FI MU

Faculty of Informatics
Masaryk University

Efficient Verification Algorithms for One-Counter Processes

by

Antonín Kučera

FI MU Report Series FIMU-RS-2000-03

March 2000

Efficient Verification Algorithms for One-Counter Processes

Antonín Kučera*
Faculty of Informatics
Masaryk University
Botanická 68a, 60200 Brno
Czech Republic
tony@fi.muni.cz

Abstract

We study the problem of strong/weak bisimilarity between processes of one-counter automata and finite-state processes. We show that the problem of weak bisimilarity between processes of one-counter nets (which are 'weak' one-counter automata) and finite-state processes is **DP**-hard (in particular, it means that the problem is both **NP** and **co-NP** hard). The same technique is used to demonstrate **co-NP**-hardness of strong bisimilarity between processes of one-counter nets. Then we design an algorithm which decides weak bisimilarity between processes of one-counter automata and finite-state processes in time which is polynomial for most 'practical' instances, giving a characterization of all hard instances as a byproduct. Moreover, we show how to efficiently compute a rather tight bound for the time which is needed to solve a given instance. Finally, we prove that the problem of strong bisimilarity between processes of one-counter automata and finite-state processes is in **P**.

*Supported by the Grant Agency of the Czech Republic, grants No. 201/98/P046 and No. 201/00/0400.

1 Introduction

In concurrency theory, *processes* are typically understood as (being associated with) states in *transition systems*, a fundamental and widely accepted model of discrete systems. Formally, a transition system is a triple $\mathcal{T} = (S, \Sigma, \rightarrow)$ where S is a set of *states*, Σ is a finite set of *actions* (or *labels*), and $\rightarrow \subseteq S \times \Sigma \times S$ is a *transition relation*. We write $s \stackrel{a}{\rightarrow} t$ instead of $(s, a, t) \in \rightarrow$ and we extend this notation to elements of Σ^* in the natural way. A state t is *reachable* from a state s iff there is $w \in \Sigma^*$ such that $s \stackrel{w}{\rightarrow} t$. A system $\mathcal{T}$ is *finite-state* iff the set of states of $\mathcal{T}$ is finite.

The *equivalence approach* to formal verification of concurrent systems is based on the following scheme: One describes the *specification* (the intended behaviour) $\mathcal{S}$ and the *implementation* $\mathcal{I}$ of a given system in some 'higher' formalism whose semantics is given in terms of transition systems, and then it is shown that $\mathcal{S}$ and $\mathcal{I}$ are *equivalent*. Actually, there are many ways how to capture the notion of process equivalence (see, e.g., [vG90]). It seems, however, that *bisimulation equivalence* [Par81, Mil89] is of special importance, as its accompanying theory has been developed very intensively. Let $\mathcal{T} = (S, \Sigma, \rightarrow)$ be a transition system. A binary relation $R \subseteq S \times S$ is a *bisimulation* iff whenever $(s, t) \in R$, then for each $s \stackrel{a}{\rightarrow} s'$ there is some $t \stackrel{a}{\rightarrow} t'$ such that $(s', t') \in R$, and for each $t \stackrel{a}{\rightarrow} t'$ there is some $s \stackrel{a}{\rightarrow} s'$ such that $(s', t') \in R$. States s, t are *bisimulation equivalent* (or *bisimilar*), written $s \sim t$, iff there is a bisimulation relating them. Bisimulations can also be used to relate states of *different* transition systems; formally, two systems can be considered as a single one by taking their disjoint union. An important variant of bisimilarity is *weak bisimilarity* introduced by Milner in his work on CCS [Mil89]. This relation distinguishes between 'external' and 'internal' computational steps, and allows to 'ignore' the internal steps (which are usually denoted by a distinguished action τ) to a certain extent. Formally, we define the *extended transition relation* $\Rightarrow \subseteq S \times \Sigma \times S$ as follows: $s \stackrel{\tau}{\Rightarrow} t$ iff t is reachable from s via a finite (and possibly empty) sequence of transitions labelled by τ (note that $s \stackrel{\tau}{\Rightarrow} s$ for each s), and $s \stackrel{a}{\Rightarrow} t$ where $a \neq \tau$

iff there are states u, v such that $s \stackrel{\tau}{\Rightarrow} u \stackrel{a}{\rightarrow} v \stackrel{\tau}{\Rightarrow} t$. The relation of *weak bisimulation* is defined in the same way as bisimulation, but '$\Rightarrow$' is used instead of '$\rightarrow$'. Processes s, t are *weakly bisimilar*, written $s \approx t$, iff there is a weak bisimulation relating them. To prevent a confusion about bisimilarity and weak bisimilarity, we refer to bisimilarity as *strong bisimilarity* in the rest of this paper.

In this paper we study the complexity of checking strong and weak bisimilarity between processes of transition systems generated by (certain subclasses of) *pushdown automata* and processes of finite-state systems. A *pushdown automaton* is a tuple $\mathcal{P} = (Q, \Gamma, \Sigma, \delta)$ where Q is a finite set of *control states*, Γ is a finite *stack alphabet*, Σ is a finite *input alphabet*, and $\delta : (Q \times \Gamma) \rightarrow 2^{\Sigma \times (Q \times \Gamma^*)}$ is a *transition function* with finite image. We can assume (w.l.o.g.) that each transition increases the height (or length) of the stack at most by one (each PDA can be efficiently transformed to this kind of normal form). To $\mathcal{P}$ we associate the transition system $\mathcal{T}_\mathcal{P}$ where $Q \times \Sigma^*$ is the set of states, Σ is the set of actions, and the transition relation is determined by

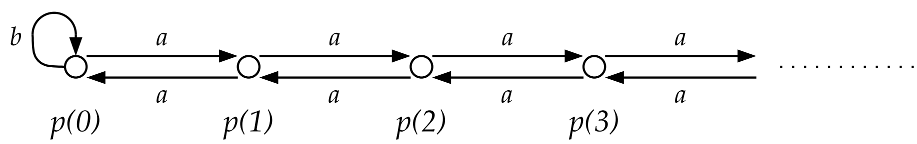
$$(p, A\alpha) \stackrel{a}{\rightarrow} (q, \beta\alpha) \text{ iff } (a, (q, \beta)) \in \delta(p, A).$$

As usual, we write $p\gamma$ instead of (p, γ) and we use ε to denote the empty word. The size of $\mathcal{P}$ is the length of a string which is obtained by writing all elements of the tuple linearly in binary. The size of a process $p\alpha$ of is the length of its corresponding binary encoding. Pushdown processes (i.e., processes of pushdown automata) have their origin in theory of formal languages [HU79], but recently (i.e., in the last decade) they have been found appropriate also in the context of concurrency theory because they provide a natural and important model of sequential systems. In this paper we mainly concentrate on a subclass of pushdown automata where the stack behaves like a *counter*. Such a restriction is reasonable because in practice we often meet systems which can be abstracted to finite-state programs operating on a single unbounded variable. For example, network protocols can maintain the count on how many unacknowledged messages have been sent, printer spool should know how many processes are waiting in the input queue, etc. Formally, a *one-counter automaton* $\mathcal{A}$ is a pushdown automaton with just two stack symbols I and Z; the transition function δ of $\mathcal{A}$ is a union of functions δ_Z and δ_I where $\delta_Z : (Q \times \{Z\}) \to 2^{\Sigma \times (Q \times (\{I\}^* \{Z\}))}$ and $\delta_I : (Q \times \{I\}) \to 2^{\Sigma \times (Q \times \{I\}^*)}$. Hence, Z works like a bottom symbol (which cannot be removed), and the number of I's which are stored in the stack represents the counter value. Processes of $\mathcal{A}$ (i.e., states of $\mathcal{T}_{\mathcal{A}}$) are of the form pI^iZ. In the rest of this paper we adopt a more intuitive notation, writing $p(i)$ instead of pI^iZ. It is worth to note that the size of $p(i)$ is $\mathcal{O}(i)$ and *not* $\mathcal{O}(\log i)$, because $p(i)$ is just a *symbolic* abbreviation for $p\alpha Z$ where α is a string of i symbols I. Again, we assume (w.l.o.g) that each transition increases the counter at most by one. A proper subclass of one-counter automata of its own interest are *one-counter nets*. Intuitively, OC-nets are 'weak' OC-automata which cannot test for zero explicitly. They are computationally equivalent to a subclass of Petri nets [Rei85] with (at most) one unbounded place. Formally, a *one-counter net* $\mathcal{N}$ is a one-counter automaton such that whenever $(a, qI^iZ) \in \delta_Z(p, Z)$, then $(a, qI^{i+1}) \in \delta_Z(p, I)$. In other words, each transition which is enabled at zero-level is also enabled at (each) non-zero-level. Hence, there are no 'zero-specific' transitions which could be used to 'test for zero'.

