# EXPSPACE-Complete Variant of Countdown Games, and Simulation on Succinct One-Counter Nets

Petr Jančar[1(✉)], Petr Osička[1], and Zdeněk Sawa[2]

[1] Department of Computer Science, Faculty of Science, Palacký University Olomouc, Olomouc, Czechia
petr.jancar@upol.cz, osicka@acm.org

[2] Department of Computer Science, FEI, Technical University of Ostrava, Ostrava, Czechia
zdenek.sawa@vsb.cz

**Abstract.** We answer an open complexity question for simulation preorder of succinct one-counter nets (i.e., one-counter automata with no zero tests where counter increments and decrements are integers written in binary), by showing that all relations between bisimulation equivalence and simulation preorder are EXPSPACE-hard for these nets. We describe a reduction from reachability games whose EXPSPACE-completeness in the case of succinct one-counter nets was shown by Hunter [RP 2015], by using other results. We also provide a direct self-contained EXPSPACE-completeness proof for a special case of such reachability games, namely for a modification of countdown games that were shown EXPTIME-complete by Jurdzinski, Sproston, Laroussinie (LMCS 2008); in our modification the initial counter value is not given but is freely chosen by the first player.

**Keywords:** Succinct one-counter net · Simulation
Countdown game · Complexity

## 1 Introduction

One-counter automata (OCA), i.e., finite automata equipped with a nonnegative counter, are studied as one of the simplest models of infinite-state systems. They can be viewed as a special case of Minsky counter machines, or as a special case of pushdown automata. In general, OCA can test the value of the counter for zero, i.e., some transitions could be enabled only if the value of the counter is zero. One-counter nets (OCN) are a "monotonic" subclass of OCA where every transition enabled for zero is also enabled for nonzero values. As usual,

This research has been supported by Grant No. 18-11193S, Grant Agency of the Czech Rep. (P. Jančar and P. Osička), and by Grant No. SP2018/172, VŠB-Techn. Univ. Ostrava (Z. Sawa).


I. Potapov and P.-A. Reynier (Eds.): RP 2018, LNCS 11123, pp. 59–74, 2018.
https://doi.org/10.1007/978-3-030-00250-3_5

we can consider deterministic, nondeterministic, and/or alternating versions of OCA and/or OCN. The basic versions are *unary*, where the counter can be incremented and decremented by one in one step, while in the *succinct* versions the possible changes can be arbitrary integers (but fixed for a given transition); as usual, the changes are assumed to be written in binary in a description of a given automaton.

Problems that have been studied on OCA and OCN include reachability, equivalence, model checking, and also different kinds of games played on these automata. One of the earliest results showed decidability of (language) equivalence for deterministic OCA [24]. (The open polynomiality question in [24] was positively answered in [2].)

Later other behavioural equivalences (besides language equivalence) have been studied. Most relevant for us is the research started by Abdulla and Čerāns who showed in [1] that simulation preorder on one-counter nets is decidable. An alternative proof of this fact was given in [14]; it was also noted that simulation equivalence is undecidable for OCA. A relation to bisimulation problems was shown in [12]. Kučera showed some lower bounds in [19]; Mayr [20] showed the undecidability of weak bisimulation equivalence on OCN.

Simulation preorder on one-counter nets turned out PSPACE-complete: the lower bound was shown by Srba [23], and the upper bound by Hofman, Lasota, Mayr, and Totzke [10]. It was also shown in [10] that deciding weak simulation on OCN can be reduced to deciding strong simulation on OCN, and thus also solved in polynomial space. (Strong) bisimulation equivalence on OCA is also known to be PSPACE-complete [3].

The mentioned results deal with unary OCA and OCN. Succinct (and parametric) OCA were considered, e.g., in [9], where reachability on succinct OCA was shown to be NP-complete. We note that PSPACE-membership of problems for the unary case easily yields EXPSPACE-membership for the succinct (binary) case. Games studied on OCA include, e.g., parity games on one-counter processes (with test for zero) [22], and are closely related to counter reachability games (e.g. [21]). Model checking problems on OCA were studied for many types of logics, e.g., LTL [5], branching time logics [7], or first-order logics [8]. DP-lower bounds for some model-checking (and also equivalence checking) problems were shown in [13].

An involved result by Göller, Haase, Ouaknine, Worrell [6] shows that model checking a fixed CTL formula on succinct one-counter automata is EXPSPACE-hard. Their proof is interesting and nontrivial, and uses two involved results from complexity theory. The technique of this proof was referred to by Hunter [11], to derive EXPSPACE-hardness of reachability games on succinct one-counter nets.

*Our Contribution.* In this paper we close a complexity gap for the simulation problem that was mentioned in [10], noting that there was a PSPACE lower bound and an EXPSPACE upper bound for the problem. We show EXPSPACE-hardness (and thus EXPSPACE-completeness) of the problem, using a defender-choice technique (cf., e.g., [16]) to reduce reachability games to any relation between simulation preorder and bisimulation equivalence.

The EXPSPACE-hardness can be derived by [6,11]. Here we present a direct proof of EXPSPACE-hardness (and completeness) of a special case of reachability games, which we call the "existential countdown games". It is a mild relaxation of the countdown games from [17] (or their variant from [18]) which is an interesting EXPTIME-complete problem. We thus provide a complete EXPSPACE-hardness proof, independent of [11] or [6].

*Organization of the Paper.* Section 2 gives the basic definitions. In Sect. 3 we show the "existential" countdown games and their EXPSPACE-completeness. Section 4 describes the reductions from reachability games to (bi)simulation relations. We finish with some additional remarks in Sect. 5.

## 2 Basic Definitions

By $\mathbb{Z}$ and $\mathbb{N}$ we denote the sets of integers and of nonnegative integers, respectively. We use $[i, j]$, where $i, j \in \mathbb{Z}$, for denoting the set $\{i, i+1, \ldots, j\}$ (which is empty when $i > j$).

**Labelled Transition Systems and (Bi)simulations.** A *labelled transition system*, an *LTS* for short, is a tuple

$$\mathcal{L} = (S, Act, (\xrightarrow{a})_{a \in Act})$$

where $S$ is the set of *states*, $Act$ is the set of *actions*, and $\xrightarrow{a} \subseteq S \times S$ is the set of *a-transitions* (transitions labelled with $a$), for each $a \in Act$. We write $s \xrightarrow{a} t$ instead of $(s, t) \in \xrightarrow{a}$. By $s \xrightarrow{a}$ we denote that $a$ is *enabled in* $s$, i.e., $s \xrightarrow{a} t$ for some $t$.

Given $\mathcal{L} = (S, Act, (\xrightarrow{a})_{a \in Act})$, a relation $R \subseteq S \times S$ is a *simulation* if for every $(s, s') \in R$ and every $s \xrightarrow{a} t$ there is $s' \xrightarrow{a} t'$ such that $(t, t') \in R$; if, moreover, for every $(s, s') \in R$ and every $s' \xrightarrow{a} t'$ there is $s \xrightarrow{a} t$ such that $(t, t') \in R$, then $R$ is a *bisimulation*.

The union of all simulations (on $S$) is the maximal simulation, denoted $\preceq$; it is a preorder, called *simulation preorder*. The union of all bisimulations is the maximal bisimulation, denoted $\sim$; it is an equivalence, called *bisimulation equivalence* (or *bisimilarity*). We obviously have $\sim \subseteq \preceq$.

