* to have an expansion precisely when it contains a pair (E', F') such that $E' \not\sim_1 F'$; and it has (the single) expansion $\emptyset$ iff it contains only pairs of vertices with out-degree zero. Note further that, due to our image-finiteness assumption, a finite set has only finitely many expansions, all of which are finite.

Comparing this definition to that of a bisimulation relation, we observe that a bisimulation relation (containing) B must contain some expansion A of B. The following fact then becomes apparent.

Fact 7 *If $A \subseteq B$ and A is an expansion of B then B is a bisimulation relation, and hence $B \subseteq \sim$.*

More generally, to decide if $E_0 \sim F_0$, that is, if there is a bisimulation B containing the pair (E_0, F_0), we might try to expand $A_0 = \{(E_0, F_0)\}$ recursively (find an expansion A_{k+1} of A_k for each $k = 0, 1, 2, \ldots$) and hope to arrive at a set $A_n \subseteq B = \bigcup_{k<n} A_k$; in this case, some $A \subseteq B$ would be an expansion of B, so Fact 7 would give us that $B \subseteq \sim$, in particular, that $E_0 \sim F_0$.

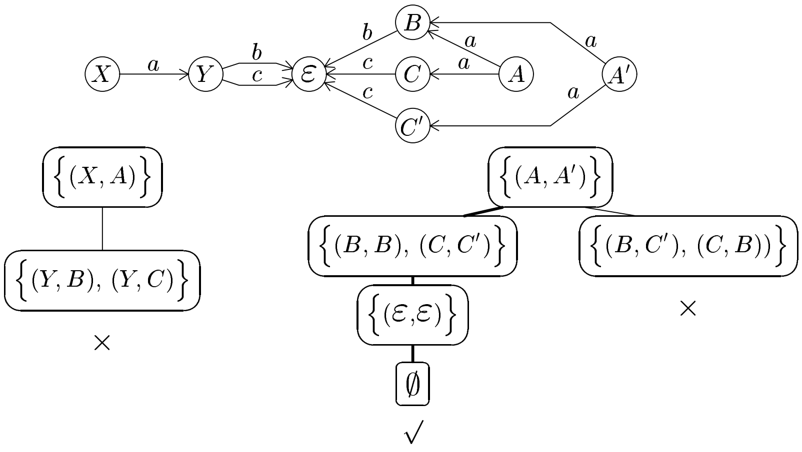
Following this idea, we adapt from Hirshfeld [8] the idea of an ***expansion tree***, which is a (generally infinite) tree whose nodes are (labelled by) sets of pairs of vertices, in which the children of a node are precisely the (finitely many) expansions of that node. The empty node is an example of a ***leaf*** (it has no successors) and is deemed to be ***successful***; all other leaves are ***unsuccessful***. We say that a ***branch*** (a full path through the tree) is ***successful*** iff it is infinite or finishes with a successful node; otherwise it is ***unsuccessful*** (it finishes with an unsuccessful node). We observe that the union of the nodes along a successful branch constitutes a bisimulation, and the following fact thus becomes apparent.

Fact 8 *$E_0 \sim F_0$ iff the expansion tree rooted at $\{(E_0, F_0)\}$ has a successful branch.*

Note that in the case $E_0 \not\sim F_0$, the expansion tree rooted at (E_0, F_0) is necessarily finite, and hence semidecidability of nonbisimilarity once again becomes apparent.

In general, we can search for a (finite) successful branch using breadth-first search (recall that the tree is finite branching). This is surely sufficient in the case of finite acyclic graphs, where the expansion trees are always finite; the search of the expansion tree rooted at $\{(E_0, F_0)\}$ either finds a successful leaf (in which case $E_0 \sim F_0$) or terminates having found only unsuccessful leaves (in which case $E_0 \not\sim F_0$).

Example 1: *Finite acyclic graphs.*
In this example, we have a finite acyclic graph containing vertices X, A



and A' (amongst others), and we demonstrate that $X \not\sim A$ and $A \sim A'$, by building the expansion trees rooted at $\{(X,A)\}$ and $\{(A,A')\}$, respectively. The first tree has only one leaf, which is unsuccessful, while the second tree has two leaves, one of which is successful (the other not).

The naïve procedure described above is of course of little use if the expansion tree of interest has infinitely many nodes. To handle such cases, we shall devise techniques for modifying the tree during construction. What we aim for is an effective construction of a tree which is still finite branching, still has only finite nodes, and such that Fact 8 is maintained: that our pair of vertices is bisimilar iff there is a successful branch in this tree. Finally, we further aim to achieve the ***finite witness property***: if there *are* successful branches, then at least one of these is finite. This property will ensure decidability, as the breadth-first search must either find this finite witness or terminate having discovered all branches are unsuccessful.

The modifications are accomplished through the use of ***rules*** which alter the definition of the children of a node in the tree and are ***safe***, meaning that they maintain Fact 8. To suggest such rules, we first observe the following.

