On Simulation-Checking with Sequential Systems

Antonín Kučera[⋆]

Faculty of Informatics, Masaryk University, Czech Republic (tony@fi.muni.cz)

Abstract. We present new complexity results for simulation-checking and model-checking with infinite-state systems generated by pushdown automata and their proper subclasses of one-counter automata and one-counter nets (one-counter nets are 'weak' one-counter automata computationally equivalent to Petri nets with at most one unbounded place).

As for simulation-checking, we show the following: a) simulation equivalence between pushdown processes and finite-state processes is **EXPTIME**-complete; b) simulation equivalence between processes of one-counter automata and finite-state processes is **coNP**-hard; c) simulation equivalence between processes of one-counter nets and finite-state processes is in **P** (to the best of our knowledge, it is the first (and rather tight) polynomiality result for simulation with infinite-state processes).

As for model-checking, we prove that a) the problem of simulation-checking between processes of pushdown automata (or one-counter automata, or one-counter nets) and finite-state processes are polynomially reducible to the model-checking problem with a fixed formula $\varphi \equiv \nu X.[z]\langle z\rangle X$ of the modal μ-calculus. Consequently, model-checking with φ is **EXPTIME**-complete for pushdown processes and **coNP**-hard for processes of one-counter automata; b) model-checking with a fixed formula $\Diamond[a]\Diamond[b]\mathtt{ff}$ of the logic EF (a simple fragment of CTL) is **NP**-hard for processes of OC nets, and model-checking with another fixed formula $\Box\langle a\rangle\Box\langle b\rangle\mathtt{tt}$ of EF is **coNP**-hard. Consequently, model-checking with any temporal logic which can express these simple formulae is computationally hard even for the (very simple) sequential processes of OC-nets.

1 Introduction

Two important approaches to formal verification of concurrent systems are *equivalence-checking* and *model-checking*. In both cases, a process is formally understood to be (associated with) a state in a *transition system*, which is a triple $\mathcal{T} = (S, Act, \rightarrow)$ where S is a set of *states*, Act is a finite set of *actions*, and $\rightarrow \subseteq S \times Act \times S$ is a *transition relation*. We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$ and we extend this notation to elements of Act^* in the natural way. A state t is *reachable* from a state s, written $s \rightarrow^* t$, iff $s \xrightarrow{w} t$ for some $w \in Act^*$.

In the equivalence-checking approach, one describes the *specification* (the intended behavior) and the actual *implementation* of a concurrent process as states in transition systems, and then it is shown that they are *equivalent*. Here the notion of equivalence can be formalized in various ways according to specific needs of a given practical

[⋆] Supported by the Grant Agency of the Czech Republic, grants No. 201/98/P046 and No. 201/00/1023.

problem (see, e.g., [24] for an overview). A favorite approach is the one of *simulation* equivalence which has been found appropriate in many situations and consequently its accompanying theory has been developed very intensively. Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. A binary relation $R \subseteq S \times S$ is a *simulation* iff whenever $(s,t) \in R$, then for each $s \xrightarrow{a} s'$ there is some $t \xrightarrow{a} t'$ such that $(s', t') \in R$. A process s is *simulated* by t, written $s \sqsubseteq_s t$, iff there is a simulation R such that $(s,t) \in R$. Processes s, t are *simulation equivalent*, written $s =_s t$, iff they can simulate each other. Simulation can also be viewed as a *game* — imagine there are two tokens put on states s and t. Now two players, Al and Ex, start to play a *simulation game* which consists of a (possibly infinite) number of *rounds* where each round is performed as follows: Al takes the token which was put on s originally and moves it along a transition labelled by (some) a; the task of Ex is to move the other token along a transition with the same label. Al wins the game iff after a finite number of rounds Ex cannot respond to Al's final attack. We see that $s \sqsubseteq_s t$ iff Ex has a universal defending strategy, i.e., Al never wins provided Ex plays in a sufficiently 'clever' way. We use simulation game as some points to give a more intuitive justification for our claims. Finally, let us note that simulation can also be used to relate states of *different* transition systems; formally, two systems are considered to be a single one by taking their disjoint union.

In the model-checking approach, desired properties of the implementation are encoded as formulae of certain temporal logic (interpreted over transition systems) and then it is demonstrated that the implementation satisfies the formulae. There are many systems of temporal logic differing in their expressive power, decidability, complexity, and other aspects (see, e.g., [23, 6]). In this paper we only work with one (fixed) formula $\varphi \equiv \nu X.[z]\langle z \rangle X$ of the *modal μ-calculus* [13] and some other (fixed) formulae of its very simple fragment which is known as the *EF logic* (the logic EF can also be seen as a natural fragment of CTL [6]). A formal definition of the syntax and semantics of the modal μ-calculus is omitted due to space constraints (we refer, e.g., to [13]). However, we do explain the meaning of φ in Section 3. Formulae of the logic EF look as follows:

$$\psi ::= \texttt{tt} \mid \psi \wedge \psi \mid \neg\psi \mid \langle a \rangle \psi \mid \Diamond \psi$$

Here a ranges over a given set of atomic actions. Dual operators to $\langle a \rangle$ and $\Diamond$ are $[a]$ and $\Box$, defined by $[a]\psi \equiv \neg\langle a \rangle \neg\psi$ and $\Box\psi \equiv \neg\Diamond\neg\psi$, respectively. Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. The *denotation* $[\![\psi]\!]$ of a formula ψ is the set of states where the formula *holds*; it is defined as follows:

$$\begin{aligned}
[\![\texttt{tt}]\!] &= S \\
[\![\psi_1 \wedge \psi_2]\!] &= [\![\psi_1]\!] \cap [\![\psi_2]\!] \\
[\![\neg\psi]\!] &= S - [\![\psi]\!] \\
[\![\langle a \rangle \psi]\!] &= \{s \in S \mid \exists t \in S : s \xrightarrow{a} t \wedge t \in [\![\psi]\!]\} \\
[\![\Diamond\psi]\!] &= \{s \in S \mid \exists t \in S : s \rightarrow^* t \wedge t \in [\![\psi]\!]\}
\end{aligned}$$

The 'language' of transition systems is not very practical – concurrent systems often have a very large (or even infinite) state-space and hence it is not feasible to define their semantics 'directly' by means of transition systems. Therefore, 'higher' languages allowing to construct compact definitions of large systems have been proposed and studied. In this paper we mainly work with (subclasses of) pushdown automata, which are considered as a fundamental model of sequential behaviors in the framework of concurrency theory (for example, one can conveniently model programs consisting of mutually recursive procedures in the syntax of PDA, and existing verification techniques for PDA are then applicable to, e.g., some problems of data-flow analysis [7]). Formally, a *pushdown automaton* is a tuple $\Delta = (Q, \Gamma, Act, \delta)$ where Q is a finite set of *control states*, Γ is a finite *stack alphabet*, Act is a finite *input alphabet*, and $\delta : (Q \times \Gamma) \to 2^{Act \times (Q \times \Gamma^*)}$ 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). In the rest of this paper we adopt a more intuitive notation, writing $pA \xrightarrow{a} q\beta \in \delta$ instead of $(a, (q, \beta)) \in \delta(p, A)$. To Δ we associate the transition system $\mathcal{T}_\Delta$ where $Q \times \Gamma^*$ is the set of states (we write $p\alpha$ instead of (p, α)), Act is the set of actions, and the transition relation is determined by $pA\alpha \xrightarrow{a} q\beta\alpha \iff pA \xrightarrow{a} q\beta \in \delta$.