**(Labelled) One-Counter Nets (OCNs and SOCNs).** A *labelled one-counter net*, or just a *one-counter net* or even just an *OCN* for short, is a triple

$$\mathcal{N} = (Q, Act, \delta),$$

where $Q$ is the finite set of *control states*, $Act$ the finite set of *actions*, and $\delta \subseteq Q \times Act \times \{-1, 0, +1\} \times Q$ is the finite set of (*labelled transition*) *rules*. By allowing $\delta \subseteq Q \times Act \times \mathbb{Z} \times Q$, and presenting $z \in \mathbb{Z}$ in the rules $(q, a, z, q')$ in binary, we get a *succinct one-counter net*, or a *SOCN* for short. (One-counter automaton arises by adding the ability to test explicitly if the counter is zero.) We present rules $(q, a, z, q')$ rather as $q \xrightarrow{a,z} q'$.

Each OCN or SOCN $\mathcal{N} = (Q, Act, \delta)$ has the *associated LTS*

$$\mathcal{L}_{\mathcal{N}} = (Q \times \mathbb{N}, Act, (\xrightarrow{a})_{a \in Act}) \tag{1}$$

where $(q,m) \xrightarrow{a} (q',n)$ iff $q \xrightarrow{a,n-m} q'$ is a rule in $\delta$. We often write a state $(q,m)$, which is also called a *configuration*, in the form $q(m)$, and we view $m$ as a value of a nonnegative counter. A rule $q \xrightarrow{a,z} q'$ thus induces transitions $q(m) \xrightarrow{a} q'(m+z)$ for all $m \geq \max\{0, -z\}$.

**Reachability Games (r-Games), Winning Areas, Ranks of States.** By a *reachability game*, or an *r-game* for short, we mean a tuple

$$\mathcal{G} = (V, V_\exists, \rightarrow, \mathcal{T}),$$

where $V$ is the set of *states* (or *vertices*), $V_\exists \subseteq V$ is the set of *Eve's states*, $\rightarrow \subseteq V \times V$ is the *transition relation* (or the set of *transitions*), and $\mathcal{T} \subseteq V$ is the set of *target states*. By *Adam's states* we mean the elements of $V_\forall = V \setminus V_\exists$.

*Eve's winning area* is $Win_\exists = \bigcup_{\lambda \in Ord} W_\lambda$, for $Ord$ being the class of ordinals, where the sets $W_\lambda \subseteq V$ are defined inductively as follows.

We put $W_0 = \mathcal{T}$; for $\lambda > 0$ we put $W_{<\lambda} = \bigcup_{\lambda' < \lambda} W_{\lambda'}$, and we stipulate:

(a) if $s \notin W_{<\lambda}$, $s \in V_\exists$, and $s \rightarrow \bar{s}$ for some $\bar{s} \in W_{<\lambda}$, then $s \in W_\lambda$;
(b) if $s \notin W_{<\lambda}$, $s \in V_\forall$, and we have $\emptyset \neq \{\bar{s} \mid s \rightarrow \bar{s}\} \subseteq W_{<\lambda}$, then $s \in W_\lambda$.

(If (a) applies, then $\lambda$ is surely a successor ordinal.)

For each $s \in Win_\exists$, by $\textsc{rank}(s)$ we denote (the unique) $\lambda$ such that $s \in W_\lambda$. A transition $s \rightarrow \bar{s}$ is *rank-reducing* if $\textsc{rank}(s) > \textsc{rank}(\bar{s})$. We note that for any $s \in Win_\exists$ with $\textsc{rank}(s) > 0$ we have: if $s \in V_\exists$, then there is at least one rank-reducing transition $s \rightarrow \bar{s}$ (in fact, $\textsc{rank}(s) = \textsc{rank}(\bar{s}) + 1$ in this case); if $s \in V_\forall$, then there is at least one transition $s \rightarrow \bar{s}$ and all such transitions are rank-reducing. This entails that $Win_\exists$ is the set of states from which Eve has a winning strategy that guarantees reaching (some state in) $\mathcal{T}$ when Eve is choosing a next transition in Eve's states and Adam is choosing a next transition in Adam's states.

We are primarily interested in the games that have (at most) countably many states and are finitely branching (the sets $\{\bar{s} \mid s \rightarrow \bar{s}\}$ are finite for all $s$). In such cases we have $\textsc{rank}(s) \in \mathbb{N}$ for each $s \in Win_\exists$.

We now define specific r-games, presented by (unlabelled) SOCNs with partitioned control-state sets.

**Succinct One-Counter Net r-Games (socn-r-Games), Problem SOCNRG.** By a *succinct one-counter net r-game*, a *socn-r-game* for short, we mean a tuple

$$\mathcal{N} = (Q, Q_\exists, \delta, p_{win})$$

where $Q$ is the finite set of *(control) states*, $Q_\exists \subseteq Q$ is the set of *Eve's (control) states*, $p_{win} \in Q$ is the *target (control) state*, and $\delta \subseteq Q \times \mathbb{Z} \times Q$ is the finite set of *(transition) rules*. We often present a rule $(q,z,q') \in \delta$ as $q \xrightarrow{z} q'$. By

*Adam's (control) states* we mean the elements of $Q_\forall = Q \setminus Q_\exists$. A socn-r-game $\mathcal{N} = (Q, Q_\exists, \delta, p_{win})$ has the *associated r-game*

$$\mathcal{G}_\mathcal{N} = (Q \times \mathbb{N}, Q_\exists \times \mathbb{N}, \rightarrow, \{(p_{win}, 0)\}) \tag{2}$$

where $(q, m) \rightarrow (q', n)$ iff $q \xrightarrow{n-m} q'$ is a rule (in $\delta$). We often write $q(m)$ instead of $(q, m)$ for states of $\mathcal{G}_\mathcal{N}$. We define the problem *SOCNRG* (to decide succinct one-counter net r-games) as follows:

*Instance:* a socn-r-game $\mathcal{N}$ (with integers $z$ in rules $q \xrightarrow{z} q'$ written in binary), and a control state $p_0$.
*Question:* is $p_0(0) \in Win_\exists$ in the game $\mathcal{G}_\mathcal{N}$?

*Remark.* We have defined the target states (in $\mathcal{G}_\mathcal{N}$) to be the singleton set $\{p_{win}(0)\}$. There are other natural variants (e.g., one in [11] defines the target set $\{p(0) \mid p \neq p_0\}$) that can be easily shown to be essentially equivalent.

## 3 EXPSPACE-Completeness of Existential Countdown Games

The EXPSPACE-hardness of SOCNRG was announced in [11], where an idea of a proof is sketched, also using a reference to an involved result [6] (which is further discussed in Sect. 5). Here we give a direct self-contained proof that does not rely on [11] or involved techniques from [6], and that even shows that SOCNRG is EXPSPACE-hard already in the special case that slightly generalizes the countdown games from [17]. (The EXPSPACE-membership follows from [11], but we add a short proof to be self-contained.)