Fact 9 *For any node $A \neq \emptyset$ and any $n \in \mathbb{N}$,*

$$\boxed{A \subseteq \sim_{n+1} \quad \textit{iff} \quad A \textit{ has a child } \ C \subseteq \sim_n .}$$

As a consequence, $A \subseteq \sim$ iff A has a child $C \subseteq \sim$.

This fact implies Fact 8 and hence any rule which respects it maintains safeness. Furthermore, the least n such that $E \not\sim_n F$ for some pair (E,F) in a node gives an upper bound on the depth of the subtree rooted at this node; in other words, if the subtree rooted at B is of depth n, then $E \sim_n F$ for all (E,F) in B. Therefore, any application of the following "abstract" rule maintains Fact 9 (and is hence safe); it can only diminish the size of the tree. (By $A^{\Uparrow}$ we mean the union of all ancestor nodes to A.)

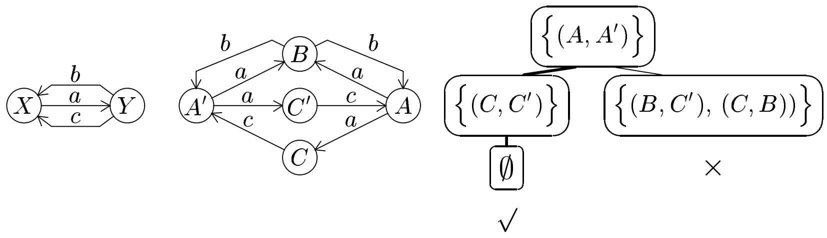
Omitting Rule: We can omit from a node A any pair (E,F) whenever there is $B \subseteq A^{\Uparrow}$ such that $\forall n \in \mathbb{N}$, if $B \subseteq \sim_n$ then $E \sim_n F$.

We call this rule *abstract* since it is in general not effectively computable. A *concrete*, that is, effectively computable, instance of this rule, based on the fact that the relations $\sim_n$ are equivalences, is provided by the following.

Equivalence Rule: Omit from node A the pair (E,F) if it belongs to the least equivalence containing $A^{\Uparrow}$.

This rule, for example, allows us to omit reflexive pairs (E,E) from expansion sets, as well as pairs (E,F) which have appeared in some ancestor node either as (E,F) or symmetrically as (F,E). However, there are only finitely many non-reflexive pairs in the equivalence generated by the finitely-many ancestor pairs; these pairs can be easily computed, showing that this rule is effective. As such, the rule is not too powerful; essentially, it only allows us to handle finite graphs.

Example 2: ***Finite graphs.***
In this example, we have a graph similar to that of Example 1 but with



cycles introduced. Using our newly-introduced simplification rule, we get the same expansion tree as in Example 1 to show that $X \not\sim A$, and the given expansion tree to show that $A \sim A'$.

Note that this is by no means the most efficient algorithm for this problem; better algorithms based on partition refinement have been devised [24, 17]. We merely present this as a basic application of the general method (which we shall apply to various classes of infinite graphs).

Finally, note that the union of the nodes along a successful branch need no longer be a bisimulation; it is now a set B which is guaranteed to have an expansion A such that for all $n \in \mathbb{N}$, $B \subseteq \sim_n$ implies $A \subseteq \sim_n$ (which implies that $B \subseteq \sim$). Sets with this property are also the outcomes (when the input pair *is* bisimilar) of the algorithms outlined further in the following sections.

2.2 Process Algebras

In this section we consider bisimulation equivalence over graphs generated by (subalgebras of) the process algebra PA. A ***PA graph*** is defined by a finite set of ***productions*** of the form $X \overset{a}{\to} \alpha$, where X ranges over a finite set V of ***variables*** (there is at least one production for each $X \in V$), a ranges over a finite set Σ of ***labels***, and α ranges over the terms of the free algebra over V generated by a non-commutative associative operator "$\cdot$" (representing *sequential composition*) and a commutative associative operator "$|$" (representing *parallel composition*); we take ε as the empty term. We shall usually drop the operator "$\cdot$", thus representing the sequential composition of terms by concatenation. The ***vertices*** are given by the terms of the algebra, and the ***edges*** are specified by the production rules as extended to terms as follows: if $\alpha \overset{a}{\to} \beta$ then $\alpha{\cdot}\gamma \overset{a}{\to} \beta{\cdot}\gamma$ and $\alpha|\gamma \overset{a}{\to} \beta|\gamma$ (recall that $|$ is commutative). The subcases of ***BPA graphs*** (***Basic Process Algebra***) and ***BPP graphs*** (***Basic Parallel Processes***) are specified by taking terms from the free algebra over V generated by only one of these operators, "$\cdot$" in the case of BPA and "$|$" in the case of BPP.

As there are only finitely many productions, the graphs are quickly seen to be finitely-branching (and hence image-finite); and taking the algebraic structure of terms into consideration, we can readily verify (by induction) that the relations $\sim_n$ (and hence also $\sim$) are not just equivalences over terms but congruences with respect to the two operators. Hence we can immediately introduce the following safe abstract rule, which is a special instance of the **Omitting Rule**.