A natural and important subclass of pushdown automata is the class of *one-counter* 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 Δ is a union of functions δ_Z and δ_I where $\delta_Z : (Q \times \{Z\}) \to 2^{Act \times (Q \times (\{I\}^* \{Z\}))}$ and $\delta_I : (Q \times \{I\}) \to 2^{Act \times (Q \times \{I\}^*)}$. Hence, Z works like a bottom symbol (which cannot be removed), and the number of pushed I's represents the counter value. Processes of $\mathcal{A}$ (i.e., states of $\mathcal{T}_\Delta$) are of the form pI^iZ which is abbreviated to $p(i)$ in the rest of this paper. 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 [22] with (at most) one unbounded place. Hence, one-counter nets can be used, e.g., to model systems consisting of producers and consumers which share an infinite buffer (a non-empty buffer enables the execution of consumers but it need not be tested for zero explicitly). Formally, a *one-counter net* $\mathcal{N}$ is a one-counter automaton such that whenever $pZ \xrightarrow{a} qI^iZ \in \delta$, then $pI \xrightarrow{a} qI^{i+1} \in \delta$. 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’.

The state of the art: Let **PDA**, **BPA**, **OC-A**, **OC-N**, and **FS** be the classes of all processes of pushdown automata, stateless pushdown automata, one-counter automata, one-counter nets, and finite-state systems, respectively. Moreover, let **PN**, **BPP**, and **PA** denote the classes of all processes of Petri nets [22], basic parallel processes [5], and process algebra [4], respectively. The problems of simulation preorder and simulation equivalence between processes of classes **A** and **B** are denoted by **A** $\sqsubseteq_s$ **B** and **A** $=_s$ **B**, respectively. The problem of simulation-checking with (certain classes of) infinite-state systems has been attracting attention for almost a decade; here we only mention some of the most relevant results. First, it was shown in [8] that the problems **BPA** $\sqsubseteq_s$ **BPA**

and **BPA** $=_s$ **BPA** are undecidable. The undecidability of **BPP** $\sqsubseteq_s$ **BPP** and **BPP** $=_s$ **BPP** was proved in [9]. An interesting positive result is [1] where it is shown that **OC-N** $\sqsubseteq_s$ **OC-N** (and hence also **OC-N** $=_s$ **OC-N**) is decidable. However, **OC-A** $\sqsubseteq_s$ **OC-A** and **OC-A** $=_s$ **OC-A** are already undecidable [12]. The problem of checking simulation between infinite and finite-state systems was first examined in [11] where it is shown that **PN** $\sqsubseteq_s$ **FS**, **FS** $\sqsubseteq_s$ **PN**, and **PN** $=_s$ **FS** are decidable. A similar positive result was later demonstrated in [16] for the **PDA** $\sqsubseteq_s$ **FS**, **FS** $\sqsubseteq_s$ **PDA**, and **PDA** $=_s$ **FS** problems; some complexity estimation were also given (see below). Moreover, the problems **PA** $\sqsubseteq_s$ **FS**, **FS** $\sqsubseteq_s$ **PA**, and **PA** $=_s$ **FS** are proved to be undecidable.

The decidability and complexity of checking other behavioral equivalences (in particular, *strong* and *weak bisimilarity* [21, 20]) between infinite and finite state systems also exist; we give a short comparison in the final section.

Our contribution: In our paper we present new complexity results for simulation-checking and model-checking problems with the above mentioned subclasses of pushdown processes. The most significant original contributions are summarized below together with a short discussion on previous work.

- **PDA** $=_s$ **FS** is **EXPTIME**-complete. Previously, there was a **coNP** lower bound for the problem [16] (this lower bound also works for **BPA** processes). In the same paper, the membership of **PDA** $=_s$ **FS** to **EXPTIME** has also been shown, hence here we only need to prove the **EXPTIME** lower bound.
- **OC-A** $=_s$ **FS** is **coNP**-hard. The problem whether this lower bound is tight is left open. Intuitively, the problem should be expected easier then for PDA processes, because there is a substantial simplification in the case of strong bisimilarity – the problem of strong bisimilarity with finite-state processes is in **P** for **OC-A** processes [14], but **PSPACE**-complete for **PDA** processes [19].
- **OC-N** $=_s$ **FS** is in **P**. In fact, we show that **OC-N** $\sqsubseteq_s$ **FS** and **FS** $\sqsubseteq_s$ **OC-N** are in **P**. To the best of our knowledge, this is the first (and rather tight) polynomiality result for simulation with infinite-state systems. Let us note that some equivalence-checking problems between processes of OC-nets and FS processes are still hard (for example, weak bisimilarity is **DP**-hard [14]), so the result is not immediate (see also the comments below).
- Next, we show that the problems of simulation preorder/equivalence between processes of **PDA** (or **OC-A**, or **OC-N**) and **FS** processes are reducible to the model-checking problem for the fixed formula $\varphi \equiv \nu X.[z]\langle z\rangle X$ of (the alternation-free fragment of) the modal μ-calculus. It is essentially a simple observation which was (in a similar form) used already in [2, 16, 18]. The point is that (due to the previous hardness results) we can conclude that the problem of model-checking with φ is **EXPTIME**-complete for **PDA** processes (the upper-bound is due to [25]) and **coNP**-hard for **OC-A** processes. An interesting thing is that the model-checking problem for stateless pushdown (i.e., **BPA**) processes and *any* fixed formula of the modal μ-calculus is already polynomial [25]. The classes of **BPA** and **OC-A** processes are rather natural but *incomparable* subclasses of **PDA** processes – we see that the absence of a finite control is a 'stronger' simplification than the replacement of the storage device (counter instead of stack) in this case.
  As simulation between **OC-N** and **FS** processes is in **P**, the aforementioned technique does not yield any hardness result for model-checking with **OC-N** processes.

Therefore, we examine the problem directly – we prove that even model-checking with a simple fixed formula $\Diamond[a]\Diamond[b]\mathtt{ff}$ of the logic EF is **NP**-hard for **OC-N** processes, and model-checking with another fixed formula $\Box\langle a\rangle\Box\langle b\rangle\mathtt{tt}$ is **coNP**-hard. Hence, we can forget about an efficient model-checking procedure for **OC-N** processes and any modal logic which can express these simple formulae (unless **P** = **NP**).