As a simple example, we might take $\mathcal{A} = (\{p\}, \{I, Z\}, \{a\}, \delta)$, where

$$\delta(pZ) = \{(b, pZ)\}, \;\; \delta(pI) = \{(a, pII), (a, p\varepsilon)\}$$

Note that $\mathcal{A}$ is *not* a one-counter net; however, $\mathcal{A}$ becomes a OC-net as soon as we delete the (only) b-transition. The associated infinite-state transition system $\mathcal{T}_{\mathcal{A}}$ looks as follows:



Observe that the out-going transitions of a OC process $q(i)$ where $i > 0$ do not depend on the actual value of i. Hence, the structure of transition systems which are associated with OC-automata (and, in particular, with OC-nets) is rather regular—they consist of a 'zero pattern' and a 'non-zero pattern' which is repeated infinitely often. Despite this regularity, some problems for OC-automata (and even for OC-nets) are computationally hard, as we shall see in the next section.

Now we give a short summary of relevant results for PDA and OC automata. The decidability of strong bisimilarity for processes of stateless PDA (which are also known as BPA processes) is due to [CHS95]. Another (incomparable) positive result is [Jan97] where it is shown that strong bisimilarity is decidable for processes of OC-automata. These results have been recently extended to general PDA in [Sén98]. The problem of weak bisimilarity is still open for all of the mentioned (sub)classes. The decidability of strong/weak bisimilarity between processes of a (general) class $\mathcal{C}$ and finite-state ones has been studied in [JKM98]. It is shown that the problem can be reduced to the model-checking problem for a temporal logic EF and processes of $\mathcal{C}$. Since EF is decidable for PDA processes, it suffices for showing the decidability, but the obtained algorithm is not very efficient—we only obtain **EXPTIME** upper-bound in this way for both strong and weak bisimilarity. Recently, **PSPACE** lower-bound for the problem of strong (and hence also weak) bisimilarity between PDA and FS processes has been given in [May99]. A somewhat surprising result is [KM99] which says that strong and weak bisimilarity between BPA processes and finite-state ones is in **P**. OC-nets are studied, e.g., in [AČ98, JMS99] where it is shown that simulation equivalence (which is coarser than strong bisimilarity) is decidable for processes of OC-nets, and in [JKM00] where a close relationship between simulation problems for OC-nets and the corresponding bisimulation problems for OC-automata is established.

In this paper we concentrate on the complexity of checking strong and weak bisimilarity between processes of OC-automata and FS processes. Our motivation is that the specification or the implementation of a system which is to be verified (see above) can often be specified as a finite-state process. Moreover, a number of 'classical' verification problems (e.g., liveness, safety) can be easily reduced to the problem of weak bisimilarity with a finite-state system. For example, if we want to check that the action a is *live* for a process g (i.e., each state which is reachable from g can reach a state which can emit a), we can rename all actions of g except a to τ and then check weak bisimilarity between g and f where f is a one-state process with the only transition $f \stackrel{a}{\to} f$.

In Section 2, it is shown that the problem of weak bisimilarity between processes of OC-nets and FS processes is **DP**-hard, even for a fixed finite-state process (intuitively, the class **DP** [Pap94] is expected to be somewhat larger than the union of **NP** and **co-NP**; however, it is still contained in the $\Delta_2 = \mathbf{P}^{\mathbf{NP}}$ level of the polynomial hierarchy). Here we have to devise a special technique for encoding, guessing, and checking assignments of Boolean variables in the structure of OC-nets. As transition systems which are associated with OC-nets are rather regular, the method is not straightforward (observe that assignments are easy to handle with a stack; it is not so easy if there is only (one) counter at our disposal). Using the same technique we also show that strong bisimilarity between processes of OC-nets is **co-NP**-hard (strong bisimilarity between processes of OC-automata and finite-state processes is already *polynomial*—see below). Assuming the expected relationship among complexity classes, the **DP**-hardness result for weak bisimilarity actually says that any deterministic algorithm which decides the problem requires exponential time in the worst case. Rather than trying to establish **DP**-completeness, we turn our attention to a more 'practical' direction—in Section 3 we design an algorithm which decides weak bisimilarity between a process $p(i)$ of a OC-automaton $\mathcal{A}$ and a process f of a finite-state system $\mathcal{F}$ in time $\mathcal{O}(n^3\, m^5\, z^3\, (i+1))$ where n is the size of $\mathcal{A}$, m is the size of $\mathcal{F}$, and z is a special constant which depends on $\mathcal{A}$. So, if there was no z, or if z was always 'small', the problem would be in **P**. However, z *can* be much (exponentially) larger than n in general. However, it follows from the way how z is defined that the automaton must be *very* perverse to make its associated z large (a good example is the automaton constructed in the **DP**-hardness proof of Section 2). Hence, we conclude that our algorithm is actually efficient for many (if not all) practical instances, giving a sort of 'characterization' of all hard instances as a byproduct. Another advantage of our algorithm is that we can efficiently estimate the time which is needed to solve a given instance—although the computation of z for a given automaton $\mathcal{A}$ may take exponential time in general, we can efficiently (i.e., in polynomial time) compute a quite reliable bound for z. All hard instances are efficiently recognized in this way; it can also happen that some 'easy' instance is incorrectly declared as hard, but we argue that such situations are quite rare. The algorithm also works for strong bisimilarity, but in this case it only needs polynomial time—we obtain (as a simple consequence) that the problem of strong bisimilarity between OC processes and finite-state ones is in **P**.

In the next sections we use $\mathbb{N}$ and $\mathbb{N}_0$ to denote the sets of positive and non-negative integers, respectively.

2 Lower Bounds

In this section we show that the problem of weak bisimilarity between processes of OC-nets and finite-state processes is **DP**-hard (even for a *fixed* finite-state process), and that the problem of strong bisimilarity between processes of OC-nets is **co-NP**-hard.