We define a *countdown game* as a socn-r-game $\mathcal{N} = (Q, Q_\exists, \delta, p_{win})$, where in every rule $q \xrightarrow{z} q'$ in $\delta$ we have $z < 0$. The problem CG is defined as follows:

*Instance:* a countdown game $\mathcal{N}$ (with integers in rules written in binary), and an initial configuration $p_0(n)$ where $n \in \mathbb{N}$ ($n$ in binary).
*Question:* is $p_0(n) \in Win_\exists$?

The problem CG (in an equivalent form) was shown EXPTIME-complete in [17].

Here we define an existential version, i.e. the problem ECG:

*Instance:* a countdown game $\mathcal{N}$ and a control state $p_0$.
*Question:* is there some $n \in \mathbb{N}$ such that $p_0(n) \in Win_\exists$?

ECG can be indeed viewed as a subproblem of SOCNRG: given an instance of ECG, it suffices to add a new Eve's state $p_0'$ and rules $p_0' \xrightarrow{1} p_0'$, $p_0' \xrightarrow{0} p_0$; the question then is if $p_0'(0) \in Win_\exists$.

In the rest of this section we prove the following theorem.

**Theorem 1.** *ECG (existential countdown games) is EXPSPACE-complete.*

**EXPSPACE-Hardness of ECG.** We use a "master" reduction. We thus fix an arbitrary language $L \subseteq \Sigma^*$ in EXPSPACE, decided by a (deterministic) Turing machine $M$ in space $2^{p(n)}$ for a fixed polynomial $p$. For any word $w \in \Sigma^*$ there is the respective computation of $M$ using at most $m = 2^{p(n)}$ tape cells (where $n = |w|$), which is accepting iff $w \in L$. We show a construction of a countdown game $\mathcal{N}^M_{w,m}$ with an initial control state $p_0$ such that there exists $k \geq 0$ where $p(k) \in \mathit{Win}_\exists$ if, and only if, $M$ accepts $w$. Moreover, for the fixed $M$, logarithmic space (with respect to $n$) is sufficient to construct $\mathcal{N}^M_{w,m}$. Thus, there is a logspace reduction from the membership problem for $L$ to the problem ECG.

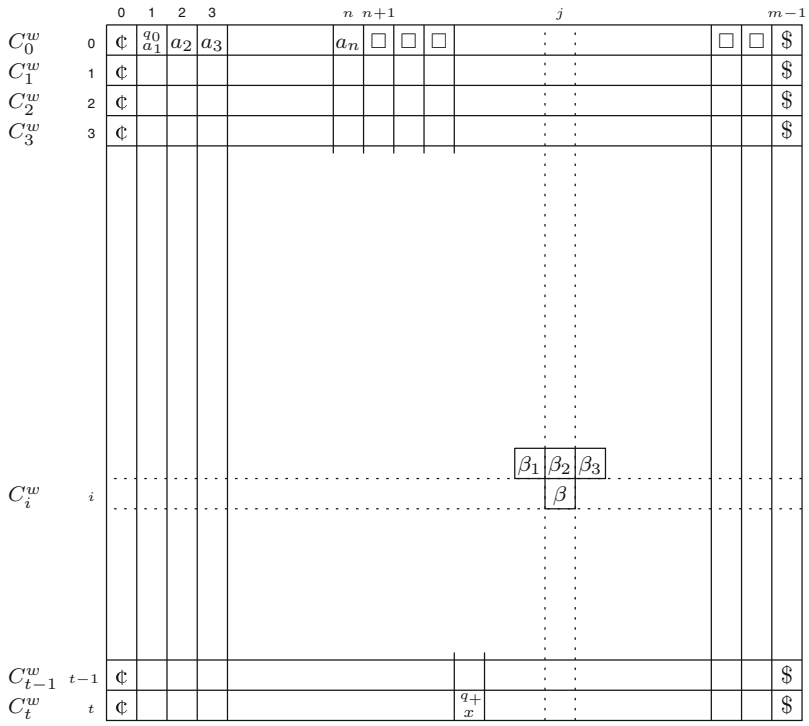


**Fig. 1.** A table of configurations in the computation of $M$ on word $w = a_1 a_2 \dots a_n$

*Construction Informally.* The construction of the countdown game $\mathcal{N}^M_{w,m}$ elaborates an idea that is already present in [4] (in Theorem 3.4) and that was also used, e.g., in [15]. We first present the game informally; this is then formalized in a straightforward technical way.

Figure 1 presents an accepting computation of $M$, on a word $w = a_1 a_2 \dots a_n$; it starts in the initial control state $q_0$ with the head scanning $a_1$. The computation is a sequence of configurations $C_0^w, C_1^w, \dots, C_t^w$, where $C_t^w$ is accepting (since the control state is $q_+$). We assume that $M$ never leaves its input to the left, hence

the tape position 0, which is filled with a special "left sentinel" ¢ for transparency, is never visited. On the other hand, the position $m-1$ of the right sentinel \$ is never visited either; the space complexity of the computation is thus at most $m-2$.

Let us imagine a game where Eve, given $w$, claims that $w$ is accepted by $M$. Nevertheless she does not present a respective accepting computation; she only produces a "row" $r \in \mathbb{N}$ and a "column" $c \in \mathbb{N}$, and a tape-symbol $x$, claiming that if we constructed the computation $C_0^w, C_1^w, \ldots, C_t^w$, then we would find that ($r = t$ and) $C_r^w$ is the accepting configuration and the symbol on position $c$ in $C_r^w$ is $(q_+, x)$ (i.e., the tape symbol is $x$, and the head happens to scan position $c$, and the control state is $q_+$).

Generally, if Eve claims that in the row $i$ and the column $j$ we would find $\beta$ if we constructed the computation (where $\beta$ is either a tape symbol or a tape symbol combined with a control state), she must present a triple $(\beta_1, \beta_2, \beta_3)$ in the previous row $i-1$ as depicted in Fig. 1, and Adam chooses one of symbols $\beta_1, \beta_2, \beta_3$ for the next round; the triple must be consistent with $\beta$ w.r.t. the rules of $M$. If in the row 0 Adam chooses a symbol that is correct in $C_0^w$, then Eve wins (otherwise Adam wins). It is easy to verify that Eve has a winning strategy in this game iff $w$ is accepted by $M$.

We note that the described game uses a pair of number-variables $i, j$, to determine the current cell in the computation table. But we can ask Eve to provide some $m \in \mathbb{N}$ in the beginning (claiming that the head only moves between positions 1 and $m-2$ during the computation); the pair $(i, j)$ can be then represented by the value $z = i \cdot m + j$ (and going from $i$ to $i-1$ amounts to subtract $m$ from $z$). If Eve sometimes claims that $\beta_1 =$ ¢, then the respective column-position is 0, which entails that the respective value $z = i \cdot m + j$ should be divisible by $m$; if Adam doubts this, it can be verified by subtracting $m$ repeatedly. Similarly, if Eve claims $\beta_3 =$ \$, then by subtracting $m-1$ we should get a number divisible by $m$. If Eve claims something else than ¢ in a position corresponding to the ¢-column, or something else than \$ in a position corresponding to the \$-column, then Adam just keeps choosing this column, and Eve's cheating is revealed in the row 0.

*Construction Formally.* Now we formalize the above idea, which is a routine technical work, in fact.