2 Results about Equivalence-Checking

Theorem 1. *The problem of simulation equivalence between PDA processes and deterministic FS processes is **EXPTIME**-hard.*

Proof. We show **EXPTIME**-hardness by reduction from the acceptance problem for alternating LBA (which is known to be **EXPTIME**-complete). An *alternating LBA* is a tuple $\mathcal{M} = (Q, \Sigma, \delta, q_0, \vdash, \dashv, p)$ where $Q, \Sigma, \delta, q_0, \vdash$, and $\dashv$ are defined as for ordinary non-deterministic LBA (in particular, $\vdash$ and $\dashv$ are the left-end and right-end markers, resp.), and $p : Q \to \{\forall, \exists, acc, rej\}$ is a function which partitions the states of Q into *universal*, *existential*, *accepting*, and *rejecting*, respectively. We assume (w.l.o.g.) that δ is defined so that 'terminated' configurations (i.e., the ones from which there are no further computational steps) are exactly accepting and rejecting configurations. A *computational tree* for $\mathcal{M}$ on a word $w \in \Sigma^*$ is any (finite or infinite) tree T satisfying the following: the root of T is (labeled by) the initial configuration $q_0{\vdash}w{\dashv}$ of $\mathcal{M}$, and if N is a node of $\mathcal{M}$ labeled by a configuration uqv where $u, v \in \Sigma^*$ and $q \in Q$, then the following holds:

- if q is accepting or rejecting, then T is a leaf;
- if q is existential, then T has one successor whose label is (some) configuration which can be reached from uqv in one computational step (according to δ);
- if q is universal, then T has m successors where m is the number of *all* configurations which can be reached from uqv in one step; those configurations are used as labels of the successors in one-to-one fashion.

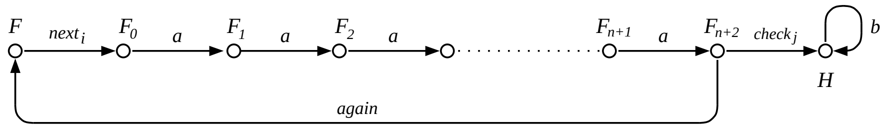
$\mathcal{M}$ *accepts* w iff there is a finite computational tree T such that all leaves of T are accepting configurations.

Now we describe a polynomial algorithm which for a given alternating LBA $\mathcal{M} = (Q, \Sigma, \delta, q_0, \vdash, \dashv, p)$ and a word $w \in \Sigma^*$ constructs a process P of a PDA system Δ and a process F of a finite-state system $\mathcal{F}$ such that

- $P \sqsubseteq_s F$, and
- $F \sqsubseteq_s P$ iff $\mathcal{M}$ does not accept w.

Hence, $\mathcal{M}$ accepts w iff $P \neq_s F$ and we are (virtually) done.

Intuition: The underlying system $\mathcal{F}$ of F looks as follows (note the $\mathcal{F}$ is deterministic):



Intuitively, the goal of F is to demonstrate that there is an accepting computational tree for $\mathcal{M}$ on w, while P aims to show the converse. The game starts with the initial configuration $q_0\vdash w\dashv$ stored in the stack of P. Now F 'chooses' the next configuration (i.e., the rule of δ which is to be applied to the current configuration stored at the top of stack) by emitting one of the $next_i$ actions. The quotes are important here because P is constructed in such a way that it has to accept the choice of F only if the control state of the current configuration is *existential*. If it is *universal*, P can 'ignore' the dictate of F and choose the next configuration according to its own will. The new configuration is then pushed to the stack of P (technically, it is done by guessing individual symbols and an auxiliary verification mechanism is added so that P cannot gain anything if it starts to cheat). As soon as P enters an accepting configuration, it 'dies' (i.e., it is not able to emit any action); and as soon as it enters a rejecting configuration, it starts to behave identically as F. Hence, if there is an accepting computational tree for $\mathcal{M}$ on w, then F can force P to enter an accepting configuration in finitely many rounds (and hence $F \not\sqsubseteq_s P$). If there is no accepting computational tree, then P can successfully defend; it either enters a rejecting configuration or the game goes forever. It means, in both cases, that $F \sqsubseteq_s P$. Moreover, a careful design of P ensures that $P \sqsubseteq_s F$ regardless whether $\mathcal{M}$ accept w or not. A full (formal) proof is omitted due to space constraints; it can be found in [15]. □

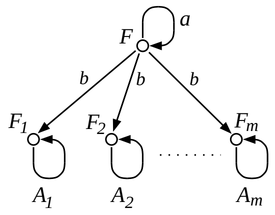
In the proof of our next theorem we use the technique for encoding assignments of Boolean variables in the structure of one-counter automata discovered in [14].

Theorem 2. *The problem of simulation equivalence between OC-A processes and FS processes is* ***coNP****-hard.*

Proof. We show **coNP**-hardness by reduction of the **coNP**-complete problem UNSAT. An instance is a Boolean formula ψ in CNF. The question is whether ψ is unsatisfiable.

Let $\psi \equiv C_1 \wedge \cdots \wedge C_m$ be a formula in CNF where C_i are clauses over propositional variables $x_1, \cdots, x_n$. We construct (in polynomial time) a process P of a OC-A system Δ and a process F of a finite-state system $\mathcal{F}$ such that $P \sqsubseteq_s F$ iff ψ is unsatisfiable. Then we simply consider the processes P' and F' which have the following outgoing transitions: $P' \xrightarrow{x} P, P' \xrightarrow{x} F$, and $F' \xrightarrow{x} F$ where x is a fresh action. Observe that P' is easily definable in the syntax of one-counter processes and F' in the syntax of finite-state processes. Clearly $F' \sqsubseteq_s P'$, and $P' \sqsubseteq_s F'$ iff $P \sqsubseteq_s F$. In other words, $P' =_s F'$ iff ψ is unsatisfiable and it proves our theorem.

It remains to show the construction of P, Δ, F, and $\mathcal{F}$. The set of actions of Δ and $\mathcal{F}$ is $Act = \{a, b, c_1, \ldots, c_m\}$. Let $A_i = Act - \{a, c_i\}$. The set of states of $\mathcal{F}$ is $\{F, F_1, \ldots, F_m\}$ and its transitions are $F \xrightarrow{a} F$, $F \xrightarrow{b} F_i$ for each $1 \leq i \leq m$, and $F_i \xrightarrow{y} F_i$ for each $y \in A_i$ and each $1 \leq i \leq m$. Hence, the system $\mathcal{F}$ looks as follows:



In the construction of Δ we rely on the following theorem of number theory (see, e.g., [3]): Let p_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 $\sum_{i=1}^{n} p_i$. Then f is $\mathcal{O}(n^3)$. This fact ensures that Δ has only polynomially-many control states (see below).