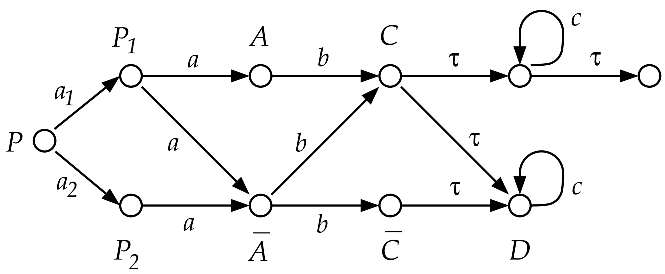


Figure 1: The finite-state system $\mathcal{F}$ used the proof of Theorem 2.1

Theorem 2.1. *The problem of weak bisimilarity between processes of one-counter nets and finite-state processes is* **DP**-*hard.*

Proof. For purposes of this proof, we first fix the finite-state system $\mathcal{F}$ of Figure 1. We show **DP**-hardness by reduction of the **DP**-complete problem SAT-UNSAT. An instance of the SAT-UNSAT problem is a pair (φ_1, φ_2) of Boolean formulae in CNF. The question is whether φ_1 is satisfiable and φ_2 unsatisfiable. First, we describe a polynomial algorithm which for a given formula φ in CNF constructs a one-counter net $\mathcal{N}_\varphi$ and its process $s_\varphi(0)$ such that φ is satisfiable iff $s_\varphi(0) \approx P_1$, and φ is unsatisfiable iff $s_\varphi(0) \approx P_2$, where P_1, P_2 are the (fixed) FS processes of the system $\mathcal{F}$. It clearly suffices for our purposes, because then we can also construct a one-counter net $\mathcal{N}$ by taking the disjoint union of $\mathcal{N}_{\varphi_1}$, $\mathcal{N}_{\varphi_2}$ and adding a new control state s together with transitions $sZ \stackrel{a_1}{\to} s_{\varphi_1}Z, sI \stackrel{a_1}{\to} s_{\varphi_1}I$ and $sZ \stackrel{a_2}{\to} s_{\varphi_2}Z, sI \stackrel{a_2}{\to} s_{\varphi_2}I$ (the non-zero transitions are added just to fulfil the constraints of the definition of OC nets). Clearly (φ_1, φ_2) is a positive instance of the SAT-UNSAT problem iff $s(0) \approx P$ where P is the fixed FS process of the system $\mathcal{F}$ (see Figure 1).

In our proof we use the following theorem of number theory (see, e.g., [BS96]): Let π_i be the i^{th} prime number, and let $f : \mathbb{N} \to \mathbb{N}$ be a function which assigns to each n the sum $\sum_{i=1}^{n} \pi_i$. Then f is $\mathcal{O}(n^3)$. (In our case, it suffices to know that the sum is asymptotically bounded by a polynomial in n.) With the help of this fact we can readily confirm that the below described construction is indeed polynomial.

Let $\varphi \equiv C_1 \wedge \cdots \wedge C_m$ be a formula in CNF where C_i are clauses over propositional variables $x_1, \cdots, x_n$. We assume (w.l.o.g.) that for every assignment $\nu : \{x_1, \cdots, x_n\} \to \{\mathtt{true}, \mathtt{false}\}$ there is at least one clause C_i such that $\nu(C_i) = \mathtt{true}$ (this can be achieved, e.g., by adding the clause $(x_1 \vee \neg x_1)$ to φ). Furthermore, we also assume that φ is not a tautology, i.e., there is at least one assignment ν such that $\nu(\varphi) = \mathtt{false}$ (it actually means that there is a clause where no variable appears both positively and negatively). The construction of $\mathcal{N}_\varphi = (Q, \{I, Z\}, \{a, b, c, \tau\}, \delta)$ will be described in a stepwise manner. The sets Q and δ are initially empty. For each clause C_i, $1 \le i \le m$, we do the following:

- We add new control states c_i and q_i to Q. Moreover, for each variable x_j and each k such that $0 \leq k < \pi_j$ we add to Q a control state $\langle C_i, x_j, k\rangle$ (observe that for each C_i we add $\mathcal{O}(n^4)$ states in this way).
- We add to δ the transition $q_i I \stackrel{c}{\rightarrow} q_i I$.
- For each $1 \leq j \leq n$ we add to δ the transitions $c_i I \stackrel{\mathcal{T}}{\rightarrow} \langle C_i, X_j, 0\rangle I$ and $c_i I \stackrel{\mathcal{T}}{\rightarrow} qI$.
- For all j, k such that $1 \leq j \leq n$ and $0 \leq k < \pi_j$ we add to δ the transition $\langle C_i, x_j, k\rangle I \stackrel{\mathcal{T}}{\rightarrow} \langle C_i, x_j, (k+1) \bmod \pi_j\rangle \varepsilon$.
- For all j, k such that $1 \leq j \leq n$ and $0 \leq k < \pi_j$ we add to δ the 'loop' $\langle C_i, x_j, k\rangle I \stackrel{c}{\rightarrow} \langle C_i, x_j, k\rangle I$.
- If a variable x_j does *not* appear positively in a clause C_i, then we add to δ the loop $\langle C_i, x_j, 0\rangle Z \stackrel{c}{\rightarrow} \langle C_i, x_j, 0\rangle Z$.
- If a variable x_j does not appear negatively in a clause C_i, then we add to δ the loops $\langle C_i, x_j, k\rangle Z \stackrel{c}{\rightarrow} \langle C_i, x_j, k\rangle Z$ for every $1 \leq k < \pi_j$.

If we draw the transition system which is generated by the current approximation of $\mathcal{N}_\varphi$, we obtain a collection of G_i graphs, $1 \leq i \leq m$; each G_i corresponds to the 'subgraph' of $\mathcal{T}_{\mathcal{N}_\varphi}$ which is obtained by restricting Q to the set of control states which have been added for the clause C_i. The structure of G_i is shown in Figure 2. (To understand this picture, observe that transition systems associated to OC-automata can be viewed as two-dimensional 'tables' where column-indexes are control states and row-indexes are counter values $(0, 1, 2, \ldots)$. As the out-going transitions of a state $q(i)$ where $i > 0$ do not depend on the actual value of i, it suffices to depict the out-going transitions at zero and (some) non-zero level.) Now we give several important facts about G_i (each fact easily follows from the previous ones):

- For each $l > 0$ we have that $c_i(l) \stackrel{\mathcal{T}}{\Rightarrow} \langle C_i, x_j, k\rangle(0)$ iff $l \bmod \pi_j = k$.
- For each $l > 0$, the state $c_i(l)$ 'encodes' the (unique) assignment ν_l defined by $\nu_l(x_j) = \texttt{true}$ iff $c_i(l) \stackrel{\mathcal{T}}{\Rightarrow} \langle C_i, x_j, 0\rangle(0)$; conversely, for each

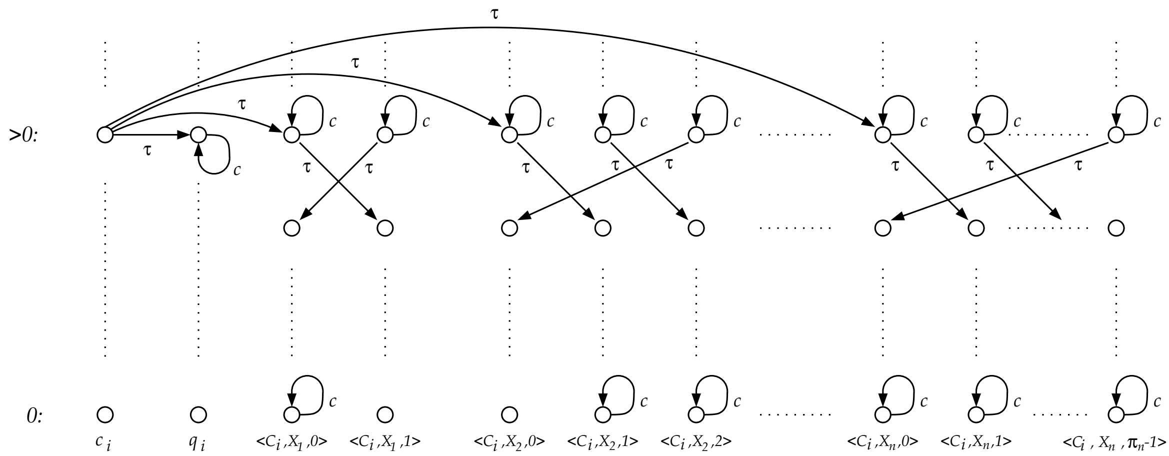


Figure 2: The structure of G_i

assignment ν there is $l \in \mathbb{N}$ such that $\nu = \nu_l$ (for example, we can put $l = \Pi_{j=0}^{n} f(j)$, where $f(j) = \pi_j$ if $\nu(x_j) = \texttt{true}$, and $f(j) = 1$ otherwise).