> **Congruence Rule**: Omit from node A the pair (α, β) if it belongs to the least congruence containing $A^{\Uparrow}$.

Again, the rule is called abstract since we do not claim its effectiveness in general. However, restricting ourselves to BPP, we notice that the terms can be viewed as nonnegative integer vectors (elements of $\mathbb{N}^k$ for $k = |V|$), and we thus work with finitely generated commutative semigroups, FGCSs (monoids, in fact). Decidability of the word problem for FGCSs (its EXPSPACE-completeness is shown in [19]) implies that the **Congruence Rule** is effective for BPP. Moreover, the result that *every* congruence on a FGCS is finitely generated [25] (so we cannot have an infinite sequence of strictly increasing congruences) shows that applying the rule makes (all branches of) our tree finite.

However, this is not how decidability was originally proved by Christensen, Hirshfeld and Moller [3]. The ingeniously simple rule underlying this algorithm is given next. We suppose (by symmetry) that all pairs (α, β) in nodes are ordered lexicographically: $\alpha <_L \beta$; note that $<_L$ is a well ordering. The rule is then as follows.

> **BPP Rule**: Do not include a pair $(\alpha', \beta|\gamma)$ in a node A when some (α, β) is in $A^{\Uparrow}$; instead consider the pair $(\alpha', \alpha|\gamma)$ (or the symmetric pair, depending on $<_L$).

The rule is obviously effective; it gives rise to a finite chain of considerations (for ever smaller, in $<_L$, right-hand sides) and finishes either by including a pair which has "no bigger" right-hand side than any pair in $A^{\Uparrow}$, or by including nothing (due to reflexivity). Dickson's Lemma [6] (that every infinite sequence in $\mathbb{N}^k$ has an infinite increasing subsequence) then guarantees the finiteness of every branch (every infinite branch is transformed into a finite successful one) and hence the finiteness of the whole tree. Furthermore, the safeness of this rule is guaranteed by the congruence property: if $\alpha \sim_n \beta$, then $\alpha|\gamma \sim_n \beta|\gamma$, so $\alpha' \sim_n \alpha|\gamma$ iff $\alpha' \sim_n \beta|\gamma$.

Remark Basing on Dickson's Lemma only, we cannot provide a primitive recursive bound for the length of branches; nevertheless it is well possible that a detailed analysis of the special case for BPP would reveal an elementary bound (bounded number of exponentiations).

Remark Note that this idea (of lexicographic ordering together with Dickson's Lemma) has been used by Hirshfeld [7] to give a short self-contained proof that *every* congruence on a FGCS is finitely generated.

In fact, knowing that $\sim$ is a finitely generated congruence gives rise to another *semidecision* procedure for $\alpha \sim \beta$: "guess" (i.e., systematically search for) a finite set B of pairs including (α, β), and an expansion A of B together with the proof that A is included in the least congruence containing B. This semidecision procedure can be combined with the semidecision procedure for the negative question described earlier to yield a decision algorithm.

Unlike the case for BPP, for BPA algebras (noncommutative semigroups) we have non-finitely generated congruences (and the word problem is undecidable; see references in [19]). For example, in the free semigroup generated by $\{a, b\}$, the infinite sequence $(bab, baab), (baab, baaab), \ldots, (ba^ib, ba^{i+1}b), \ldots$ has the property that every element does not belong to the least congruence generated by the predecessing elements. Nevertheless, Christensen, Hüttel and Stirling [4] show that bisimulation equivalence on any BPA graph *is* a finitely generated congruence.

Given this fact, we can simply use the idea of combining two semidecision procedures as above. Nevertheless, we give a collection of safe and effective rules for modifying the expansion tree which illustrate the use of certain decompositions, that is, replacing "large" pairs by "smaller" ones. The rules will be instances of the next abstract rule.

Replacing Rule: To a node A, we can add a new sibling node where we replace a pair (E, F) by a (finite) set S of pairs when there is $B \subseteq A^{\Uparrow}$ such that for all $n \in \mathbb{N}$, if $B \subseteq \sim_n$ and $S \subseteq \sim_n$ then $E \sim_n F$.

This rule is readily seen to be safe. Furthermore, if each of two siblings arises from the other through the application of this rule, then we can omit one of them; in this sense, the **BPP Rule** given above is a special instance of this rule.

Before giving our rules, we make the following technical definitions. We say that a term (sequence of variables) α is ***normed*** iff there is a path in the graph from α to ε; we denote by $\mathrm{norm}(\alpha)$ the length of a shortest such path. If α is not normed, then we say that it is ***unnormed***, and we note that $\alpha \sim \alpha\beta$ whenever α is unnormed. Hence we can assume that every term is of the form α or αX where α is normed and X is unnormed.