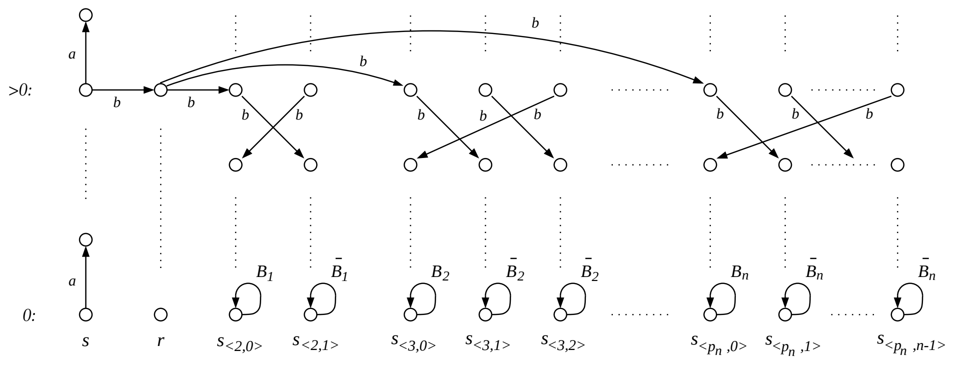
The set of control states Q of Δ is $\{s, r\} \cup \{s_{\langle p_i, j\rangle} \mid 1 \le i \le n, 0 \le j < p_i\}$. For each $1 \le i \le n$ we now define two sets of actions.

- $B_i = \{c_j \mid 1 \le j \le m,$ the variable x_i appears positively in the clause $C_j\}$
- $\overline{B}_i = \{c_j \mid 1 \le j \le m,$ the variable x_i appears negatively in the clause $C_j\}$

Transitions of Δ are defined as follows:

- $sZ \xrightarrow{a} sIZ$, $sI \xrightarrow{a} sII$, $sI \xrightarrow{b} rI$,
- $rI \xrightarrow{b} s_{\langle p_i,0\rangle}I$ for each $1 \le i \le n$,
- $s_{\langle p_i,j\rangle}I \xrightarrow{b} s_{\langle p_i,(j+1) \bmod p_i\rangle}\varepsilon$ for each $1 \le i \le n$ and each $0 \le j < p_i$,
- $s_{\langle p_i,0\rangle}Z \xrightarrow{y} s_{\langle p_i,0\rangle}Z$ for each $0 \le i \le n$ and each $y \in B_i$.
- $s_{\langle p_i,j\rangle}Z \xrightarrow{y} s_{\langle p_i,j\rangle}Z$ for each $0 \le i \le n$, each $1 \le j < p_i$, and each $y \in \overline{B}_i$.

The structure of the transition system associated to Δ is depicted in the following figure (transition systems associated to OC systems can be viewed as two-dimensional 'tables' with an infinite height where control states are used as column indexes and counter values as row indexes; as the outgoing transitions of a process $p(i)$ for $i > 0$ do not depend on the exact value of i, it suffices to depict the out-going transitions at the zero level and (some) non-zero level):



The initial state is $s(0)$. Intuitively, P first increases its counter, emitting a sequence of a's. Then it emits the first b action and changes its control state to r (preserving the value stored in the counter). To each state $r(l)$ we associate the (unique) assignment ν_l defined by $\nu_l(x_i) = \texttt{tt}$ iff $r(l) \to^* s_{\langle p_i,0\rangle}(0)$ (i.e., $\nu_l(x_i) = \texttt{ff}$ iff $r(l) \to^* s_{\langle p_i,j\rangle}(0)$ for some $1 \le j < p_i$). Conversely, for each 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 $f(j) = p_j$ if $\nu(x_j) = \texttt{tt}$, and $f(j) = 1$ otherwise). Now it is easy to check that a clause C_k is true for an assignment ν_l iff at least one of the 'bottom' states $s_{\langle p_i,j\rangle}(0)$ where a c_k-loop is enabled (see above) is reachable from $r(l)$.

Let $s(l)$ be a state of P such that $\nu_l(\psi) = \mathtt{ff}$. It means that there is $1 \leq k \leq m$ such that $\nu_l(C_k) = \mathtt{ff}$. Hence, the process F can safely match the transition $s(l) \xrightarrow{b} r(l)$ by $F \xrightarrow{b} F_k$ (from that point on it can do everything except c_k). However, if there is some l such that $\nu_l(\psi) = \mathtt{tt}$, then F does not have any 'safe' matching move for the transition $s(l) \xrightarrow{b} r(l)$ because none of its F_k successors can do all of the c_i actions. Hence, ψ is unsatisfiable iff $s(0) \sqsubseteq_s F$. □

Now we prove that simulation preorder and simulation equivalence between processes of one-counter *nets* and finite-state processes can be decided in polynomial time. To the best of our knowledge, these are the first polynomiality results for simulation with infinite-state systems. Intuitively, the crucial property which makes our proofs possible (and which does not hold for general one-counter automata) is the following kind of 'monotonicity' — if $p(i)$ is a process of a one-counter net, then $p(i) \sqsubseteq_s p(j)$ for every $j \geq i$.

It should be noted that in our next constructions we prefer simplicity to optimality. Therefore, it does not pay to evaluate the degrees of polynomials explicitly (though it would be of course possible) because they would considerably decrease after some straightforward optimizations. Our only aim here is to prove the membership to **P**.

Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. A family of $\sqsubseteq_s^i$, $i \in \mathbb{N}_0$ relations is defined inductively as follows:

- $s \sqsubseteq_s^0 t$ for all $s, t \in S$;
- $s \sqsubseteq_s^{i+1} t$ iff $s \sqsubseteq_s^i t$ and for each $s \xrightarrow{a} s'$ there is some $t \xrightarrow{a} t'$ such that $s' \sqsubseteq_s^i t'$.

Intuitively, $s \sqsubseteq_s^i t$ iff Ex has a defending strategy for the first i rounds of the simulation game. If we restrict ourselves to processes of *finitely-branching* transition systems (where each state has only finitely many a-successors for every action a), then $s \sqsubseteq_s t$ iff $s \sqsubseteq_s^i t$ for every $i \in \mathbb{N}_0$ (observe that transition systems generated by PDA are finitely-branching). This enables the following (straightforward) polynomial-time algorithm for checking simulation between finite-state processes:

Lemma 1. *Let $\mathcal{F} = (F, Act, \rightarrow)$ and $\mathcal{G} = (G, Act, \rightarrow)$ be finite-state systems with m and n states, respectively. Let $k = m \cdot n$. For all $f \in F$ and $g \in G$ we have that $f \sqsubseteq_s^k g$ iff $f \sqsubseteq_s^{k+1} g$ iff $f \sqsubseteq_s g$. Moreover, the relation $\sqsubseteq_s^k$ can be computed in time which is polynomial in the size of $\mathcal{F}$ and $\mathcal{G}$.*

Proof. If we start to construct the family of '$\sqsubseteq_s^i$' relations according to the above stated definition, we must reach the greatest fixed-point after (at most) k refinement rounds, because '$\sqsubseteq_s^0$' contains only k elements and $\sqsubseteq_s^i \subseteq \sqsubseteq_s^{i+1}$ for each $i \in \mathbb{N}_0$. It is clear that each refinement step can be computed in time which is polynomial in the size of $\mathcal{F}$ and $\mathcal{G}$. □

Lemma 2. *The problem whether a OC-N process can be simulated by a finite-state process is in **P**.*

Proof. Let $\mathcal{N} = (Q, \{I, Z\}, Act, \delta)$ be a one-counter net and $\mathcal{F} = (F, Act, \rightarrow)$ a finite-state system. We show that (a description of) the simulation preorder between processes of $\mathcal{N}$ and $\mathcal{F}$ can be computed in time which is polynomial in the size of $\mathcal{N}$ and $\mathcal{F}$.