- For each $l > 0$ we have the following:
  - $\nu_l(C_i) = \texttt{false}$ iff $c_i(l) \approx \overline{C}$ for the state $\overline{C}$ of the system $\mathcal{F}$. Indeed, if $\nu_l(C_i) = \texttt{false}$, then $c_i(l)$ cannot reach any of the 'zero-states' where the action c is disabled—it can only emit c's (possibly with some intermediate τ's) without a possibility to terminate.
  - $\nu_l(C_i) = \texttt{true}$ (i.e., the clause C_i is true for ν_l) iff $c_i(l) \approx C$ for the state C of the system $\mathcal{F}$. This also explains the role of the control state q_i — we need it to make sure that the transition $C \xrightarrow{\tau} D$ can always be matched.

We finish the construction of $\mathcal{N}_\varphi$ by connecting the G_i components together. To do that, we add two new control states s_φ and r to Q, and enrich δ by adding the transitions $s_\varphi Z \xrightarrow{\tau} s_\varphi IZ$, $s_\varphi I \xrightarrow{\tau} s_\varphi II$, $s_\varphi I \xrightarrow{\tau} rI$, and $rI \xrightarrow{b} c_i I$ for every $1 \leq i \leq m$. The structure of $\mathcal{T}_{\mathcal{N}_\varphi}$ is shown in Figure 3. Now we can observe the following:

- For each $l > 0$ we have that

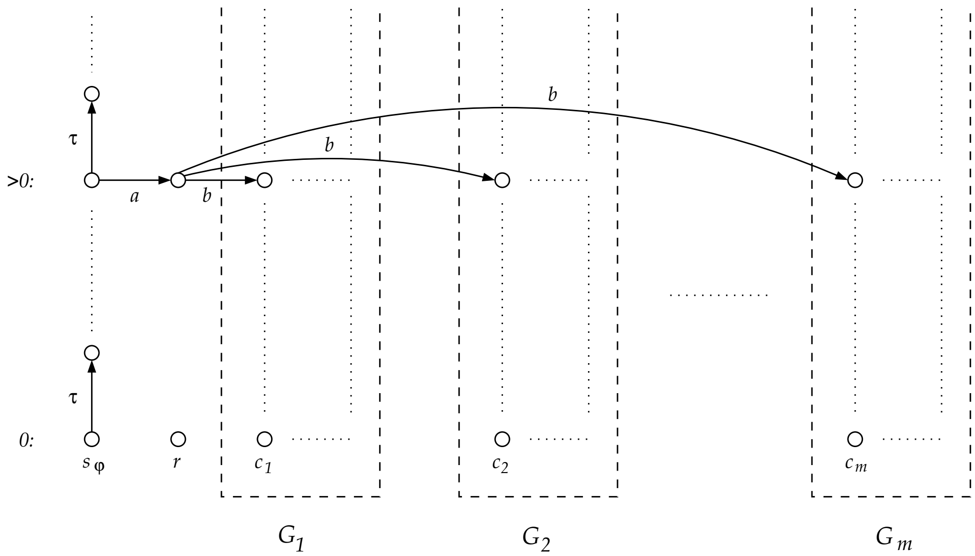


Figure 3: The structure of $\mathcal{T}_{\mathcal{N}_\varphi}$

- $\nu_l(\varphi) = \texttt{true}$ iff $r(l) \approx A$ for the state A of the system $\mathcal{F}$. To see this, realize that $\nu_l(\varphi) = \texttt{true}$ iff $\nu_l(C_i) = \texttt{true}$ for each $1 \le i \le m$ iff $c_i(l) \approx C$ for each $1 \le i \le m$ due to the previous observations.
- $\nu_l(\varphi) = \texttt{false}$ iff $r(l) \approx \overline{A}$ for the state $\overline{A}$ of the system $\mathcal{F}$. A proof is similar to the previous case; here we also need the assumption that at least one clause of φ is true for ν_l (so that we can be sure that the transition $\overline{A} \xrightarrow{b} C$ can be matched by $r(l)$).

- φ is unsatisfiable iff $s_\varphi(0) \approx P_2$ for the state P_2 of $\mathcal{F}$. Indeed, $s_\varphi(0)$ can perform its $\stackrel{a}{\Rightarrow}$ move only by going to some (arbitrary) $r(l)$. If φ is unsatisfiable, then $\nu_l(\varphi) = \texttt{false}$ for each such $r(l)$, hence all $\stackrel{a}{\Rightarrow}$ successors of $s_\varphi(0)$ are weakly bisimilar to $\overline{A}$ (see above), hence $s_\varphi(0) \approx P_2$. If φ is satisfiable, then there is a move $s_\varphi(0) \stackrel{a}{\Rightarrow} r(l)$ for some l such that $\nu_l(\varphi) = \texttt{true}$, hence $r(l) \approx A$ and $r(l) \not\approx \overline{A}$. Therefore, P_2 cannot match the $s_\varphi(0) \stackrel{a}{\Rightarrow} r(l)$ move and thus $s_\varphi(0) \not\approx P_2$.
- φ is satisfiable iff $s_\varphi(0) \approx P_1$ for the state P_1 of $\mathcal{F}$. It is checked in the same way as above. Here we use the assumption that φ is not a

tautology, i.e., $s_\varphi(0)$ can always choose an assignment which makes φ false (i.e., $s_\varphi(0)$ can always match the transition $P_1 \xrightarrow{a} \overline{A}$). □

The main reason why we could not extend the hardness result to some higher complexity class (e.g., **PSPACE**) is that there is no apparent way how to implement a 'stepwise-guessing' of Boolean variables which would allow to encode, e.g., the QBF problem; each such attempt resulted in an exponential blow-up in the number of control states. However, we can still re-use our technique to prove the following:

Theorem 2.2. *The problem of strong bisimilarity between processes of one-counter nets is* **co-NP***-hard.*

Proof. We use a similar construction as in the proof of Theorem 2.1. Given a formula φ in CNF, we construct two one-counter nets $\mathcal{N}, \overline{\mathcal{N}}$ and their processes $s(0), \overline{s}(0)$ such that φ is unsatisfiable iff $s(0) \sim \overline{s}(0)$. The net $\mathcal{N}$ is just a slight modification of the net $\mathcal{N}_\varphi$ of Theorem 2.1 — we only rename all τ-labels to c. A key observation is that φ is unsatisfiable iff after each sequence of transitions of the form c^*a (i.e., after each choice of an assignment) there is a b-transition to a state which can only emit an infinite sequence of c actions without a possibility to terminate (i.e., at least one clause is false for any assignment). The net $\overline{\mathcal{N}}$ is a 'copy' of $\mathcal{N}$ but we also add a new control state q and transitions $qI \xrightarrow{c} qI$, $\overline{r}I \xrightarrow{b} qI$ where $\overline{r}$ is a 'twin' of the state r of $\mathcal{N}_\varphi$. We put s and $\overline{s}$ to be the corresponding twins of the state s_φ of $\mathcal{N}_\varphi$. Now we can easily check that φ is unsatisfiable iff $s(0) \sim \overline{s}(0)$ — the crucial argument is stated above. □

3 Efficient Algorithms

In this section we design an algorithm which decides weak bisimilarity between processes of OC-automata and finite-state processes. As expected, the algorithm requires exponential time in the worst case. However, it works rather efficiently for many (and we believe that almost all) 'practical' instances. It also works for strong bisimilarity where it needs only polynomial time.