Assume a fixed deterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, \{q_+, q_-\})$, where $Q$ is the set of (control) states, $q_0 \in Q$ the initial state, $q_+ \in Q$ the accepting state, $q_- \in Q$ the rejecting state, $\Sigma$ the input alphabet, $\Gamma \supseteq \Sigma$ the tape alphabet, satisfying $\square \in \Gamma \setminus \Sigma$ for the special blank tape-symbol $\square$, and $\delta : (Q \setminus \{q_+, q_-\}) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ is the transition function.

Putting $\Delta = \Gamma \cup (Q \times \Gamma)$, we define the consistency relation $\vdash \subseteq \Delta^3 \times \Delta$ in a standard way: $(\beta_1, \beta_2, \beta_3) \vdash \beta$ (to be read "$\beta$ is consistent with $(\beta_1, \beta_2, \beta_3)$") if $\beta_i \in Q \times \Gamma$ for at most one $i \in \{1, 2, 3\}$ and the following conditions hold:

- if $\beta_1\beta_2\beta_3 = (q, x)yz$ and $\delta(q, x) = (q', x', d)$, then $\beta = (q', y)$ if $d = +1$ and $\beta = y$ otherwise (i.e., if $d = -1$);

- if $\beta_1\beta_2\beta_3 = x(q,y)z$ and $\delta(q,y) = (q',y',d)$, then $\beta = y'$ (for any $d \in \{-1,+1\}$);
- if $\beta_1\beta_2\beta_3 = xy(q,z)$ and $\delta(q,z) = (q',z',d)$, then $\beta = (q',y)$ if $d = -1$ and $\beta = y$ otherwise;
- if $\beta_1\beta_2\beta_3 = xyz$, then $\beta = y$.

We note that $\vdash$ is a partial function, in fact. By a *configuration* of $M$ we mean a mapping $C : \mathbb{Z} \to \Delta$ where $C(j) \neq \square$ for only finitely many $j \in \mathbb{Z}$ and $C(j) \in Q \times \Gamma$ for precisely one $j \in \mathbb{Z}$, called the *head-position*; if $C(j) = (q_+, x)$ for the head-position $j$ (and $x \in \Gamma$) then $C$ is *accepting*, and if $C(j) = (q_-, x)$ then $C$ is *rejecting*.

We put $C \vdash C'$ (thus overloading the symbol $\vdash$) if $\big(C(j-1), C(j), C(j+1)\big) \vdash C'(j)$ for all $j \in \mathbb{Z}$. This relation $\vdash$ is again a partial function; if $C$ is *final*, i.e. accepting or rejecting, then there is no $C'$ such that $C \vdash C'$.

Given a word $w = a_1 a_2 \cdots a_n \in \Sigma^*$ (hence $|w| = n$), we define the respective initial configuration as $C_0^w$ where $C_0^w(1) = (q_0, a_1)$ if $n \geq 1$ and $C_0^w(1) = (q_0, \square)$ if $n = 0$, $C_0^w(j) = a_j$ for all $j \in [2,n]$, and $C_0^w(j) = \square$ for all $j \leq 0$ and all $j > n$. If $C_i^w$ is not final, then we define $C_{i+1}^w$ so that $C_i^w \vdash C_{i+1}^w$. The *computation on* $w$ is either the finite sequence $C_0^w, C_1^w, C_2^w, \ldots, C_t^w$ where $C_t^w$ is final (accepting or rejecting), or the infinite sequence $C_0^w, C_1^w, C_2^w, \ldots$; formally we put $C_i^w(j) = \bot$ (for $\bot \notin \Delta$) if there is a final $C_t^w$ and $i > t$.

By $L(M)$ we denote the *language accepted by* $M$, i.e. the set $\{w \in \Sigma^* \mid$ the computation on $w$ finishes with an accepting configuration$\}$.

Now we assume that the Turing machine $M$ uses a bounded space for the input $w = a_1 a_2 \ldots a_n$, and in particular that we have $m \in \mathbb{N}$ such that during the computation of $M$ on $w$ the head-position is never outside $[1, m-2]$; for technical convenience and without loss of generality we also assume that $m \geq n \geq 1$ and $m > 3$. We can imagine that the computation is presented as a table depicted in Fig. 1, and the columns 0 and $m-1$ are filled with special symbols ¢ and \$ (where ¢, \$ $\notin \Delta$), i.e., $C_i^w(0) =$ ¢ and $C_i^w(m-1) =$ \$ for each $i \geq 0$. We also extend relation $\vdash$ accordingly to incorporate these special symbols. In particular, whenever $(\beta_1, \beta_2, \beta_3) \vdash \beta$, we have $\beta \in \Delta$, $\beta_1 \in \Delta \cup \{¢\}$, $\beta_2 \in \Delta$, and $\beta_3 \in \Delta \cup \{\$\}$, and we exclude those combinations of $\beta_1, \beta_2, \beta_3$ that would correspond to a move of the head of the machine $M$ to a position containing ¢ or \$.

Given a Turing machine $M$, its input $w = a_1 a_2 \cdots a_n$, and a number $m$, satisfying the above assumptions, we construct a corresponding countdown game

$$\mathcal{N}_{w,m}^M = (\overline{Q}, \overline{Q}_\exists, \delta_\mathcal{N}, p_{win})$$

where

- $\overline{Q}_\exists = \{p_0, p_2, p_{win}, p_{bad}, s_¢, s_\$\} \cup \{s_\beta \mid \beta \in \Delta\}$,
- $\overline{Q}_\forall = \{p_1\} \cup \{s_{(\beta_1,\beta_2,\beta_3)} \mid \beta_i \in \Delta\} \cup \{s_{(¢,\beta_2,\beta_3)} \mid \beta_i \in \Delta\} \cup \{s_{(\beta_1,\beta_2,\$)} \mid \beta_i \in \Delta\}$

(recall that $\overline{Q}_\forall = \overline{Q} \setminus \overline{Q}_\exists$), and the set $\delta_\mathcal{N}$ consists of the rules in Fig. 2 (for all $\beta \in \Delta$, $\beta_1 \in \Delta \cup \{¢\}$, $\beta_2 \in \Delta$, $\beta_3 \in \Delta \cup \{\$\}$).

The relation of the countdown game $\mathcal{N}_{w,m}^M$ to the computation of $M$ on $w$ is stated in the following proposition.

$$p_0 \xrightarrow{-1} s_{(q_+,x)} \quad \text{(where } x \in \Gamma\text{)} \tag{1}$$

$$s_\beta \xrightarrow{-(m-2)} s_{(\beta_1,\beta_2,\beta_3)} \quad \text{(where } (\beta_1,\beta_2,\beta_3) \vdash \beta\text{)} \tag{2}$$

$$s_{(\beta_1,\beta_2,\beta_3)} \xrightarrow{-3} s_{\beta_1} \qquad s_{(\beta_1,\beta_2,\beta_3)} \xrightarrow{-2} s_{\beta_2} \qquad s_{(\beta_1,\beta_2,\beta_3)} \xrightarrow{-1} s_{\beta_3} \tag{3}$$

$$s_\text{¢} \xrightarrow{-m} s_\text{¢} \qquad s_\text{¢} \xrightarrow{-2} p_{win} \qquad s_\$ \xrightarrow{-(m-1)} s_\text{¢} \tag{4}$$

$$s_\beta \xrightarrow{-(2+j)} p_{win} \quad \text{(where } 1 \le j \le n \text{ and } C_0^w(j) = \beta\text{)} \tag{5}$$

$$s_\square \xrightarrow{-(n+1)} p_1 \qquad p_1 \xrightarrow{-(m-n)} p_{bad} \qquad p_1 \xrightarrow{-1} p_2 \tag{6}$$

$$p_2 \xrightarrow{-1} p_2 \qquad p_2 \xrightarrow{-1} p_{win}$$

**Fig. 2.** Rules of $\mathcal{N}_{w,m}^M$

**Proposition 2.** *For the countdown game $\mathcal{N}_{w,m}^M$ there exists $k \ge 0$ such that $p_0(k) \in Win_\exists$ iff the computation of $M$ on $w$ never moves the head out of $[1, m-2]$ and finishes in an accepting configuration.*