The first step of our algorithm is a construction of a *characteristic finite-state system* of $\mathcal{N}$, denoted $\mathcal{F}_{\mathcal{N}}$, which is defined as follows: $\mathcal{F}_{\mathcal{N}} = (\overline{Q}, Act, \rightarrow)$ where $\overline{Q} = \{\overline{p} \mid p \in Q\}$ and $\overline{p} \xrightarrow{a} \overline{q}$ iff $pI \xrightarrow{a} qI^i \in \delta(p, I)$ for some $i \in \mathbb{N}_0$. Hence, a process $\overline{p}$ of $\mathcal{F}_{\mathcal{N}}$ intuitively corresponds to a 'limit process' $p(\infty)$ of $\mathcal{N}$ (in particular, observe that $p(i) \sqsubseteq_s \overline{p}$ for all $p \in Q$ and $i \in \mathbb{N}_0$). It is obvious that the system $\mathcal{F}_{\mathcal{N}}$ can be constructed in linear time.

Next, for all $\overline{p} \in \overline{Q}$ and $f \in F$ we check whether $\overline{p} \sqsubseteq_s f$. It can be done in polynomial time (see Lemma 1). Now observe that if $\overline{p} \sqsubseteq_s f$ for given $\overline{p}$ and f, we can conclude that $p(i) \sqsubseteq_s f$ for *any* $i \in \mathbb{N}_0$, because $p(i) \sqsubseteq_s \overline{p}$. If $\overline{p} \not\sqsubseteq_s f$, then $\overline{p} \not\sqsubseteq_s^k f$ where $k = |Q| \cdot |F|$ (see Lemma 1). Hence, $\overline{p}$ can win the simulation game over f in (at most) k steps. It is clear that the process $p(k)$ can 'mimic' this winning strategy of $\overline{p}$, because the counter can be decreased at most by k within the first k moves (note that if we allowed to test the counter for zero, then $p(k)$ *could not* mimic the first k moves of $\overline{p}$ in general). The same applies to *any* process $p(i)$ where $i \geq k$, because then $p(k) \sqsubseteq_s p(i)$. To sum up, at this point we know if $p(i) \sqsubseteq_s f$ for all $p \in Q, f \in F$, and $i \geq k$. It remains to decide simulation between pairs of the form $(p(i), f)$ where $0 \leq i < k$. As there are only $|Q| \cdot |F| \cdot k = |Q|^2 \cdot |F|^2$ such states, we can use a simple refinement technique similar to the one of Lemma 1. Formally, we define a family of $\mathcal{R}^j$ relations inductively as follows:

– $\mathcal{R}^0 = \{(p(i), f) \mid i < k, p \in Q, f \in F\}$
– $\mathcal{R}^{j+1}$ consists of those pairs of the form $(p(i), f)$ for which we either have that $\overline{p} \sqsubseteq_s g$, or $(p(i), f) \in \mathcal{R}^j$ and for each move $p(i) \xrightarrow{a} q(l)$ there is a move $f \xrightarrow{a} g$ such that $\overline{q} \sqsubseteq_s g$ or $(q(l), g) \in \mathcal{R}^j$.

Let $\mathcal{R}$ be the greatest fixed point of this refinement procedure. First, observe that $\mathcal{R}$ is computable in **P** because it is reached in (at most) $|Q|^2 \cdot |F|^2$ refinement steps and each step can be obviously computed in polynomial time. Now let us consider a pair of the form $(p(i), f)$ where $p \in Q$, $i < k$, and $f \in F$. If $(p(i), f) \notin \mathcal{R}$, then obviously $p(i) \not\sqsubseteq_s f$. On the other hand, if $(p(i), f) \in \mathcal{R}$, then $p(i) \sqsubseteq_s f$ because we can readily confirm that the relation $\mathcal{R} \cup \{(q(l), g) \mid q \in Q, l \in \mathbb{N}_0, g \in F, \overline{q} \sqsubseteq_s g\}$ is a simulation. □

Lemma 3. *The problem whether a finite-state process can be simulated by a OC-N process is in **P**.*

Proof. Let $\mathcal{N} = (Q, \{I, Z\}, Act, \delta)$ be a one-counter net and $\mathcal{F} = (F, Act, \rightarrow)$ a finite-state system. Similarly as in the previous lemma we show that (a description of) the simulation preorder between processes of $\mathcal{F}$ and $\mathcal{N}$ can be computed in time which is polynomial in the size of $\mathcal{N}$ and $\mathcal{F}$. However, the argument is slightly more complicated in this case.

We start with one auxiliary definition. For all $f \in F$ and $p \in Q$ we define the *frontier counter value*, denoted $\mathcal{V}(f, p)$, to be the least $i \in \mathbb{N}_0$ such that $f \sqsubseteq_s p(i)$; if there is no such i, we put $\mathcal{V}(f, p) = -1$. Our aim is to show that every frontier counter value is bounded by $|Q| \cdot |F|$, i.e., $\mathcal{V}(f, p) \leq |Q| \cdot |F|$ for all $f \in F$ and $p \in Q$. Let m

be the maximal frontier value. It suffices to prove that for each n such that $1 \leq n \leq m$ there are $f \in F$ and $p \in Q$ such that $\mathcal{V}(f,p) = n$. Let us suppose the converse, i.e., there is some $n \geq 1$ such that there is at least one frontier value greater then n, some frontier values are (possibly) less than n, but no frontier value equals to n. It follows directly from the definition of frontier points that the greatest simulation among the processes of $\mathcal{F}$ and $\mathcal{N}$ is the following relation $\mathcal{R}$:

$$\mathcal{R} = \{(f, p(i)) \mid f \in F, p \in Q, \mathcal{V}(f,p) \geq 0, i \geq \mathcal{V}(f,p)\}$$

Now we show that if there is some n with the above stated properties, than we can actually construct a simulation which is strictly larger than $\mathcal{R}$, which is a contradiction. Let $\mathcal{R}'$ be the following finite relation:

$$\mathcal{R}' = \{(g, q(c)) \mid g \in F, q \in Q, \mathcal{V}(g,q) > n, c = \mathcal{V}(g,q) - 1\}$$

As $n < m$, $\mathcal{R}'$ is clearly nonempty. We show that $\mathcal{R} \cup \mathcal{R}'$ is a simulation. To do that, it suffices to check the simulation condition for pairs of $\mathcal{R}'$, because $\mathcal{R}$ itself is a simulation. Let $(g, q(c)) \in \mathcal{R}'$ and $g \xrightarrow{a} h$. We need to find some move $q(c) \xrightarrow{a} \alpha$ such that the pair (h, α) is related by $\mathcal{R} \cup \mathcal{R}'$. However, as $(g, q(c)) \in \mathcal{R}'$, we have that $c = \mathcal{V}(g,q) - 1$ and hence $(g, q(c+1)) \in \mathcal{R}$. Therefore, there must be some move $q(c+1) \xrightarrow{a} r(l)$ such that $(h, r(l)) \in \mathcal{R}$ (also observe that $l \geq n$). It means that $q(c) \xrightarrow{a} r(l-1)$ (here we use the fact that $c \geq 1$). Now if $(h, r(l-1)) \in \mathcal{R}$, we are done immediately. If it is not the case, then l is the frontier counter value for h and r by definition, i.e., $l = \mathcal{V}(h,r)$. As $l \geq n$ and there is no frontier value which equals to n, we conclude that $l > n$ — but it means that $(h, r(l-1)) \in \mathcal{R}'$ by definition of $\mathcal{R}'$.