Let $\mathcal{T} = (S, \Sigma, \rightarrow)$ be a transition system. For each $i \in \mathbb{N}_0$ we define the relation $\approx_i$ inductively as follows: $\approx_0 = S \times S$; and $s \approx_{i+1} t$ iff $s \approx_i t$ and for each $s \stackrel{a}{\Rightarrow} s'$ there is some $t \stackrel{a}{\Rightarrow} t'$ such that $s' \approx_i t'$, and vice versa. (The $\approx_i$ relations are also used to relate states of different transition systems; formally, we consider two transition systems to be a single one by taking their disjoint union.) Our algorithm relies on the following theorem established in [JKM98]:

Theorem 3.1. *Let $\mathcal{G} = (G, \Sigma, \rightarrow)$ be a (general) transition system and $\mathcal{F} = (F, \Sigma, \rightarrow)$ a finite-state system. We say that a state $g \in G$ is* good *w.r.t. $i \in \mathbb{N}_0$ iff there is $f \in F$ such that $g \approx_i f$; g is* bad *w.r.t. i iff g is not good w.r.t. i.*

Let $g \in G$, $f \in F$, and $k = |F|$. It holds that $g \approx f$ iff $g \approx_k f$ and each state which is reachable from g is good w.r.t. k.

For the rest of this section we fix a one-counter automaton $\mathcal{A} = (Q, \{I, Z\}, \Sigma, \delta)$ of size n, and a finite-state system $\mathcal{F} = (F, \Sigma, \rightarrow)$ of size m.

To decide weak bisimilarity between processes $p(i)$ of $\mathcal{A}$ and f of $\mathcal{F}$, it suffices (by Theorem 3.1) to find out if $p(i) \approx_m f$ and whether $p(i)$ can reach a state which is bad w.r.t. m. We do that by constructing a constant z such that for each state $q(j)$ of $\mathcal{T}_\mathcal{A}$ where $j \geq (4m+1)z$ we have that $q(j) \approx_m q(j-z)$. In other words, each state of $\mathcal{T}_\mathcal{A}$ is (up to $\approx_m$) represented by another (and effectively constructible) state whose counter value is bounded by $(4m+1)z$. Then we convert this 'initial part' of $\mathcal{T}_\mathcal{A}$ to a finite-state system $\mathcal{F}_\mathcal{A}$ and construct the $\approx_m$ relation between states of $\mathcal{F}_\mathcal{A}$ and $\mathcal{F}$. The question if $p(i) \approx_m f$ is then easy to answer (we look if the representant of $p(i)$ within $\mathcal{F}_\mathcal{A}$ is related with f by $\approx_m$). The question if $p(i)$ can reach a state which is bad w.r.t. m still requires some development—we observe that states which are bad w.r.t. m are 'regularly distributed' in $\mathcal{T}_\mathcal{A}$ and construct a description of that distribution (which is 'read' from $\mathcal{F}_\mathcal{A}$) in a form of a finite-state automaton $\mathcal{M}$ which recognizes all bad states. Then we use an algorithm of [ER97] which constructs from $\mathcal{M}$ an automaton $\mathcal{M}'$ recognizing the set of all states which can reach a state recognized by $\mathcal{M}$, and look whether $\mathcal{M}'$ accepts $p(i)$. All procedures we use are polynomial in the size of $\mathcal{F}_{\mathcal{A}}$. Hence, it is only the size of z which can make the problem computationally hard. The construction of z can require exponential time; however, we give an algorithm which efficiently (i.e., in polynomial time) computes a reliable upper bound Z for z.

Intuitively, the only difference between processes $p(i), p(j)$, where $i \neq j$, is the way how they can access the 'zero level'. As long as the counter remains positive, each process can 'mimic' moves of the other process by entering the same control state and performing the same operation on the counter. Observe that the counter can be generally decremented by an unbounded value in a single $\stackrel{a}{\Rightarrow}$ step (due to an unbounded number of τ-transitions). The next definitions and lemmata reveal a crucial periodicity in the structure of $\mathcal{T}_{\mathcal{A}}$ which shows that decrementing the counter 'too much' in one $\stackrel{a}{\Rightarrow}$ step is not the thing which allows to demonstrate a possible difference between $p(i), p(j)$.

For each $l \in \mathbb{N}_0$ we define a family of binary relations $\stackrel{a}{\Rightarrow}_l$, $a \in \Sigma$, over the set of states of $\mathcal{T}_{\mathcal{A}}$ as follows: $p(i) \stackrel{a}{\Rightarrow}_l q(j)$ iff there is a sequence of transitions from $p(i)$ to $q(j)$ which forms one '$\stackrel{a}{\Rightarrow}$' move and the counter value remains greater or equal l in all states which appear in the sequence (including $p(i)$ and $q(j)$).

Definition 3.2. *We define a function $step_{\mathcal{A}} : Q \to 2^Q$ by $step_{\mathcal{A}}(p) = \{q \in Q \mid p(2) \stackrel{\tau}{\Rightarrow}_1 q(1)\}$.*

Since the reachability problem for one-counter automata (and even for pushdown automata) is in **P**, the function $step_{\mathcal{A}}$ is effectively constructible in polynomial time. As the out-going transitions of a state $p(i)$ for $i > 0$ do not depend on the actual value of i, for each $i \in \mathbb{N}$ we have that $q \in step_{\mathcal{A}}(p)$ iff $p(i+1) \stackrel{\tau}{\Rightarrow}_i q(i)$.

We extend $step_{\mathcal{A}}$ to subsets of Q by $step_{\mathcal{A}}(M) = \bigcup_{p \in M} step_{\mathcal{A}}(p)$. For each $p \in Q$ we now define the sequence C_p inductively as follows: $C_p(1) =$

$\{p\}$ and $C_p(i+1) = step_{\mathcal{A}}(C_p(i))$. The next lemma is easy to prove by a straightforward induction on i.

Lemma 3.3. *For all $p \in Q$ and $i, j \in \mathbb{N}$ we have that $q \in C_p(j)$ iff $p(i+j) \overset{\tau}{\Rightarrow}_i q(i)$.*

Another simple observation is that the sequence C_p is (for every $p \in Q$) of the form $C_p = \alpha_p \beta_p^\omega$ where α_p, β_p are finite sequences of pairwise different subsets of Q (due to the assumption that the elements of α_p and β_p are pairwise different we also have that α_p and β_p are *unique*). Note that β_p can also consist of just one element $\emptyset$. We define the *prefix* and *period* of p, denoted $pre(p)$ and $per(p)$, to be the length of α_p and β_p, respectively. Now we put

$$z = \max\{pre(p) \mid p \in Q\} \cdot \mathrm{lcm}\{per(p) \mid p \in Q\}$$

where $\mathrm{lcm}(M)$ denotes the least common multiply of elements of M. As we shall see, $\max\{pre(p) \mid p \in Q\}$ is always $\mathcal{O}(n^2)$. However, $\mathrm{lcm}\{per(p) \mid p \in Q\}$ can be *exponential* in n (for example, examine the net $\mathcal{N}_\varphi$ constructed in the proof of Theorem 2.1). As we already mentioned, the size of z is the only thing which can make the considered problem hard. Hence, we obtain a kind of 'characterization' of all hard instances—OC-automata which are presented in hard instances must contain many 'decreasing τ-cycles' of an incomparable length. Also observe that the construction of z can require exponential time, because $per(p)$ for a given p can be exponential in n (in the end of this section we show how to compute a reasonable upper bound Z for z efficiently). The following lemma is immediate:

Lemma 3.4. *For all $p \in Q$ and $i \geq z$ we have that $C_p(i) = C_p(i+z)$.*

The next three lemmata provide a crucial observation about the structure of $\mathcal{T}_{\mathcal{A}}$ and precisely formulate the intuition that 'decreasing the counter too much in one $\overset{a}{\Rightarrow}$ step does not help'.