BPA Rules:
1. If $(X\alpha, Y\beta)$ is in A and some $(X\alpha', Y\beta')$ is in $A^{\Uparrow}$, then we create a sibling node for A containing (α, α') and (β, β') instead of $(X\alpha, Y\beta)$.
2. If $(X\alpha, Y\beta)$ is in A where X and Y are normed, then we create sibling nodes containing ("decomposition pairs") $(X, Y\gamma)$, $(\gamma\alpha, \beta)$ instead, where $\mathrm{norm}(X) = \mathrm{norm}(Y\gamma)$ (there are only finitely many of these).

The symmetric rules, and the use of the **Equivalence Rule**, are supposed implicitly; the safeness is readily verified. Since the rules apply recursively to all newly-created sibling nodes, the finite-branching of our tree can be cast in doubt. But we can demonstrate that the (modified) tree *is* finitely branching by using the following measure of ***size***. We take $\mathrm{size}(\alpha, \beta) = \max\{\mathrm{xnorm}(\alpha), \mathrm{xnorm}(\beta)\}$ where for normed γ and unnormed X, $\mathrm{xnorm}(\gamma) = \mathrm{xnorm}(\gamma X) = \mathrm{norm}(\gamma)$. There are only finitely many pairs with size $\leq n$ (for any n), and we observe that the size of any pair in any node created due to A is bounded by the maximum size in $A \cup A^{\Uparrow}$. In this sense we replace "large" pairs by "smaller" ones.

The finite-witness property is then guaranteed as follows. Suppose there is an infinite branch but no finite successful branch. Then we could extract an infinite sequence of pairs $(X\alpha_1, Y\beta_1), (X\alpha_2, Y\beta_2), \ldots$ with $\alpha_i \not\sim \alpha_j$ or $\beta_i \not\sim \beta_j$ for each $i \neq j$ which are non-decomposable in the above sense, but for which $X\alpha_i \sim Y\beta_i$ for each i; this leads to a contradiction [4] (due to image-finiteness).

Remark Again, this idea provides no reasonable complexity bound. However, the problem is studied in more detail by Burkart, Caucal and Steffen [2], where the algorithm is modified to operate within an elementary bound.

Remark We have exploited certain decomposition properties for BPA. In the normed case, both for BPP and BPA, a unique decomposition (into "prime" processes as defined by Milner and Moller [20]) is guaranteed, which Hirshfeld, Jerrum and Moller [10, 11] exploited to give (nontrivial) proofs that polynomial-time algorithms exist for these cases.

Finally, for the case of PA, the answer is known only in the normed case. Hirshfeld and Jerrum [9] use, in principle, the idea of the modified expansion tree, having found sufficient rules for replacing "large" pairs by "smaller" ones which induce them. They furthermore establish a bound for the "largeness" of the resulting pairs, which results in an upper bound on the complexity. The proof makes substantial use of the decomposition properties of BPA and BPP; but the main challenge is to handle "mixed pairs" $(\alpha{\cdot}\alpha', \beta|\beta')$; it turns out that such (bisimilar) pairs have a surprisingly rich structure. Furthermore, it is not clear if $\sim$ is a finitely generated congruence in the case of (normed) PA. The nodes of the tree in [9] are no longer just finite sets of pairs, but they can also contain "schemes" representing infinite sets.

Remark The authors of [9] do not expect their technique to be directly generalized for the whole class of (unnormed) PA graphs. They illustrate our still insufficient insight by noting that a polynomial algorithm as well as undecidability are still possible in this case.

2.3 State-extended Process Algebras

Richer classes of graphs have been studied, arising from process algebras by adding a finite-state control. Formally, the productions are of the form $pX \stackrel{a}{\to} q\alpha$ where p and q denote states of the finite control. The extension to terms is as follows: if $p\alpha \stackrel{a}{\to} q\beta$ then $p(\alpha{\cdot}\gamma) \stackrel{a}{\to} q(\beta{\cdot}\gamma)$ and $p(\alpha|\gamma) \stackrel{a}{\to} q(\beta|\gamma)$.

State-extended BPA graphs are also called ***PDA graphs*** (from ***push-down automata***). Though language equivalent to BPA, they constitute a richer family of graphs, and to solve the decidability question for bisimilarity needs more insight. Stirling [27] proves the decidability in the *normed* case; a PDA vertex $p\alpha$ is deemed to be normed iff for any $q\beta$ which is reachable from $p\alpha$ there is some $q'\varepsilon$ reachable from $q\beta$.

Stirling also invents a way how to replace "large" pairs $(p\alpha, q\beta)$ by "smaller" ones. To this aim he introduces special stack symbols (constants) which enable to finitely describe (infinite) "regular behaviours" of the (suffix of the) stack and enable a "congruence property" (wrt "stacking"). He shows that it is sufficient to have finitely many such constants but gives no bound on their number, so no bound for the complexity is provided.