Let $k = |Q| \cdot |F|$. Now let us realize that if we could decide simulation for all pairs of the form $(f, p(k))$ in polynomial time, we would be done — observe that if $f \sqsubseteq_s p(k)$, then clearly $f \sqsubseteq_s p(i)$ for all $i \geq k$. As all frontier counter values are bounded by k (see above), we can also conclude that if $f \not\sqsubseteq_s p(k)$ then $f \not\sqsubseteq_s p(i)$ for all $i \geq k$. Simulation between the k^2 remaining pairs of the form $(f, p(i))$ where $i < k$ could be then decided in the same way as the previous lemma, i.e., by computing the greatest fixed-point of a refinement procedure defined by

- $\mathcal{R}^0 = \{(f, p(i)) \mid f \in F, p \in Q, i < k\}$
- $\mathcal{R}^{j+1}$ consists of those pairs of the form $(f, p(i))$ such that $(f, p(i)) \in \mathcal{R}^j$ and for each move $f \xrightarrow{a} g$ there is a move $p(i) \xrightarrow{a} q(l)$ such that either $(g, q(l)) \in \mathcal{R}^j$, or $l = k$ and $g \sqsubseteq_s q(k)$.

The greatest fixed-point is reached after (at most) k^2 refinement steps and each step can be computed in polynomial time.

Now we prove that simulation for the pairs of the form $(f, p(k))$ can be indeed decided in polynomial time. To do that, we show that $f \sqsubseteq_s p(k)$ iff $f \sqsubseteq_s^{2k^2} p(k)$. It clearly suffices — as $p(k)$ cannot increase the counter to more than $2k^2 + k$ in $2k^2$ moves, we can decide whether $f \sqsubseteq_s^{2k^2} p(k)$ simply by computing the '$\sqsubseteq_s^{2k^2}$' relation between the states of the system $\mathcal{F}$ and a finite-state system $(S, \Sigma, \rightarrow)$ where $S = \{(p,i) \mid p \in Q, 0 \leq i < 2k^2 + k\}$ and $\rightarrow$ is given by $(p,i) \xrightarrow{a} (q,j)$ iff $p(i) \xrightarrow{a} q(j)$;

then we just look if $f \sqsubseteq_s^{2k^2} (p, k)$. This can be of course done in polynomial time (see Lemma 1).

Let $j \in \mathbb{N}_0$ be the least number such that $f \not\sqsubseteq_s^j p(k)$. Then Al can win the simulation game in j rounds, which means that there is a sequence

$$(f_j, p_j(l_j)) \xrightarrow{a_j} (f_{j-1}, p_{j-1}(l_{j-1})) \xrightarrow{a_{j-1}} \cdots \xrightarrow{a_2} (f_1, p_1(l_1)) \xrightarrow{a_1} (f_0, -)$$

of game positions where $f = f_j$, $p(k) = p_j(l_j)$, and $f_i \not\sqsubseteq_s^i p_i(l_i)$ for each $1 \leq i \leq j$. The Al's attack at a position $(f_i, p_i(l_i))$ is $f_i \xrightarrow{a_i} f_{i-1}$, and Ex's defending move is $p_i(l_i) \xrightarrow{a_i} p_{i-1}(l_{i-1})$ (observe that, in particular, $f_1 \not\sqsubseteq_s^1 p_1(l_1)$ and hence $p_1(l_1)$ cannot emit the action a_1). Moreover, we assume (w.l.o.g.) that Ex defends 'optimally', i.e., $f_i \sqsubseteq_s^{i-1} p_i(l_i)$ for each $1 \leq i \leq j$. The first step is to show that $l_i \leq 2k$ for each $1 \leq i \leq j$. Suppose the converse, i.e., there is some i with $l_i > 2k$. As the counter can be increased at most by one in a single transition, we can select a (strictly) increasing sequence of indexes $s_0, s_1, \ldots, s_k$ such that $l_{s_i} = k + i$ for each $0 \leq i \leq k$. Furthermore, as $k = |Q| \cdot |F|$, there must be two indexes s_u, s_v where $u < v$ such that $f_{s_u} = f_{s_v}$ and $p_{s_u} = p_{s_v}$. Let us denote $f_{s_u} = f_{s_v}$ by f' and $p_{s_u} = p_{s_v}$ by p'. Now we see (due to the optimality assumption) that $f' \sqsubseteq_s^{s_u - 1} p'(k + u)$ and $f' \not\sqsubseteq_s^{s_v} p'(k + v)$. As $s_u - 1 \geq s_v$, we also have $f' \not\sqsubseteq_s^{s_u - 1} p'(k + v)$. However, as $u < v$ we obtain $f' \sqsubseteq_s^{s_u - 1} p'(k + u) \sqsubseteq_s p'(k + v)$, hence $f' \sqsubseteq_s^{s_u - 1} p'(k + v)$ and we derived a contradiction. The rest is now easy — if $j > 2k^2$ (i.e., if Al cannot win in $2k^2$ rounds) then there must be some $u > v$ such that $f_u = f_v$, $p_u = p_v$, and $l_u = l_v$. It follows directly from the fact that $k = |Q| \cdot |F|$ and that each l_i is at most $2k$. Now we can derive a contradiction in the same way as above — denoting $f_u = f_v$ by f', $p_u = p_v$ by p', and $l_u = l_v$ by l', we obtain (due to the optimality assumption) that $f' \sqsubseteq_s^{u-1} p'(l')$ and $f' \not\sqsubseteq_s^v p'(l')$. As $u - 1 \geq v$, we have the desired contradiction. □

An immediate consequence of Lemma 2 and Lemma 3 is the following theorem:

Theorem 3. *The problem of simulation equivalence between OC-N processes and FS processes is in **P**.*

3 Results about Model-Checking

In this section we show that there is a close relationship between simulation-checking problems and the model-checking problem for the formula $\varphi \equiv \nu X.[z]\langle z\rangle X$ of the modal μ-calculus. It is essentially a simple observation which was (in a similar form) used already in [2, 16, 18].

As we omitted a formal definition of syntax and semantics of this logic, we clarify the meaning of φ at this point. Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. Let $f_\varphi : 2^S \rightarrow 2^S$ be a function defined as follows:

$$f_\varphi(M) = \{s \in S \mid \forall (s \xrightarrow{z} s') \text{ we have that } \exists (s' \xrightarrow{z} s'') \text{ such that } s'' \in M\}$$

The denotation of φ (i.e., the set of states where φ holds), written $[\![\varphi]\!]$, is defined by

$$[\![\varphi]\!] = \bigcup \{U \subseteq S \mid U \subseteq f_\varphi(U)\}$$

Hence, $[\![\varphi]\!]$ is the greatest fixed-point of the (monotonic) function f_φ. As usual, we write $t \models \varphi$ instead of $t \in [\![\varphi]\!]$.