Lemma 3.5. *For all $p \in Q$ and $j \in \mathbb{N}$ it holds that*

- *if there is a sequence of τ-transitions from $p(j+2z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+z) \overset{\tau}{\Rightarrow} q(l)$;*

- *if there is a sequence of τ-transitions from $p(j+z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+2z) \stackrel{\tau}{\Rightarrow} q(l)$;*

Proof. We show only the first part (the other one is similar). As there is a sequence of τ-transitions from $p(j+2z)$ to $q(l)$ which decreases the counter to j, there must be an intermediate state $r(j)$ such that $p(j+2z) \stackrel{\tau}{\Rightarrow}_j r(j) \stackrel{\tau}{\Rightarrow} q(l)$. As $p(j+2z) \stackrel{\tau}{\Rightarrow}_j r(j)$, we obtain that $r \in C_p(2z)$ due to Lemma 3.3, hence also $r \in C_p(z)$ by Lemma 3.4. From this we have (again by Lemma 3.3) that $p(j+z) \stackrel{\tau}{\Rightarrow}_j r(j)$, hence $p(j+z) \stackrel{\tau}{\Rightarrow} q(l)$ as required. □

Lemma 3.6. *For all $p \in Q$ and $j \in \mathbb{N}$ it holds that*

- *if there is a sequence of transitions forming one '$\stackrel{a}{\Rightarrow}$' move from $p(j+4z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+3z) \stackrel{a}{\Rightarrow} q(l)$;*

- *if there is a sequence of transitions forming one '$\stackrel{a}{\Rightarrow}$' move from $p(j+3z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+4z) \stackrel{a}{\Rightarrow} q(l)$;*

Proof. Again, we only show the first part. The case when $a = \tau$ has been handled in Lemma 3.5. If $a \neq \tau$, we can distinguish two cases:

- The counter is decreased to $j+2z$ at some point in the considered sequence of transitions before emitting the action a. Then there is a state $r(j+2z)$ such that $p(j+4z) \stackrel{\tau}{\Rightarrow} r(j+2z) \stackrel{a}{\Rightarrow} q(l)$. Now we can apply Lemma 3.5 and conclude that $p(j+3z) \stackrel{\tau}{\Rightarrow} r(j+2z)$ which suffices.

- The action a is emitted before decreasing the counter to $j+2z$. Then there is a state $r(j+2z)$ such that $p(j+4z) \stackrel{a}{\Rightarrow}_{j+2z}$ and $r(j+2z)$ can enter the state $q(l)$ in a sequence of τ-transitions which decreases the counter to j. Hence, by Lemma 3.5 we know that $r(j+z) \stackrel{\tau}{\Rightarrow} q(l)$. Now it suffices to realize that since $p(j+4z) \stackrel{a}{\Rightarrow}_{j+2z} r(j+2z)$, we also have $p(j+3z) \stackrel{a}{\Rightarrow}_{j+z} r(j+z)$. To sum up, $p(j+3z) \stackrel{a}{\Rightarrow}_{j+z} r(j+z) \stackrel{\tau}{\Rightarrow} q(l)$ and we are done. □

Lemma 3.7. *Let $p \in Q$ and $k \in \mathbb{N}_0$. For each $c > (4k+1)z$ we have that $p(c) \approx_k p(c-z)$.*

Proof. By induction on k. The base case ($k = 0$) is immediate. Now let $c > (4(k+1)+1)z$. We prove that for each '$\stackrel{a}{\Rightarrow}$' move of $p(c)$ there is a '$\stackrel{a}{\Rightarrow}$' move of $p(c-z)$ such that the resulting pair of states is related by $\approx_k$, and vice versa. Let $p(c) \stackrel{a}{\Rightarrow} q(l)$. We distinguish two cases:

- $p(c) \stackrel{a}{\Rightarrow}_{c-4z+1} q(l)$. It means that $l \geq c - 4z + 1$, hence $l > 4(k+1)z$. Furthermore, we have $p(c-z) \stackrel{a}{\Rightarrow}_{c-5z+1} q(l-z)$ (this move is possible because $c > 5z$). Now $q(l) \approx_k q(l-z)$ by induction hypotheses.

- The counter is decreased to $c - 4z$ by the considered sequence of transitions. Then $p(c-z) \stackrel{a}{\Rightarrow} q(l)$ by Lemma 3.6. Clearly $q(l) \approx_k q(l)$.

The other direction is shown in a similar way. $\square$

Now we are almost in a position to prove the first main theorem of this section. It remains to extend our equipment with the following tool:

Definition 3.8. *Let $\mathcal{P} = (Q, \Gamma, \Sigma, \delta)$ be a pushdown automaton, $\mathcal{M} = (S, \Gamma, \gamma, F)$ a nondeterministic finite-state automaton (note that the input alphabet of $\mathcal{M}$ is the stack alphabet of $\mathcal{P}$), and $Init : Q \to S$ a total function. A process $p\alpha$ of $\mathcal{P}$ is* recognized *by the pair $(\mathcal{M}, Init)$ iff $\hat{\gamma}(Init(p), \alpha) \cap F \neq \emptyset$ where $\hat{\gamma}$ is the natural extension of γ to elements of $S \times \Gamma^*$.*

The next theorem is taken from [ER97].

Theorem 3.9. *Let $\mathcal{P} = (Q, \Gamma, \Sigma, \delta)$ be a pushdown automaton, $\mathcal{M} = (S, \Gamma, \gamma, F)$ a finite-state automaton, and $Init : Q \to S$ a total function. Let N be the set of processes recognized by $(\mathcal{M}, Init)$. Then one can effectively construct an automaton $\mathcal{M}' = (S, \Gamma, \gamma', F)$ in time $\mathcal{O}(|\delta| \cdot |S|^3)$ such that $(\mathcal{M}', Init)$ recognizes the set*

$$Pre^*(N) = \{q\beta \mid q\beta \to^* p\alpha \text{ for some } p\alpha \in N\}$$

of all predecessors of N.

Theorem 3.10. *The problem of weak bisimilarity between processes $p(i)$ of $\mathcal{A}$ and f of $\mathcal{F}$ is decidable in $\mathcal{O}(n^3\, m^5\, z^3\, (i+1))$ time.*[1]

Proof. By Theorem 3.1, we need to find out whether $p(i) \approx_m f$ and whether $p(i)$ can reach a state which is bad w.r.t. m. Due to Lemma 3.7 we know that the set of all states of $\mathcal{T}_{\mathcal{A}}$ up to $\approx_m$ can be represented by the subset of states of $\mathcal{T}_{\mathcal{A}}$ where the counter value is at most $(4m+1)z$. Formally, we first define the function $\mathcal{B} : (Q \times \mathbb{N}_0) \to (Q \times \mathbb{N}_0)$ as follows (where $\langle q, j\rangle$ is just another notation for $q(j)$):

$$\mathcal{B}(\langle q, j\rangle) = \begin{cases} \langle q, j\rangle & \text{if } j \leq (4m+1)z; \\ \langle q, (4m+1)z\rangle & \text{if } j > (4m+1)z \text{ and } (j \bmod z) = 0; \\ \langle q, 4mz + (j \bmod z)\rangle & \text{if } j > (4m+1)z \text{ and } (j \bmod z) \neq 0. \end{cases}$$

An immediate consequence of Lemma 3.7 is that for all $q \in Q$ and $j \in \mathbb{N}_0$ we have $q(j) \approx_m \mathcal{B}(q(j))$. Now we define a finite-state system $\mathcal{F}_{\mathcal{A}} = (F_{\mathcal{A}}, \Sigma, \hookrightarrow)$ where $F_{\mathcal{A}}$ is the image of $\mathcal{B}$ (i.e., $F_{\mathcal{A}} = \{q(j) \mid q \in Q, 0 \leq j \leq (4m+1)z\}$), Σ is the set of actions of $\mathcal{A}$, and $\hookrightarrow$ is the least relation satisfying the following: if $r(k) \xrightarrow{a} s(l)$ is a transition of $\mathcal{T}_{\mathcal{A}}$, then $\mathcal{B}(r(k)) \overset{a}{\hookrightarrow} \mathcal{B}(s(l))$. Observe that $\mathcal{F}_{\mathcal{A}}$ is actually the 'initial part' of $\mathcal{T}_{\mathcal{A}}$; the only difference is that all up-going transitions of states at level $(4m+1)z$ are 'bent' down to the corresponding $\approx_m$-equivalent states at level $4mz+1$. Note that for each $q(j)$ we still have that $q(j) \approx_m \mathcal{B}(q(j))$ (when $\mathcal{B}(q(j))$ is seen as a state of $\mathcal{F}_{\mathcal{A}}$). The number of states of $\mathcal{F}_{\mathcal{A}}$ is $\mathcal{O}(n\, m\, z)$; moreover, the number of out-going transitions at each 'level' of $\mathcal{T}_{\mathcal{A}}$ is $\mathcal{O}(n)$, hence the size of $\hookrightarrow$ is $\mathcal{O}(n\, m\, z)$, which means that the total size of $\mathcal{F}_{\mathcal{A}}$ is also $\mathcal{O}(n\, m\, z)$.