*Proof.* The configurations of $\mathcal{N}_{w,m}^M$ of the form $s_\beta(k)$ with $\beta \in \Delta \cup \{\text{¢}, \$\}$ and $k = 2+i\cdot m+j$ where $i \ge 0$ and $0 \le j < m$ correspond to the situation where Eve claims that in the table of configurations of computation of $M$ on $w$ (see Fig. 1) the cell in row $i$ and column $j$ contains symbol $\beta$. We will show that she has a winning strategy from $s_\beta(k)$ exactly when this is the case, i.e., $s_\beta(k) \in Win_\exists$ iff $C_i^w(j) = \beta$. (In our construction, we need to add number 2 to $i \cdot m + j$ to ensure that in every move in the game the value of the counter is decremented by at least 1. It would not be necessary if we allow moves that do not change the counter value.)

To prove that $s_\beta(k) \in Win_\exists$ iff $C_i^w(j) = \beta$ (for $k = 2 + i \cdot m + j$), we start by the following facts that are easy to check (recall that Eve wins iff the configuration $p_{win}(0)$ is reached):

(a) $s_\text{¢}(k) \in Win_\exists$ iff $k = 2 + i \cdot m$ for some $i \in \mathbb{N}$;
(b) $s_\$(k) \in Win_\exists$ iff $k = 2 + i \cdot m + (m-1)$ for some $i \in \mathbb{N}$;
(c) for $1 \le j \le n$, $s_\beta(2+j) \in Win_\exists$ iff $\beta = C_0^w(j)$;
(d) $p_1(k) \in Win_\exists$ iff $2 \le k < m - n$;
(e) for $n < j < m-1$, $s_\beta(2+j) \in Win_\exists$ iff $\beta = \square$.

Assume now that $\beta \in \Delta \cup \{\text{¢}, \$\}$, $i \ge 0$, $0 \le j < m$, and $k = 2 + i \cdot m + j$. To show $s_\beta(k) \in Win_\exists$ iff $C_i^w(j) = \beta$, we proceed by induction on $i$:

- *Base case* $i = 0$: In this case $k = 2 + j$, so we need to show that $s_\beta(2 + j) \in Win_\exists$ iff $C_0^w(j) = \beta$. This follows easily from facts (a), (b), (c), (e) mentioned above because the initial configuration $C_0^w$ consists of symbol ¢ ($j = 0$, fact (a)), the input word with the initial control state ($1 \le j \le n$, fact (c)), blanks ($n < j < m-1$, fact (e)), and symbol \$ ($j = m-1$, fact (b)).
- *Induction step* $i > 0$: If $\beta = \text{¢}$, Eve wins iff $j = 0$ (by fact (a)). Similarly, if $\beta = \$$, she wins iff $j = m - 1$ (by fact (b)).
  Assume now that $\beta \in \Delta$. Eve will lose if she uses any rule from groups (5) or (6) in $s_\beta(k)$, as can be easily checked, and she cannot play rules from

group (4), so she is forced to use some rule from group (2). By playing $s_\beta(k) \xrightarrow{m-2} s_{(\beta_1,\beta_2,\beta_3)}(k')$, where $k' = 2 + (i-1)\cdot m + j + 2$, she chooses a triple $(\beta_1, \beta_2, \beta_3)$ satisfying $(\beta_1, \beta_2, \beta_3) \vdash \beta$, where $\beta_1$, $\beta_2$, $\beta_3$ are symbols supposedly occurring on positions $j-1$, $j$, $j+1$ in configuration $C^w_{i-1}$. If $\beta$ is incorrect (i.e., if $C^w_i(j) \neq \beta$), then at least one of $\beta_1$, $\beta_2$, $\beta_3$ must be also incorrect, and if $\beta$ is correct, then Eve can choose correct $\beta_1$, $\beta_2$, $\beta_3$. Now Adam can challenge some of the symbols $\beta_1$, $\beta_2$, $\beta_3$ by choosing $\ell \in \{1,2,3\}$ and playing $s_{(\beta_1,\beta_2,\beta_3)}(k') \xrightarrow{-(4-\ell)} s_{\beta_\ell}(k'')$, for $k'' = 2+(i-1)\cdot m+j+(\ell-2)$. By the induction hypothesis he thus has a possibility to preclude Eve's win precisely when one of $\beta_1$, $\beta_2$, $\beta_3$ is incorrect. In particular, if $j = 0$ and $\beta \neq \text{¢}$, or if $j = m-1$ and $\beta \neq \$$, then Adam repeatedly chooses $\ell = 2$, thus staying in the same column, and Eve has no possibility to install the correct ¢, resp. \$, in this column anymore.

It is now clear that if the computation of $M$ on $w$ never moves the head outside $[1, m-2]$ and finishes in an accepting configuration $C^w_r$ with $C^w_r(j) = (q_+, x)$ for some $x \in \Gamma$ and $1 \leq j < m-1$, then Eve can force her win in $p_0(k)$ where $k = 3+r\cdot m+j$ by playing $p_0(k) \xrightarrow{-1} s_{(q_+,x)}(2+r\cdot m+j)$. It is also easy to check that if the computation moves the head outside $[1, m-2]$ or is not accepting, then Eve cannot force her win from $p_0(k)$ for any $k \in \mathbb{N}$. □

We note that the control states in $\mathcal{N}^M_{w,m}$ are determined by $M$. The rules of $\mathcal{N}^M_{w,m}$, except those in group (5), depend only on $M$ and "parameters" $n = |w|$ and $m$.

To finish the EXPSPACE-hardness part of the proof of Theorem 1, we assume an arbitrary fixed language $L$ in EXPSPACE. There is thus a Turing machine $M$ and a polynomial $p$ such that $M$ accepts $L$ and the head-position in the computation of $M$ on any $w$ (in the alphabet of $L$) never moves out of the interval $[1, m-2]$ where $m = 2^{p(n)}$ for $n = |w|$. Given $w$, it is straightforward to construct $\mathcal{N}^M_{w,m}$, by filling the parameters $n, m$, and the rules in group (5), into a fixed scheme. Since $m$ can be presented in binary by using $p(n)+1$ bits, we can construct $\mathcal{N}^M_{w,m}$ in logarithmic work-space (from a given $w$).