Remark Very close to this topic is the long-standing open problem of the decidability of language equivalence for deterministic pushdown automata, which has only recently been solved (positively) by Sénizergues [26]. The full version of his proof is more than 70 pages, and still longer is his announced proof for decidability of bisimilarity for the whole class of PDA graphs. Based on insight got from Sénizergues' proof and on the techniques which we have sketched above, Stirling provides (in May 1999) a much shorter proof (approx. 20 pages) of DPDA-equivalence, and conjectures that his "structural" technique will yield a shorter proof of decidability of bisimilarity as well.

Here we describe in more detail the case of graphs generated by one-counter machines. It is a subcase of PDA, where all considered graph vertices are of the form $pXX\cdots XZ$, and the productions are of the appropriately restricted form; here the number of symbols X is interpreted as the value of the counter while Z is a special "bottom-of-stack" symbol allowing to test for zero. We use the more natural notation $p(m)$ to denote the vertex pX^mZ.

A decidability proof is given by Jančar [13]; however, in view of the above Remark, it is (probably) subsumed by more general results. Nevertheless the technique in [13] is different from what we mentioned so far, and fairly simple to describe. Moreover, it is more amenable for yielding a complexity bound in this (sub)case. Performing again a systematic search, the algorithm constructs (in the case when the given pair *is* bisimilar) a finite representation of a (generally infinite) bisimulation relation (not using the "large-replace-by-small" approach).

Remark This approach could be again illustrated on BPP. Here $\sim$ is a congruence and every congruence on a finitely generated commutative semigroup is semilinear [5] (see [7] for a short proof). The property of a semilinear relation being a bisimulation is easily seen to be expressible in Presburger arithmetic.

Hence due to its decidability (see, e.g., [23]), the semidecidability of bisimilarity for BPP can be shown as follows: systematically generate descriptions of all semilinear relations and check for each of them if it is a bisimulation containing the given pair.

Considering (pairs of) one-counter processes (i.e., vertices in a respective graph), a conceptual advantage is that we have a natural presentation in terms of "colouring" the set $\mathbb{N} \times \mathbb{N}$ (viewed as the integral grid in the first quadrant of the two-dimensional plane). By a ***colouring*** with colours from a finite set C we mean a function $c : \mathbb{N} \times \mathbb{N} \to C$. Given a one-counter machine M (that is, the respective collection of PDA productions) with control state set Q, by a ***colouring*** c ***related to*** M we mean any colouring arising as the product $c = \prod_{p,q \in Q} c_{(p,q)}$ where $c_{(p,q)} : \mathbb{N}\times\mathbb{N} \to \{black, white\}$. Such a colouring c ***represents*** the relation $\mathcal{R}_c$ on the set of vertices of the respective graph as follows: $p(m)\ \mathcal{R}_c\ q(n)$ iff $c_{(p,q)}(m,n)=black$.

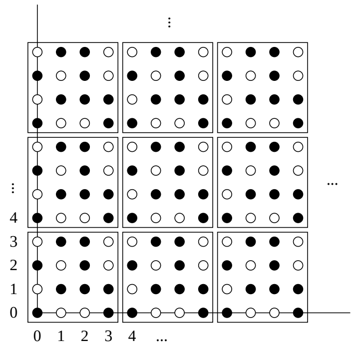
We distinguish the colouring c^M representing bisimulation equivalence:

$$c^M = \prod_{p,q \in Q} c^M_{(p,q)}$$

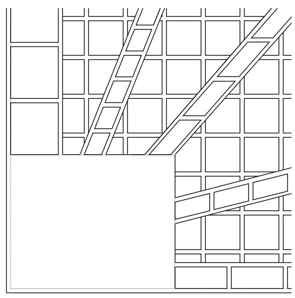
where $c^M_{(p,q)}(i,j) = black$ iff $p(i) \sim q(j)$ and $c^M_{(p,q)}(i,j) = white$ iff $p(i) \not\sim q(j)$. The key fact here is the following.

Fact 10 *For a one-counter machine M, the colouring c^M is regular.*

Here by a ***regular colouring*** we mean a colouring arising from a periodic "background" colouring by changing the colours in finitely many belts (whose slopes are nonnegative rational or ∞) so that each belt colouring is periodic as well, and by final recolouring of an initial square. The following pictures illustrate these notions.



A periodic colouring.



Scheme of a regular colouring.

Therefore, knowing Fact 10, we can generate all (descriptions of) regular colourings c (related to M) and check for each of them whether $\mathcal{R}_c$ is a bisimulation relation containing the given pair (this is straightforward since it is sufficient to check just finitely many pairs, due to the periodicity). This gives a semidecision procedure for bisimilarity. (It can be presented as a decision procedure as in [16] where a similar technique for simulation is used.)

The idea behind the technical details showing Fact 10 can be sketched as follows. If $p(m)$ and $q(n)$ are bisimilar then their distances (i.e., lengths of shortest sequences) to certain vertices with 0 as the counter value must be the same. Such distances are, in principle, linear in the counter value, and the constraint of the mentioned equality yields the linear belts. Periodicity (inside the belts and on the background) then follows from some simple observations.