Theorem 4. *Let P be a process of a PDA system $\Delta = (Q, \Gamma, Act, \delta)$, and F a process of a finite-state system $\mathcal{F} = (S, Act, \rightarrow)$. Then it possible to construct (in polynomial time) processes A, B of a PDA system Δ_1 and a process C of a PDA system Δ_2 such that $P \sqsubseteq_s F$ iff $A \models \varphi$, $F \sqsubseteq_s P$ iff $B \models \varphi$, and $P =_s F$ iff $C \models \varphi$.*

Proof. Intuitively, the processes A, B and C 'alternate' the transitions of P and F in an appropriate way. We start with the definition of Δ_1. The set of control states of Δ_1 is $Q \times S \times (Act \cup \{?\}) \times \{0, 1\}$, the set of actions is Act, the stack alphabet $\overline{\Gamma}$ is $\Gamma \cup \{Z\}$ where $Z \notin \Gamma$ is a fresh symbol (bottom of stack). The set of transitions is the least set $\overline{\delta}$ satisfying the following:

– if $pX \xrightarrow{a} q\alpha$ is a rule of δ, then $(p, F, ?, 0)X \xrightarrow{z} (q, F, a, 1)\alpha$ and $(p, F, a, 0)X \xrightarrow{z} (q, F, ?, 1)\alpha$ are rules of $\overline{\delta}$ for each $F \in S$;
– if $F \xrightarrow{a} F'$, then $(p, F, ?, 1)X \xrightarrow{z} (p, F', a, 0)X$ and $(p, F, a, 1)X \xrightarrow{z} (p, F', ?, 0)X$ are rules of $\overline{\delta}$ for all $p \in Q$ and $X \in \overline{\Gamma}$;

Let $P \equiv p\alpha$. We put $A \equiv (p, F, ?, 0)\alpha Z$ and $B \equiv (p, F, ?, 1)\alpha Z$. Observe that A alternates the moves of P and F — first P performs a transition whose label is stored in the finite control and passes the token to F (by changing 0 to 1); then F emits some transition with the same (stored) label and passes the token back to P. The new bottom symbol Z is added to ensure that F cannot 'die' within A just due to the emptiness of the stack. Now it is obvious that $P \sqsubseteq_s F$ iff $A \models \varphi$; the fact that $Q \sqsubseteq_s P$ iff $B \models \varphi$ can be justified in the same way.

The way how to define C is now easy to see – it suffices to ensure that the only transitions of C are $C \xrightarrow{z} C'$ and $C \xrightarrow{z} C''$ where $C' \xrightarrow{z} A$ and $C'' \xrightarrow{z} B$. It can be achieved by a straightforward extension of Δ_1. □

The proof of Theorem 4 carries over to processes of one-counter automata and one-counter nets immediately (observe there is no need to add a new bottom symbol when constructing Δ_1 and Δ_2 because the zero-marker of one-counter systems is never removed from the stack by definition.

Corollary 1. *The model-checking problem for φ is*

– ***EXPTIME****-complete for PDA processes;*
– ***coNP****-hard for OC-A processes;*

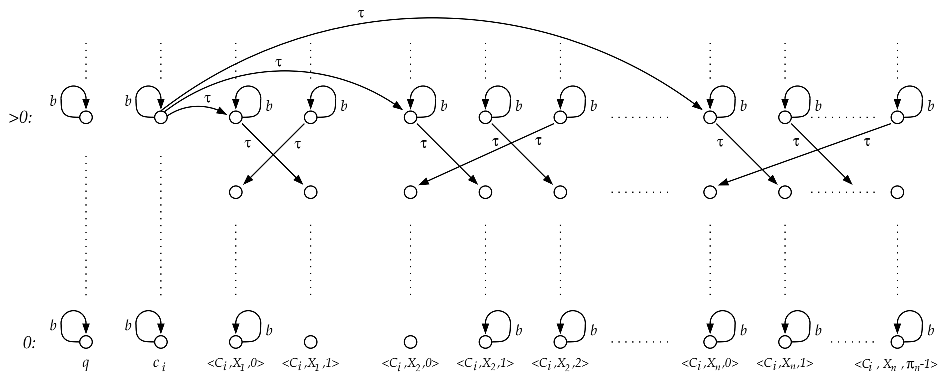
As simulation between OC-N and FS processes is in **P**, Theorem 4 does not imply any hardness result for model-checking with OC-N processes. Therefore, we examine this problem 'directly' by showing that a simple fixed formula $\Diamond[a]\Diamond[b]\mathtt{ff}$ of the logic EF is **NP**-hard for OC-N processes. In our proof we use a slightly modified version of the construction which was given in [14] to prove **DP**-hardness of weak bisimilarity between OC-N and FS processes. To make this paper self-contained, we present a full proof here.

Theorem 5. *Let $p(0)$ be a process of a one-counter net $\mathcal{N}$. The problem if $p(0) \models \Diamond[a]\Diamond[b]\mathtt{ff}$ is* ***NP****-hard.*

Proof. 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 construct a OC-N system $\mathcal{N} = (Q, \{I, Z\}, \{a, b, \tau\}, \delta)$ and its process $p(0)$ such that φ is satisfiable iff $p(0) \models \Diamond[a]\Diamond[b]\mathtt{ff}$. The construction of $\mathcal{N}$ will be described in a stepwise manner. The sets Q and δ are initialized as follows: $Q = \{q\}$, $\delta = \{qI \xrightarrow{b} qI, qZ \xrightarrow{b} qZ\}$. Now, for each clause C_i, $1 \leq i \leq m$, we do the following:

– Let π_j denote the j^{th} prime number. We add a new control state c_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 $\langle C_i, x_j, k\rangle$.
– For each newly added control state s we add to δ the transitions $sI \xrightarrow{a} qI, sZ \xrightarrow{a} qZ$.
– For each $1 \leq j \leq n$ we add to δ the transitions $c_iI \xrightarrow{\tau} \langle C_i, X_j, 0\rangle I$.
– For all j, k such that $1 \leq j \leq n$ and $0 \leq k < \pi_j$ we add to δ the transition $\langle C_i, x_j, k\rangle I \xrightarrow{\tau} \langle C_i, x_j, (k+1) \bmod \pi_j\rangle \varepsilon$.
– We add to δ the 'loops' $c_iI \xrightarrow{b} c_iI, c_iZ \xrightarrow{b} c_iZ$.
– For all j, k such that $1 \leq j \leq n$ and $0 \leq k < \pi_j$ we add to δ the loop $\langle C_i, x_j, k\rangle I \xrightarrow{b} \langle C_i, x_j, k\rangle I$.
– If a variable x_j does *not* appear positively in a clause C_i, then we add to δ the loop $\langle C_i, x_j, 0\rangle Z \xrightarrow{b} \langle C_i, x_j, 0\rangle Z$.
– If a variable x_j does not appear negatively in a clause C_i, then we add to δ the loops $\langle C_i, x_j, k\rangle Z \xrightarrow{b} \langle C_i, x_j, k\rangle Z$ for every $1 \leq k < \pi_j$.