**ECG Is in EXPSPACE.** Given an ECG-instance $\mathcal{N} = (Q, Q_\exists, \delta, p_{win})$, $p_0$, we can stepwise construct $W(0), W(1), W(2), \ldots$ where $W(j) = (Q \times \{j\}) \cap Win_\exists$. For determining $W(n)$ it suffices to know the segment $W(n-\text{M}), W(n-\text{M}+1), \ldots, W(n-1)$ where M is the maximum value by which the counter can be decremented in one step. By the pigeon-hole principle, such exponential-size segments must repeat inside $W(0), W(1), \ldots, W(2^{|Q|\text{M}})$. Hence if there is $n$ such that $p_0(n) \in Win_\exists$, then there is such $n$ of double-exponential size; exponential space is thus sufficient for finding such $n$.

## 4 Reachability Game Reduces to (Bi)simulation Game

We show a reduction for general r-games, and then apply it to the case of socn-r-games. This yields a log-space reduction of SOCNRG to behavioural relations.

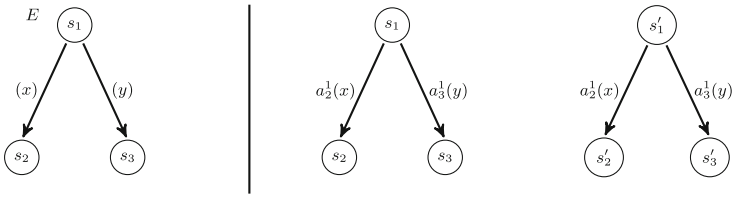


**Fig. 3.** Eve's state $s_1$ in $\mathcal{G}$ (left) is mimicked by the pair $(s_1, s'_1)$ in $\mathcal{L}(\mathcal{G})$ (right); it is thus Attacker who chooses $(s_2, s'_2)$ or $(s_3, s'_3)$ as the next current pair.

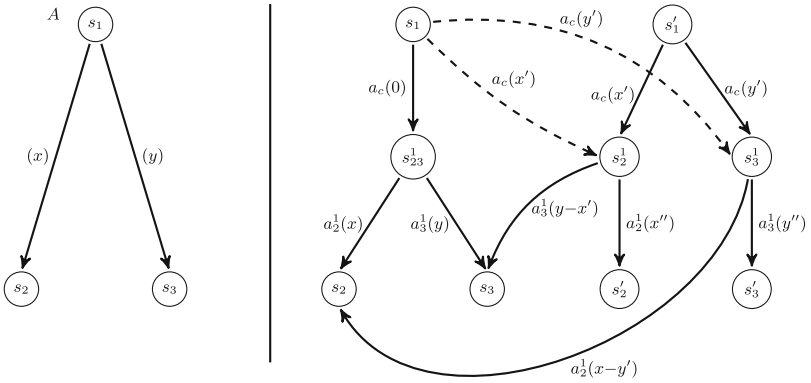


**Fig. 4.** In $(s_1, s'_1)$ it is, in fact, Defender who chooses $(s_2, s'_2)$ or $(s_3, s'_3)$ (when Attacker avoids pairs with equal states); to take the counter-changes into account correctly, we put $x' = \min\{x, 0\}$, $x'' = \max\{x, 0\}$, and $y' = \min\{y, 0\}$, $y'' = \max\{y, 0\}$ (hence $x = x' + x''$ and $y = y' + y''$). (The dashed edges are viewed as the other edges.)

### 4.1 Reduction in a General Framework

We assume an r-game $\mathcal{G}$, and below we define a "mimicking" LTS $\mathcal{L}(\mathcal{G})$. In illustrating Figs. 3 and 4 we now ignore the bracketed parts of transition-labels; hence, e.g., in Fig. 3 we can see the transition $s_1 \to s_2$ in $\mathcal{G}$ on the left and the (corresponding) transitions $s_1 \xrightarrow{a_2^1} s_2$ and $s'_1 \xrightarrow{a_2^1} s'_2$ in $\mathcal{L}(\mathcal{G})$ on the right. We also use an informal (bi)simulation game terminology: in a current pair of states (e.g., in $(s_1, s'_1)$ in $\mathcal{L}(\mathcal{G})$), Attacker performs a transition on one side, and Defender responds with a "same-label" transition on the other side, which yields a new current pair; in the bisimulation game Attacker chooses the sides freely, in the simulation game he must always choose the left-hand side.

So let $\mathcal{G} = (V, V_\exists, \to, \mathcal{T})$ be an r-game, where $V_\forall = V \setminus V_\exists$; we define $\mathcal{L}(\mathcal{G}) = (S, Act, (\xrightarrow{a})_{a \in Act})$ as follows. We put

$$S = V \cup V' \cup \{\langle s, \bar{s}\rangle \mid s \in V_\forall, s \to \bar{s}\} \cup \{\langle s, X\rangle \mid s \in V_\forall, X = \{\bar{s} \mid s \to \bar{s}\} \neq \emptyset\}$$

where $V' = \{s' \mid s \in V\}$ is a "copy" of $V$. (In Fig. 4 we write, e.g., $s_3^1$ instead of $\langle s_1, s_3\rangle$, and $s_{23}^1$ instead of $\langle s_1, \{s_2, s_3\}\rangle$.)

We put $Act = \{a_c, a_{win}\} \cup \{a_{\langle s,\bar{s}\rangle} \mid s \to \bar{s}\}$ and define $\xrightarrow{a}$ for $a \in Act$ as follows. If $s \in V_\exists$ and $s \to \bar{s}$, then $s \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}$ and $s' \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}'$ (in Fig. 3 we write, e.g., $a_3^1$ instead of $a_{\langle s_1,s_3\rangle}$). If $s \in V_\forall$ and $X = \{\bar{s} \mid s \to \bar{s}\} \neq \emptyset$, then:

(a) $s \xrightarrow{a_c} \langle s, X\rangle$, and $s \xrightarrow{a_c} \langle s, \bar{s}\rangle$, $s' \xrightarrow{a_c} \langle s, \bar{s}\rangle$ for all $\bar{s} \in X$ (cf. Fig. 4 where $s = s_1$ and $X = \{s_2, s_3\}$ and consider dashed edges as normal edges; $a_c$ is a "choice-action");

(b) for each $\bar{s} \in X$ we have $\langle s, X\rangle \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}$ and $\langle s, \bar{s}\rangle \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}'$; moreover, for each $\bar{\bar{s}} \in X \setminus \{\bar{s}\}$ we have $\langle s, \bar{s}\rangle \xrightarrow{a_{\langle s,\bar{\bar{s}}\rangle}} \bar{\bar{s}}$ (e.g., in Fig. 4 we thus have $s_2^1 \xrightarrow{a_2^1} s_2'$ and $s_2^1 \xrightarrow{a_3^1} s_3$).

For each $s \in \mathcal{T}$ we have $s \xrightarrow{a_{win}} s$ (for special $a_{win}$ that is not enabled in $s'$).