Remark As mentioned above, a complexity bound can be achieved by a more detailed analysis; nevertheless, it was not done in [13].

3 Undecidability

We certainly cannot expect bisimulation equivalence to be decidable for a class of graphs representing systems which are able to faithfully model universal devices, in particular Minsky machines [21], which are simple straight-line programs which operate on nonnegative counters. It is sufficient for us to assume just two counters, initialized to 0, since the halting problem is undecidable even in this simple case. Formally, a ***Minsky machine*** M is a sequence of labelled instructions

$$\mathtt{X}_0 : \mathrm{comm}_0\,; \quad \mathtt{X}_1 : \mathrm{comm}_1\,; \quad \cdots \quad \mathtt{X}_{n-1} : \mathrm{comm}_{n-1}\,; \quad \mathtt{X}_n : \mathrm{comm}_n$$

where each of the first n instructions is either of the form

$$\mathtt{X}_\ell : \mathtt{c}_0 := \mathtt{c}_0{+}\mathtt{1};\ \mathtt{goto}\ \mathtt{X}_j \qquad \text{or} \qquad \mathtt{X}_\ell : \mathtt{c}_1 := \mathtt{c}_1{+}\mathtt{1};\ \mathtt{goto}\ \mathtt{X}_j$$

or of the form

$$\begin{array}{lcl} \mathtt{X}_\ell : \mathtt{if}\ \mathtt{c}_0 = 0\ \mathtt{then}\ \mathtt{goto}\ \mathtt{X}_j & \text{or} & \mathtt{X}_\ell : \mathtt{if}\ \mathtt{c}_1 = 0\ \mathtt{then}\ \mathtt{goto}\ \mathtt{X}_j \\ \quad\ \mathtt{else}\ \mathtt{c}_0 := \mathtt{c}_0{-}\mathtt{1};\ \mathtt{goto}\ \mathtt{X}_k & & \quad\ \mathtt{else}\ \mathtt{c}_1 := \mathtt{c}_1{-}\mathtt{1};\ \mathtt{goto}\ \mathtt{X}_k \end{array}$$

The machine M starts executing with the value 0 in the counters $\mathtt{c}_0$ and $\mathtt{c}_1$ and the control at label $\mathtt{X}_0$. When the control is at label $\mathtt{X}_\ell$ $(0 \le \ell < n)$, the machine executes instruction comm_ℓ, appropriately modifying the contents of the respective counter and transferring the control to the appropriate label as directed by the instruction. The machine halts if and when the control reaches the `halt` instruction at label $\mathtt{X}_n$.

State-extended PA can easily model Minsky machines, as we can use the parallel composition of two sequential terms $(XX\ldots XZ)|(YY\ldots YZ)$ to model the two counters in the obvious way. So a straightforward reduction from the halting problem shows undecidability of bisimilarity for state-extended PA. Recall that there is no such reduction for (normed) state-extended BPA, that is, for (normed) PDA, due to the decidability in that case, which we discussed in Section 2.3. However, it turns out that ***state-extended BPP***—which we further denote ***MSA*** (***multiset automata***)—allow for such a reduction, even in the normed case, though they cannot faithfully model Minsky machines. They represent a subclass of labelled place/transition Petri nets and thus they lack the ability to test-for-zero. In the rest of this section, we outline the reduction, thus demonstrating undecidability of bisimilarity for MSA.

The idea is to construct an MSA which "weakly" models a given Minsky machine, and then take two slightly modified copies. The modifications guarantee that whenever **Alice** "cheats" (i.e., makes a "zero" move while the counter of the modelled Minsky machine is positive), **Bob** can punish her by reaching a pair of equal (and hence trivially bisimilar) vertices.

To the Minsky machine M as presented above, we define the (description of the) MSA graph as follows.

- The set of labels is $\Sigma = \{ i, d, z, h \}$.
- The set of control states is $Q = \{ p_0, p_1, \ldots, p_{n-1}, p_n, q_0, q_1, \ldots, q_{n-1}, q_n \}$.
- The set of variables is $V = \{ Z, 0, 1 \}$.
- For each machine instruction

  $$\mathtt{X}_\ell : \mathtt{c}_b := \mathtt{c}_b\mathtt{+1};\ \mathtt{goto}\ \mathtt{X}_j$$

  we have the productions

  $$p_\ell Z \xrightarrow{i} p_j(b|Z) \qquad \text{and} \qquad q_\ell Z \xrightarrow{i} q_j(b|Z).$$
- For each machine instruction

  $$\begin{array}{l}\mathtt{X}_\ell : \mathtt{if}\ \mathtt{c}_b = \mathtt{0}\ \mathtt{then}\ \mathtt{goto}\ \mathtt{X}_j \\ \qquad \mathtt{else}\ \mathtt{c}_b := \mathtt{c}_b\mathtt{-1};\ \mathtt{goto}\ \mathtt{X}_k\end{array}$$

  we have the productions

  $$\begin{array}{lll} p_\ell b \xrightarrow{d} p_k & p_\ell Z \xrightarrow{z} p_j Z & p_\ell b \xrightarrow{z} q_j b \\ q_\ell b \xrightarrow{d} q_k & q_\ell Z \xrightarrow{z} q_j Z & q_\ell b \xrightarrow{z} p_j b \end{array}$$
- We have the one final production

  $$p_n Z \xrightarrow{h} p_n$$

The MSA graph reflects the (computation of the) machine M in the following sense.