Now, let us realize that if we have a finite-state system of size t, it takes $\mathcal{O}(t^3)$ time to compute the associated '$\overset{a}{\Rightarrow}$' relation (for each state s and action a we need $\mathcal{O}(t)$ time to compute the set $\{r \mid s \overset{a}{\Rightarrow} r\}$). Therefore, we need $\mathcal{O}(n^3\, m^3\, z^3)$ time to construct the extended transition relations for $\mathcal{F}_{\mathcal{A}}$ and $\mathcal{F}$. To compute the $\approx_m$ relation between the states of $\mathcal{F}_{\mathcal{A}}$ and $\mathcal{F}$, we define $\mathcal{R}^0 = F_{\mathcal{A}} \times F$, and $\mathcal{R}^{i+1} = Exp(\mathcal{R}^i)$ where the function $Exp : (F_{\mathcal{A}} \times F) \to$

[1] Note that we need a non-constant time even in the particular case when $i = 0$ (the problem is still **DP**-hard). That is why we write '$i+1$'.

$(F_{\mathcal{A}} \times F)$ refines its argument according to the definition of $\approx_i$ — a pair $(r(j), g)$ belongs to $Exp(\mathcal{R})$ iff it belongs to $\mathcal{R}$ and for each '$\stackrel{a}{\Rightarrow}$' move of one component there is a corresponding '$\stackrel{a}{\Rightarrow}$' move of the other component such that the resulting pair of states belongs to $\mathcal{R}$. Clearly, for each pair $(r(j), g)$ of $F_{\mathcal{A}} \times F$ we have that $r(j) \approx_m g$ iff $(r(j), g) \in \mathcal{R}^m$. It remains to clarify the time costs. The function Exp is computed m times. Each time, $\mathcal{O}(n\, m^2\, z)$ pairs are examined. For each such pair we have to check the membership to $Exp(\mathcal{R})$. This takes only $\mathcal{O}(n\, m^2\, z)$ time, because the extended transition relations have already been computed. To sum up, we need $\mathcal{O}(n^3\, m^5\, z^3)$ time in total.

To check if $p(i) \approx_m f$, we simply look if $(\mathcal{B}(p(i), f)) \in \mathcal{R}^m$. It remains to find out whether $p(i)$ can reach a state $q(j)$ which is bad w.r.t. m. Observe that $q(j)$ is bad w.r.t. m iff the state $\mathcal{B}(q(j))$ of $\mathcal{F}_{\mathcal{A}}$ is bad w.r.t. m. Therefore, we can easily construct a finite-state automaton $\mathcal{M}$ and a function $Init$ such that the pair $(\mathcal{M}, Init)$ recognizes the set of all bad states of $\mathcal{T}_{\mathcal{A}}$ — we put $\mathcal{M} = (S, \{I, Z\}, \gamma, \{fin\})$ where $S = \{fin\} \cup \{p(i) \mid p \in Q, 0 \leq i \leq (4m+1)z\}$ and γ is the least transition function satisfying the following:

- $p(i+1) \in \gamma(p(i), I)$ for all $p \in Q$, $0 \leq i < (4m+1)z$;
- $p(4mz+1) \in \gamma(p((4m+1)z), I)$ for each $p \in Q$;
- if a state $p(i)$ of $\mathcal{F}_{\mathcal{A}}$ is bad, then $fin \in \gamma(p(i), Z)$.

The function $Init$ is defined by $Init(p) = p(0)$ for all $p \in Q$. Note that $\mathcal{M}$ has $\mathcal{O}(n\, m\, z)$ states. Now we compute the automaton $\mathcal{M}'$ of Theorem 3.9 (it takes $\mathcal{O}(n^2\, m\, z)$ time) and check if $(\mathcal{M}', Init)$ recognizes $p(i)$. This can be done in $\mathcal{O}(n\, m\, z\, (i+1))$ time because $\mathcal{M}'$ has the same set of states as $\mathcal{M}$.

We see that $\mathcal{O}(n^3\, m^5\, z^3\, (i+1))$ time suffices for all of the aforementioned procedures. $\square$

Our algorithm also works for strong bisimilarity in the following way: If we are to decide strong bisimilarity between $p(i)$ and f, we first rename all τ-transitions of $\mathcal{A}$ and $\mathcal{F}$ with some (fresh) action e (it does not change anything from the point of view of strong bisimilarity, because here the τ-transitions are treated as 'ordinary' ones). As there are no τ-transitions anymore, there is no difference between strong and weak bisimilarity, hence we can use the designed algorithm. Also observe that if there are no τ's then $z = 1$, so we can conclude:

Corollary 3.11. *The problem of strong bisimilarity between processes $p(i)$ of $\mathcal{A}$ and f of $\mathcal{F}$ is in* $\mathbf{P}$.

As we already mentioned, the construction of z can take exponential time. Now we show how to compute a rather tight upper bound Z for z in polynomial time.

Lemma 3.12. *Let $p \in Q$. Let α, β be finite sequences of subsets of Q such that $C_p = \alpha\beta^\omega$. Then $pre(p) \leq length(\alpha)$ and $per(p)$ divides $length(\beta)$.*

Proof. Let α_p, β_p be the unique sequences such that $pre(p) = length(\alpha_p)$, $per(p) = length(\beta_p)$, and $C_p = \alpha_p\beta_p^\omega$. First we show that $length(\beta_p)$ divides $length(\beta)$. Suppose the converse. Let $\overline{\alpha}$ be the first $length(\alpha_p) \cdot length(\beta_p) \cdot length(\alpha) \cdot length(\beta)$ elements of C_p. We (immediately) have that $C_p = \overline{\alpha}\beta_p^\omega = \overline{\alpha}\beta^\omega$, hence $\beta_p^\omega = \beta^\omega$. Let S be the first element of β_p (and β). As the elements of β_p are by definition pairwise different, it holds that the i^{th} element of $\beta_p^\omega = \beta^\omega$ is S iff $(i \bmod length(\beta_p)) = 1$. Now let $i = length(\beta) + 1$. As the first element of β is S, the i^{th} element of $\beta^\omega = \beta_p^\omega$ is also S. However, $(i \bmod length(\beta_p)) \neq 1$ because $length(\beta_p)$ does not divide $length(\beta)$ and we have a contradiction.