**Lemma 3.** *For an r-game $\mathcal{G} = (V, V_\exists, \to, \mathcal{T})$ and its "mimicking" LTS $\mathcal{L}(\mathcal{G}) = (S, Act, (\xrightarrow{a})_{a\in Act})$, the following conditions hold for every $s \in V$ and every relation $\rho$ satisfying $\sim \subseteq \rho \subseteq \preceq$:*

*(a) if $s \in Win_\exists$ (in $\mathcal{G}$), then $s \not\preceq s'$ (in $\mathcal{L}(\mathcal{G})$) and thus $(s, s') \notin \rho$;*
*(b) if $s \notin Win_\exists$, then $s \sim s'$ and thus $(s, s') \in \rho$.*

*Proof.* (a) For the sake of contradiction suppose that there is $s \in Win_\exists$ such that $s \preceq s'$; we consider such $s$ with the least rank. We note that $\text{RANK}(s) > 0$, since $s \in \mathcal{T}$ entails $s \not\preceq s'$ due to the transition $s \xrightarrow{a_{win}} s$. If $s \in V_\exists$, then let $s \to \bar{s}$ be a rank-reducing transition. Attacker's move $s \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}$, from the pair $(s, s')$, must be responded with $s' \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}'$; but we have $\bar{s} \not\preceq \bar{s}'$ by the "least-rank" assumption, which contradicts with the assumption $s \preceq s'$. If $s \in V_\forall$, then $X = \{\bar{s} \mid s \to \bar{s}\}$ is nonempty (since $s \in Win_\exists$) and $\text{RANK}(\bar{s}) < \text{RANK}(s)$ for all $\bar{s} \in X$. For the pair $(s, s')$ we now consider Attacker's move $s \xrightarrow{a_c} \langle s, X\rangle$. Defender can choose $s' \xrightarrow{a_c} \langle s, \bar{s}\rangle$ for any $\bar{s} \in X$ (recall that $\text{RANK}(\bar{s}) < \text{RANK}(s)$). In the current pair $(\langle s, X\rangle, \langle s, \bar{s}\rangle)$ Attacker can play $\langle s, X\rangle \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}$, and this must be responded by $\langle s, \bar{s}\rangle \xrightarrow{a_{\langle s,\bar{s}\rangle}} \bar{s}'$. But we again have $\bar{s} \not\preceq \bar{s}'$ by the "least-rank" assumption, which contradicts with $s \preceq s'$.

(b) It is easy to verify that the following set is a bisimulation in $\mathcal{L}(\mathcal{G})$:

$$I \cup \{(s, s') \mid s \in V \setminus Win_\exists\} \cup \{(\langle s, X\rangle, \langle s, \bar{s}\rangle) \mid s \in V_\forall \setminus Win_\exists, \bar{s} \in V \setminus Win_\exists\}$$

where $I = \{(s, s) \mid s \in S\}$. □

We note that the transitions corresponding to the dashed edges in Fig. 4 could be omitted if we only wanted to show that $s \in Win_\exists$ iff $s \not\preceq s'$.

### 4.2 SOCNRG Reduces to Behavioural Relations on SOCNs

We now note that the LTS $\mathcal{L}(\mathcal{G}_\mathcal{N})$ "mimicking" the r-game $\mathcal{G}_\mathcal{N}$ associated with a socn-r-game $\mathcal{N}$ (recall (2)) can be presented as $\mathcal{L}_{\mathcal{N}'}$ for a SOCN $\mathcal{N}'$ (recall (1)) that is efficiently constructible from $\mathcal{N}$:

**Lemma 4.** *There is a log-space algorithm that, given a socn-r-game $\mathcal{N}$, constructs a SOCN $\mathcal{N}'$ such that the LTSs $\mathcal{L}(\mathcal{G}_{\mathcal{N}})$ and $\mathcal{L}_{\mathcal{N}'}$ are isomorphic.*

*Proof.* We again use Figs. 3 and 4 for illustration; now $s_i$ are viewed as control states and the bracketed parts of edge-labels are counter-changes (in binary).

Given a socn-r-game $\mathcal{N} = (Q, Q_\exists, \delta, p_{win})$, we first consider the r-game $\mathcal{N}^{csg} = (Q, Q_\exists, \rightarrow, \{p_{win}\})$ ("the control-state game of $\mathcal{N}$") arising from $\mathcal{N}$ by *forgetting the counter-changes*; hence $q \rightarrow \bar{q}$ iff there is a rule $q \xrightarrow{z} \bar{q}$. In fact, we will assume that there is at most one rule $q \xrightarrow{z} \bar{q}$ in $\delta$ (of $\mathcal{N}$) for any pair $(q, \bar{q}) \in Q \times Q$; this can be achieved by harmless modifications.

We construct the (finite) LTS $\mathcal{L}(\mathcal{N}^{csg})$ ("mimicking" $\mathcal{N}$). Hence each $q \in Q$ has the copies $q, q'$ in $\mathcal{L}(\mathcal{N}^{csg})$, and other states are added (as also depicted in Fig. 4 where $s_i$ are now in the role of control states); there are also the respective labelled transitions in $\mathcal{L}(\mathcal{N}^{csg})$, with labels $a_{\langle q,\bar{q}\rangle}$, $a_c$, $a_{win}$.

It remains to add the counter changes (integer increments and decrements in binary), to create the required SOCN $\mathcal{N}'$. For $q \in Q_\exists$ this adding is simple, as depicted in Fig. 3: if $q \xrightarrow{z} \bar{q}$ (in $\mathcal{N}$), then we simply extend the label $a_{\langle q,\bar{q}\rangle}$ in $\mathcal{L}(\mathcal{N}^{csg})$ with $z$; for $q \xrightarrow{a_{\langle q,\bar{q}\rangle}} \bar{q}$ and $q' \xrightarrow{a_{\langle q,\bar{q}\rangle}} \bar{q}'$ in $\mathcal{L}(\mathcal{N}^{csg})$ we get $q \xrightarrow{a_{\langle q,\bar{q}\rangle},z} \bar{q}$ and $q' \xrightarrow{a_{\langle q,\bar{q}\rangle},z} \bar{q}'$ in $\mathcal{N}'$.

For $q \in Q_\forall$ (where $Q_\forall = Q \setminus Q_\exists$) it is tempting to the same, i.e. to extend the label $a_{\langle q,\bar{q}\rangle}$ with $z$ when $q \xrightarrow{z} \bar{q}$, and extend $a_c$ with 0. But this might allow cheating for Defender: she could thus mimic choosing a transition $q(k) \xrightarrow{x} \bar{q}(k+x)$ even if $k + x < 0$. This is avoided by the modification that is demonstrated in Fig. 4 (by $x = x' + x''$, etc.); put simply: Defender must immediately prove that the transition she is choosing to mimic is indeed performable. Formally, if $X = \{\bar{q} \mid q \rightarrow \bar{q}\} \neq \emptyset$ (in $\mathcal{L}(\mathcal{N}^{csg})$), then in $\mathcal{N}'$ we put $q \xrightarrow{a_c,0} \langle q, X\rangle$ and $\langle q, X\rangle \xrightarrow{a_{\langle q,\bar{q}\rangle},z} \bar{q}$ for each $q \xrightarrow{z} \bar{q}$ (in $\mathcal{N}$); for each $q \xrightarrow{z} \bar{q}$ we also define $z' = \min\{z, 0\}$, $z'' = \max\{z, 0\}$ and put $q' \xrightarrow{a_c,z'} \langle q, \bar{q}\rangle$, $\langle q, \bar{q}\rangle \xrightarrow{a_{\langle q,\bar{q}\rangle},z''} \bar{q}'$. Then for any pair $q \xrightarrow{\bar{z}} \bar{q}$, $q \xrightarrow{\bar{\bar{z}}} \bar{\bar{q}}$ where $\bar{q} \neq \bar{\bar{q}}$ we put $\langle q, \bar{q}\rangle \xrightarrow{a_{\langle q,\bar{q}\rangle},\bar{\bar{z}}-\bar{z}'} \bar{\bar{q}}$.