- When M is at the command labelled X_ℓ with the values x and y in its counters, this is reflected by the "p-vertex" $p_\ell(0^x|1^y|Z)$ as well as the "q-vertex" $q_\ell(0^x|1^y|Z)$ (where 0^x represents $0|0|\ldots|0$ with x occurrences of 0).
- If this command is an increment, then the associated p-vertex has only one outgoing edge, labelled by i (for "increment"), leading to the p-vertex reflecting the state of M upon executing the increment command; similarly for the associated q-vertex.
- If this command is a successful test for zero (that is, the relevant counter has the value 0), then the associated p-vertex (as well as the associated q-vertex) has only one outgoing edge, labelled by z (for "zero"), leading to the p-vertex (q-vertex) reflecting the state of M upon executing the respective command.

– If this command is a decrement (that is, a failed test for zero), then the associated p-vertex (as well as the associated q-vertex) has three outgoing edges, exactly one of which is labelled d (for "decrement") which again leads to the p-vertex (q-vertex) reflecting the state of M upon executing the command.
– In this last instance, the additional two outgoing edges, both labelled by z, reflect the weakness of MSA (or, more generally, Petri nets) in their inability to test for zero (a weakness which works in their favour with respect to several important positive decidability results such as the reachability problem [18]). One of the two edges represents an "honest cheating", and arises from the rule $p_\ell Z \xrightarrow{z} p_j Z$ ($q_\ell Z \xrightarrow{z} q_j Z$), while the other edge represents "knowingly cheating", and arises from the rule $p_\ell b \xrightarrow{z} q_j b$ ($q_\ell b \xrightarrow{z} p_j b$); in this final case, the edge leads to the domain of the other copy weakly modelling M.

Fact 11 $p_0 Z \sim q_0 Z$ *iff the Minsky machine M does not halt.*

To see this, we note that if M halts, then a winning strategy for **Alice** in the bisimulation game would be to mimic the behaviour of M in either of the two copies (in p-vertices or q-vertices). **Bob**'s only option in response would be to do the same in the other copy. Upon termination, the pair of vertices reached in the game will be $p_n(0^x|1^y|Z)$ and $q_n(0^x|1^y|Z)$ for some values x and y. **Alice** may then choose the edge $p_n(0^x|1^y|Z) \xrightarrow{h} p_n(0^x|1^y)$ which cannot be matched by **Bob**. Hence $p_0 Z$ and $q_0 Z$ are not bisimilar.

On the other hand, if M fails to halt, then a winning strategy for **Bob** would be to mimic **Alice**'s moves for as long as **Alice** mimics M, and to cheat knowingly or honestly, respectively, in the instance that **Alice** cheats honestly or knowingly, respectively, so as to arrive at the situation where the two vertices are identical; from here **Bob** can copy every move of **Alice** verbatim. Hence $p_0 Z$ and $q_0 Z$ are bisimilar.

The above MSA graph can be made normed (in the same sense as for PDA) by adding a new label e along with the following productions (one for each $\ell = 0, 1, \ldots, n$ and each $X = Z, 0, 1$).

$$p_\ell X \xrightarrow{e} p_\ell \qquad p_\ell X \xrightarrow{e} q_\ell \qquad\qquad q_\ell X \xrightarrow{e} q_\ell \qquad q_\ell X \xrightarrow{e} p_\ell$$

This guarantees that every vertex has an outgoing path to some vertex $r\varepsilon$, and continues to allow **Bob** to produce a pair of identical vertices if **Alice** chooses one of these non-M-mimicking edges. The same argument can then be made to show that $p_0 Z$ and $q_0 Z$ are bisimilar exactly when the Minsky machine M does not halt. We thus arrive at our result.

Fact 12 *Bisimilarity is undecidable over the class of normed MSA graphs.*

4 Additional Remarks

The undecidability proof in the previous section stems from the proof for labelled Petri nets given by Jančar [12] which was adapted for MSA by Moller [22] (there referred to as PPDA, *parallel pushdown automata*). We thus have undecidability of bisimulation equivalence over a very restricted class of Petri nets: those with only two unbounded places and a minimal degree of nondeterminism. Decidability results for various subcases have been obtained in [12, 15].

The study of the decidability of bisimilarity over various classes of graphs became a popular topic in concurrency theory with the appearance of the first, then-surprising, proof of the decidability result for normed BPA by Baeten, Bergstra, Klop [1]. Since then there has been a great deal of research on the topic, and we have only touched on a fragment of it in this tutorial paper. In particular, we have not discussed *weak* bisimilarity here, which seems much less amenable for obtaining decidability results. (Some partial results are obtained in [14]). For further references in the area, we refer to the survey paper [22].