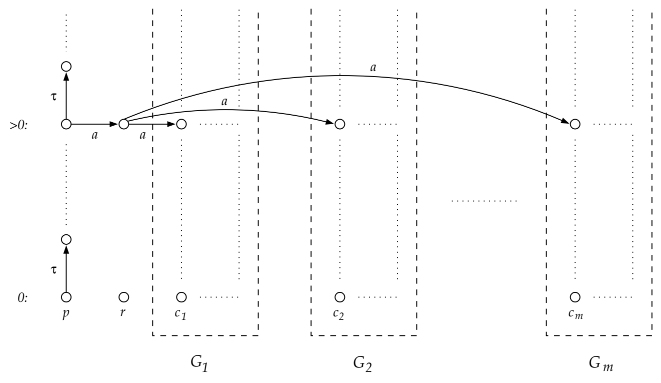
If we draw the transition system which is generated by the current approximation of $\mathcal{N}$, we obtain a collection of G_i graphs, $1 \leq i \leq m$; each G_i corresponds to the 'subgraph' of the transition system associated to $\mathcal{N}$ which is obtained by restricting Q to the set of control states which have been added for the clause C_i. The structure of G_i is shown in the following picture (the a-transitions to the states of the form $q(j)$ are omitted as the picture would become too complicated).



Now we can observe the following:

– For each $l > 0$, the state $c_i(l)$ 'encodes' the (unique) assignment ν_l in the same way as in the proof of Theorem 2, i.e., ν_l is defined by $\nu_l(x_j) = \mathtt{tt}$ iff $c_i(l) \to^* \langle C_i, x_j, 0\rangle(0)$; conversely, for each 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 $\nu(x_j) = \mathtt{tt}$, and $f(j) = 1$ otherwise).
– For each $l > 0$ we have that $\nu_l(C_i) = \mathtt{tt}$ iff $c_i(l) \models \Diamond[b]\mathtt{ff}$. Indeed, observe that $\nu_l(C_i) = \mathtt{tt}$ iff $c_i(l)$ can reach some of the 'zero-states' where the action b is disabled.

We finish the construction of $\mathcal{N}$ by connecting the G_i components together. To do that, we add two new control states p and r to Q, and enrich δ by adding the transitions $pZ \xrightarrow{\tau} pIZ$, $pI \xrightarrow{\tau} pII$, $pI \xrightarrow{a} qI$, $pZ \xrightarrow{a} qZ$, $pI \xrightarrow{\tau} rI$, and $rI \xrightarrow{a} c_iI$ for every $1 \le i \le m$. The structure of of the transition system associated to $\mathcal{N}$ is shown below (again, the a-transitions to the states of the form $q(j)$ are omitted).



Now we can observe the following:

– The only states which can (potentially) satisfy the formula $[a]\Diamond[b]\mathtt{ff}$ are those of the form $r(l)$, because all other states have an a-transition to a state of the form $q(j)$ where is it impossible to get rid of b's.
– A state $r(l)$ satisfies the formula $[a]\Diamond[b]\mathtt{ff}$ iff $c_i(l) \models \Diamond[b]\mathtt{ff}$ for all $1 \le i \le m$ iff $\nu_l(C_i) = \mathtt{tt}$ for each $1 \le i \le m$ (due to the previous observations) iff $\nu_l(\varphi) = \mathtt{tt}$.

Hence, φ is satisfiable iff there is $l \in \mathbb{N}$ such that $r(l)$ satisfies $[a]\Diamond[b]\mathtt{ff}$ iff $p(0) \models \Diamond[a]\Diamond[b]\mathtt{ff}$. □

Corollary 2. *Let $p(0)$ be a process of a one-counter net $\mathcal{N}$. The problem if $p(0) \models \Box\langle a\rangle\Box\langle b\rangle\mathtt{tt}$ is **coNP**-hard.*

4 Conclusions

This paper fills some gaps in our knowledge on complexity of simulation-checking and model-checking with (subclasses of) pushdown automata. The following table gives a summary of known results (contributions of this paper are in boldface). For comparison, related results about checking strong and weak bisimilarity (denoted by $\sim$ and

$\approx$, respectively) with finite-state processes are also shown. The overview supports the claim that simulation tends to be computationally harder than bisimilarity; to the best of our knowledge, there is so far no result violating this ‘rule of thumb’.

	PDA	BPA	OC-A	OC-N
$\sim$ FS	PSPACE-complete [19]	$\in$ P [17]	$\in$ P [14]	$\in$ P [14]
$\approx$ FS	PSPACE-hard [19] $\in$ EXPTIME [10]	$\in$ P [17]	DP-hard [14]	DP-hard [14]
$=_s$ FS	**EXPTIME-complete**	coNP-hard [16]	**coNP-hard**	$\in$ **P**

References

1. 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.
2. H.R. Andersen. *Verification of Temporal Properties of Concurrent Systems*. PhD thesis, Arhus University, 1993.
3. E. Bach and J. Shallit. *Algorithmic Number Theory. Vol. 1, Efficient Algorithms*. The MIT Press, 1996.
4. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.
5. S. Christensen. *Decidability and Decomposition in Process Algebras*. PhD thesis, The University of Edinburgh, 1993.
6. E.A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, B, 1991.
7. J. Esparza and J. Knop. An automata-theoretic approach to interprocedural data-flow analysis. In *Proceedings of FoSSaCS'99*, volume 1578 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 1999.
8. J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):353–371, 1994.
9. Y. Hirshfeld. Petri nets and the equivalence problem. In *Proceedings of CSL'93*, volume 832 of *Lecture Notes in Computer Science*, pages 165–174. Springer, 1994.
10. 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.
11. P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proceedings of CONCUR'95*, volume 962 of *Lecture Notes in Computer Science*, pages 348–362. Springer, 1995.
12. 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.
13. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
14. A. Kučera. Efficient verification algorithms for one-counter processes. In *Proceedings of ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2000.
15. A. Kučera. On simulation-checking with sequential systems. Technical report FIMU-RS-2000-05, Faculty of Informatics, Masaryk University, 2000.

16. A. Kučera and R. Mayr. Simulation preorder on simple process algebras. In *Proceedings of ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 503–512. Springer, 1999.
17. 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.
18. F. Laroussinie and Ph. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *Proceedings of FoSSaCS 2000*, volume 1784 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2000.
19. R. Mayr. On the complexity of bisimulation problems for pushdown automata. In *Proceedings of IFIP TCS'2000*, volume 1872 of *Lecture Notes in Computer Science*. Springer, 2000.
20. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
21. 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.
22. W. Reisig. *Petri Nets—An Introduction*. Springer, 1985.
23. C. Stirling. Modal and temporal logics. *Handbook of Logic in Computer Science*, 2:477–563, 1992.
24. 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.
25. I. Walukiewicz. Pushdown processes: Games and model checking. In *Proceedings of CAV'96*, volume 1102 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1996.