Finally, $p_{win} \xrightarrow{a_{win}} p_{win}$ in $\mathcal{L}(\mathcal{N}^{csg})$ is extended to $p_{win} \xrightarrow{a_{win},0} p_{win}$ in $\mathcal{N}'$. □

Recalling the EXPSPACE-hardness of SOCNRG (from [11] or from Theorem 1), Lemmas 3 and 4 yield:

**Theorem 5.** *For succinct labelled one-counter nets (SOCNs), deciding any relation containing bisimulation equivalence and contained in simulation preorder is EXPSPACE-hard.*

## 5 Additional Remarks

Theorem 5 shows the hardness, but simulation preorder on succinct one-counter nets and bisimulation equivalence (even) on succinct one-counter automata are also in EXPSPACE (as follows by the membership of their unary versions in PSPACE), hence they are EXPSPACE-complete. Hunter [11] also shows that various extensions of the problem SOCNRG are in EXPSPACE as well.

We recall that our EXPSPACE-hardness proof of ECG (in Sect. 3) is, in fact, a particular instance of a simple general method. A slight modification also yields a proof of EXPTIME-hardness of countdown games that is an alternative to the proof in [17].

One particular application of countdown games was shown by Kiefer [18] who modified them to show EXPTIME-hardness of bisimilarity on BPA processes. Our EXPSPACE-complete modification does not seem easily implementable by BPA processes, hence the EXPTIME-hardness result in [18] has not been improved here. (The known upper bound for bisimilarity on BPA is 2-EXPTIME.)

We have not discussed the upper bounds here. The proofs in [1,10,14] reveal a periodicity of simulation preorder, captured by "linear-belt" theorems. It is worth to note that the period of a simulation-belt can be double-exponential in the size of the respective succinct one-counter net. This is derivable by recalling the reduction from exponential-space Turing machines and by noting that we can choose a machine $M$ such that for every $n$ there is an accepted word of length $n$ for which $M$ performs $2^{2^{\Omega(n)}}$ steps.

Finally we mention that the involved result in [6] shows that, given any fixed language $L$ in EXPSPACE, for any word $w$ (in the alphabet of $L$) we can construct a succinct one-counter automaton that performs a computation which is accepting iff $w \in L$. Such a computation needs to access concrete bits in the (reversed) binary presentation of the counter value. A straightforward direct access to such bits is destructive (the counter value is lost after the bit is read) but this can be avoided: instead of a "destructive reading" the computation just "guesses" the respective bits, and it is forced to guess correctly by a carefully constructed CTL formula that is required to be satisfied by the computation. This result is surely deeper than the EXPSPACE-hardness of existential countdown games, though the former does not seem to entail the latter immediately.

## References

1. Abdulla, P.A., Čerāns, K.: Simulation is decidable for one-counter nets. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 253–268. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055627
2. Böhm, S., Göller, S., Jančar, P.: Equivalence of deterministic one-counter automata is NL-complete. In: STOC 2013, pp. 131–140. ACM (2013)
3. Böhm, S., Göller, S., Jančar, P.: Bisimulation equivalence and regularity for real-time one-counter automata. J. Comput. Syst. Sci. **80**(4), 720–743 (2014). Preliminary versions appeared at CONCUR 2010 and MFCS 2011

4. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. J. ACM **28**(1), 114–133 (1981)
5. Demri, S., Lazić, R., Sangnier, A.: Model checking freeze LTL over one-counter automata. In: Amadio, R. (ed.) FoSSaCS 2008. LNCS, vol. 4962, pp. 490–504. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78499-9_34
6. Göller, S., Haase, C., Ouaknine, J., Worrell, J.: Model checking succinct and parametric one-counter automata. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010 Part II. LNCS, vol. 6199, pp. 575–586. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14162-1_48
7. Göller, S., Lohrey, M.: Branching-time model checking of one-counter processes. In: STACS 2010. LIPIcs, vol. 5, pp. 405–416. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
8. Göller, S., Mayr, R., To, A.W.: On the computational complexity of verifying one-counter processes. In: LICS 2009, pp. 235–244. IEEE Computer Society (2009)
9. Haase, C., Kreutzer, S., Ouaknine, J., Worrell, J.: Reachability in succinct and parametric one-counter automata. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 369–383. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04081-8_25
10. Hofman, P., Lasota, S., Mayr, R., Totzke, P.: Simulation problems over one-counter nets. Log. Methods Comput. Sci. **12**(1), 46 pp. (2016). Preliminary versions appeared at FSTTCS 2013, LICS 2013, and in Totzke's Ph.D. thesis (2014)
11. Hunter, P.: Reachability in succinct one-counter games. In: Bojańczyk, M., Lasota, S., Potapov, I. (eds.) RP 2015. LNCS, vol. 9328, pp. 37–49. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24537-9_5
12. Jančar, P., Kučera, A., Moller, F.: Simulation and bisimulation over one-counter processes. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 334–345. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46541-3_28
13. Jančar, P., Kučera, A., Moller, F., Sawa, Z.: DP lower bounds for equivalence-checking and model-checking of one-counter automata. Inf. Comput. **188**(1), 1–19 (2004)
14. Jančar, P., Moller, F., Sawa, Z.: Simulation problems for one-counter machine. In: Pavelka, J., Tel, G., Bartošek, M. (eds.) SOFSEM 1999. LNCS, vol. 1725, pp. 404–413. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-47849-3_28
15. Jančar, P., Sawa, Z.: A note on emptiness for alternating finite automata with a one-letter alphabet. Inf. Process. Lett. **104**(5), 164–167 (2007)
16. Jančar, P., Srba, J.: Undecidability of bisimilarity by defender's forcing. J. ACM **55**(1), 5:1–5:26 (2008)
17. Jurdzinski, M., Sproston, J., Laroussinie, F.: Model checking probabilistic timed automata with one or two clocks. Log. Methods Comput. Sci. **4**(3), 28 pp. (2008)
18. Kiefer, S.: BPA bisimilarity is EXPTIME-hard. Inf. Process. Lett. **113**(4), 101–106 (2013)
19. Kučera, A.: Efficient verification algorithms for one-counter processes. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 317–328. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45022-X_28
20. Mayr, R.: Undecidability of weak bisimulation equivalence for 1-counter processes. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 570–583. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45061-0_46
21. Reichert, J.: On the complexity of counter reachability games. Fundam. Inform. **143**(3–4), 415–436 (2016)

22. Serre, O.: Parity games played on transition graphs of one-counter processes. In: Aceto, L., Ingólfsdóttir, A. (eds.) FoSSaCS 2006. LNCS, vol. 3921, pp. 337–351. Springer, Heidelberg (2006). https://doi.org/10.1007/11690634_23
23. Srba, J.: Beyond language equivalence on visibly pushdown automata. Log. Methods Comput. Sci. **5**(1), 22 pp. (2009)
24. Valiant, L.G., Paterson, M.: Deterministic one-counter automata. J. Comput. Syst. Sci. **10**(3), 340–350 (1975)