References

1. J.C.M. Baeten, J.A. Bergstra and J.W. Klop (1993). Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the ACM* **40**:653–682. (Preliminary version in the proceedings of PARLE'87, *Lecture Notes in Computer Science* **259**:94–113, 1987.)
2. O. Burkart, D. Caucal and B. Steffen (1995). An elementary decision procedure for arbitrary context-free processes. Proceedings of MFCS'95. *Lecture Notes in Computer Science* **969**:423–433.
3. S. Christensen, Y. Hirshfeld and F. Moller (1993). Bisimulation equivalence is decidable for basic parallel processes. Proceedings of CONCUR'93, *Lecture Notes in Computer Science* **715**:143–157.
4. S. Christensen, H. Hüttel and C. Stirling (1995). Bisimulation equivalence is decidable for all context-free processes. *Information and Computation* **121**(2):143–148. (Preliminary version in the proceedings of CONCUR'92, *Lecture Notes in Computer Science* **630**:138–147, 1992.)
5. S. Eilenberg and M.P. Schützenberger (1969). Rational sets in commutative monoids. *Journal of Algebra* **13**:173–191.
6. L.E. Dickson (1913). Finiteness of the odd perfect and primitive abundant numbers with distinct factors. *American Journal of Mathematics* **35**:413–422.
7. Y. Hirshfeld (1994). Congruences in commutative semigroups. Research report ECS-LFCS-94-291, Department of Computer Science, University of Edinburgh.
8. Y. Hirshfeld (1997). Bisimulation trees and the decidability of weak bisimulations. *Electronic Notes in Theoretical Computer Science* **5**, http://www.elsevier.nl/locate/entcs/volume5.html.
9. Y. Hirshfeld and M. Jerrum (1999). Bisimulation equivalence is decidable for normed Process Algebra. Proceedings of ICALP'99, to appear, *Lecture Notes in Computer Science* **1644**.
10. Y. Hirshfeld, M. Jerrum and F. Moller (1996). A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science* **158**:143–159. (Preliminary version in the proceedings of FOCS'94:623–631, 1994.)
11. Y. Hirshfeld, M. Jerrum and F. Moller (1996). A polynomial algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Mathematical Structures in Computer Science* **6**:251–259.

12. P. Jančar (1995). Undecidability of bisimilarity for Petri nets and related problems. *Theoretical Computer Science* **148**:281–301. (Preliminary version in the proceedings of STACS'94, *Lecture Notes in Computer Science* **775**:581–592, 1994.)
13. P. Jančar (1997). Bisimulation equivalence is decidable for one-counter processes. Proceedings of ICALP'97, *Lecture Notes in Computer Science* **1256**:549-559. (To appear in *Information and Computation.*)
14. P. Jančar, A. Kučera and R. Mayr (1998). Deciding bisimulation-like equivalences with finite-state processes. Proceedings of ICALP'98, *Lecture Notes in Computer Science* **1443**:200–211. (To appear in *Theoretical Computer Science.*)
15. P. Jančar, J. Esparza and F. Moller (1999). Petri nets and regular processes. *Journal of Computer and System Sciences* (to appear). (Research report 162, Computing Science Department, Uppsala University.
16. P. Jančar and F. Moller (1999). Simulation of one-counter nets via colouring. Research report 159, Computing Science Department, Uppsala University.
17. P.C. Kanellakis and S.A. Smolka. CCS expressions, finite-state processes and three problems of equivalence. *Information and Computation* **86**:43–68, 1990.
18. E. Mayr (1984). An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing* **13**:441–460.
19. E.W. Mayr and A.R. Meyer (1982). The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics* **46**:305–329.
20. R. Milner and F. Moller (1993). Unique decomposition of processes. *Theoretical Computer Science* **107**:357–363. (Preliminary version in the *Bulletin of the EATCS* **41**:226–232, 1990.)
21. M. Minsky (1967). **Computation: Finite and Infinite Machines**. Prentice-Hall.
22. F. Moller (1996). Infinite results. Proceedings of CONCUR'96, *Lecture Notes in Computer Science* **1119**:195–216.
23. D.C. Oppen (1978). A $2^{2^{2^{p(n)}}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Science* **16**:323–332.
24. R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing* **16**:937–989, 1987.
25. L. Redei (1965). **The theory of finitely generated commutative semigroups.** Oxford University Press.
26. G. Sénizergues (1997). The Equivalence Problem for Deterministic Pushdown Automata is Decidable. Proceedings of ICALP'97, *Lecture Notes in Computer Science* **1256**:671–681.
27. C. Stirling (1998). Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science* **195**:113–131. (Preliminary version in the proceedings of CONCUR'96, *Lecture Notes in Computer Science* **1119**:217–232, 1996.)