As $\beta_p^\omega = \beta^\omega$ and $length(\beta_p)$ divides $length(\beta)$, we can also conclude that $\beta = \beta_p^j$ where $j = length(\beta)/length(\beta_p)$. Hence, $C_p = \alpha\beta^\omega = \alpha\beta_p^\omega = \alpha_p\beta_p^\omega$. Therefore $length(\alpha_p) \leq length(\alpha)$ because α_p is clearly the minimal sequence γ with the property $C_p = \gamma\beta_p^\omega$. □

Definition 3.13. *In our next proofs we often need to study properties of the sequence C_p in detail. To do that, for each $p \in Q$ we define the tree T_p as follows:*

- *the root of T_p is (labelled by) p;*

- *each node g of T_p labelled by (some) q has $|step_{\mathcal{A}}(q)|$ successors which are labelled (in one-to-one fashion) by elements of $step_{\mathcal{A}}(q)$.*

For each node g of T_p we define its Distance, *denoted $Dist(g)$, to be the distance of g from the root of T_p plus 1. Moreover, for each $i \in \mathbb{N}$ we define the set $Nodes^i_p$ of all nodes with the same Distance i. Observe that for each $i \in \mathbb{N}$ we have that $C_p(i) = \bigcup_{g \in Nodes^i_p} label(g)$.*

Lemma 3.14. *Let $p \in Q$ be a self-embedding control state. Then $pre(p) \leq |Q|^2$ and $per(p) \leq |Q|$. Moreover, $pre(p)$ and $per(p)$ can be computed in $\mathcal{O}(|Q|^4)$ (and hence also $\mathcal{O}(n^4)$) time.*

Proof. As p is self-embedding, there is a node g of T_p with minimal Distance i such that $i > 2$ and $label(g) = p$. First we prove that $i \leq |Q|$. Suppose the converse, i.e., $i > |Q|$. Then the path from the root of T_p to g passes through two different nodes g_1, g_2 such that $Dist(g_1) < Dist(g_2) \leq |Q|$ and $label(g_1) = label(g_2) = q$ for some $q \in Q$. As g is a descendant of g_2, we have that $p \in C_q(j+1)$ where $j = Dist(g) - Dist(g_2)$. Hence, we also have that $p \in C_p(Dist(g_1) + j)$. As $Dist(g_1) + j < Dist(g)$, we obtain a contradiction—and therefore we can conclude that i cannot be greater then $|Q|$.

Now realize that if there are (some) $j, k \in \mathbb{N}$ such that $C_p(j) \subseteq C_p(k)$, then also $C_p(j') \subseteq C_p(j' + (k-j))$ for any $j' \geq j$. If follows easily from the definition of C_p. For the same reason we have that if $C_p(j) = C_p(k)$, then $C_p(j') = C_p(j' + (k-j))$ for any $j' \geq j$. As $\{p\} = C_p(1) \subseteq C_p(i)$, we see (due to the previous observation) that

$$C_p(1) \subseteq C_p(i) \subseteq C_p(2i-1) \subseteq C_p(3i-2) \subseteq \cdots \subseteq C_p(|Q|i - |Q| + 1).$$

As the length of this increasing sequence is $|Q| + 1$, the last two elements (i.e., the elements $C_p((|Q|-1)i - |Q| + 2)$ and $C_p(|Q|i - |Q| + 1)$) must be equal. Hence, if we define α to be the first $(|Q|-1)i - |Q| + 1$ elements of C_p, and β the next $i-1$ elements of C_p, we obtain that $C_p = \alpha\beta^\omega$. As i is bounded by $|Q|$, we have that $length(\alpha)$ is bounded by $|Q|^2$ and $length(\beta)$ is bounded by $|Q|$. Those bounds are also valid for $pre(p)$ and $per(p)$ by Lemma 3.12. Hence,

$per(p)$ and $pre(p)$ can be computed simply by constructing the first $|Q|^2+|Q|$ elements of C_p, which can be (comfortably) done in time $\mathcal{O}(|Q|^4)$. $\square$

Theorem 3.15. *Let us define*

$$Z = (|Q|^2 + |Q|) \cdot \mathrm{lcm}\{per(p) \mid p \in Q \textit{ is self-embedding } \}$$

Then Z can be computed in time which is polynomial in n. Moreover, $z \leq Z$.

Proof. The fact that Z can be computed in polynomial time is obvious—the only potential problem might be the computation of $\mathrm{lcm}\{per(p) \mid p \in Q \text{ is self-embedding }\}$; however, by Lemma 3.14 we know that we actually compute the least common multiply of $\mathcal{O}(n)$ numbers whose size is $\mathcal{O}(n)$, and it can be comfortably done in time which is polynomial in n. Now we prove that $z \leq Z$. To do that, it suffices to show that for each $p \in Q$ we have that $pre(p) \leq |Q|^2 + |Q|$ and $per(p)$ divides $\mathrm{lcm}\{per(p) \mid p \in Q \text{ is self-embedding }\}$. If p is self-embedding, both things are obvious. Now let $p \in Q$ be a non-self-embedding control state. To demonstrate that $pre(p)$ and $per(p)$ satisfy the above mentioned conditions, the structure of the tree T_p must be examined in a greater detail. A simple observation is that (the label of) each node whose Distance is $|Q| + 1$ is either self-embedding or it is a descendant of a self-embedding node; hence, for each $g \in Nodes_p^{|Q|+1}$ we can define its *self-embedding ancestor*, denoted $\mathcal{E}(g)$, to be the label of the (unique) node with minimal Distance lying on the path from the root to g whose label is a self-embedding control state. For each $j \geq |Q| + 1$ it holds that

$$C_p(j) = \bigcup_{g \in Nodes_p^{|Q|+1}} C_{\mathcal{E}(g)}(j - Dist(\mathcal{E}(g)) + 1)$$

We define α to be the first $|Q|^2 + |Q|$ elements of C_p, and β the next

$$\mathrm{lcm}\{per(\mathcal{E}(g)) \mid g \in Nodes_p^{|Q|+1}\}$$

elements of C_p; as $pre(r) \leq |Q|^2$ for each self-embedding control state r (due to Lemma 3.14), we see that $C_p = \alpha\beta^\omega$. Now we can apply Lemma 3.12 to conclude that $pre(p) \leq length(\alpha) = |Q|^2 + |Q|$ and $per(p)$ divides $length(\beta)$ which divides $\mathrm{lcm}\{per(q) \mid q \in Q$ is self-embedding $\}$. $\square$

References

[AČ98] P.A. Abdulla and K. Čerāns. Simulation is decidable for one-counter nets. In *Proceedings of CONCUR'98*, volume 1466 of *Lecture Notes in Computer Science*, pages 253–268. Springer, 1998.

[BS96] E. Bach and J. Shallit. *Algorithmic Number Theory. Vol. 1, Efficient Algorithms*. The MIT Press, 1996.

[CHS95] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.

[ER97] J. Esparza and P. Rossmanith. An automata approach to some problems on context-free grammars. In *Foundations of Computer Science, Potential – Theory – Cognition*, volume 1337 of *Lecture Notes in Computer Science*, pages 143–152. Springer, 1997.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Jan97] P. Jančar. Bisimulation equivalence is decidable for one-counter processes. In *Proceedings of ICALP'97*, volume 1256 of *Lecture Notes in Computer Science*, pages 549–559. Springer, 1997.

[JKM98] P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. In *Proceedings of ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 1998.

[JKM00] P. Jančar, A. Kučera, and F. Moller. Simulation and bisimulation over one-counter processes. In *Proceedings of STACS 2000*, volume

1770 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 2000.

[JMS99] P. Jančar, F. Moller, and Z. Sawa. Simulation problems for one-counter machines. In *Proceedings of SOFSEM'99*, volume 1725 of *Lecture Notes in Computer Science*, pages 404–413. Springer, 1999.

[KM99] A. Kučera and R. Mayr. Weak bisimilarity with infinite-state systems can be decided in polynomial time. In *Proceedings of CONCUR'99*, volume 1664 of *Lecture Notes in Computer Science*, pages 368–382. Springer, 1999.

[May99] R. Mayr. *Private communication*. 1999.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Pap94] Ch. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[Rei85] W. Reisig. *Petri Nets—An Introduction*. Springer, 1985.

[Sén98] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proceedings of 39th Annual Symposium on Foundations of Computer Science*, pages 120–129. IEEE Computer Society Press, 1998.

[vG90] R.J. van Glabbeek. The linear time—branching time spectrum. In *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.

Publications in the FI MU Report Series are in general accessible
via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic