One-Counter Automata with Counter Observability

Benedikt Bollig

LSV, ENS Cachan, CNRS, Inria, Université Paris-Saclay, France
bollig@lsv.fr

Abstract

In a one-counter automaton (OCA), one can produce a letter from some finite alphabet, increment and decrement the counter by one, or compare it with constants up to some threshold. It is well-known that universality and language inclusion for OCAs are undecidable. In this paper, we consider OCAs with counter observability: Whenever the automaton produces a letter, it outputs the current counter value along with it. Hence, its language is now a set of words over an infinite alphabet. We show that universality and inclusion for that model are PSPACE-complete, thus no harder than the corresponding problems for finite automata. In fact, by establishing a link with visibly one-counter automata, we show that OCAs with counter observability are effectively determinizable and closed under all boolean operations. Moreover, it turns out that they are expressively equivalent to strong automata, in which transitions are guarded by MSO formulas over the natural numbers with successor.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases One-counter automata, inclusion checking, observability, visibly one-counter automata, strong automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.20

1 Introduction

One-counter automata (OCAs) are a fundamental model of infinite-state systems. Their expressive power resides between finite automata and pushdown automata. Unlike finite automata, however, OCAs are not robust: They lack closure under complementation and have an undecidable universality, equivalence, and inclusion problem [12, 14]. Several directions to overcome this drawback have been taken. One may underapproximate the above decision problems in terms of bisimilarity [15] or overapproximate the system behavior by a finite-state abstraction, e.g., in terms of the downward closure or preserving the Parikh image [25, 20].

In this paper, we consider a new and simple way of obtaining a robust model of one-counter systems. Whenever the automaton produces a letter from a finite alphabet Σ, it will also output the *current* counter value along with it (transitions that manipulate the counter are not concerned). Hence, its language is henceforth a subset of $(\Sigma \times \mathbb{N})^*$. For obvious reasons, we call this variant *OCAs with counter observability*. We will show that, under the observability semantics, OCAs form a robust automata model: They are closed under all boolean operations. Moreover, their universality and inclusion problem are in PSPACE and, as a simple reduction from universality for finite automata shows, PSPACE-complete.

These results may come as a surprise given that universality for OCAs is undecidable and introducing counter observability seems like an extension of OCAs. But, actually, the problem becomes quite different. The fact that a priori hidden details from a run (in terms of the counter values) are revealed makes the model more robust and the decision problems easier. Note that this is also what happens in input-driven/visibly pushdown automata [16, 3]

36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016).
Editors: Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen; Article No. 20; pp. 20:1–20:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

or their restriction of visibly OCAs [5, 22]. They all recognize languages over a *finite* alphabet and the stack/counter operation can be deduced from the letter that is read. Interestingly, our proofs establish a link between the observability semantics and visibly OCAs. This link is not immediate and relies on a couple of technical lemmas. However, it somehow explains the robustness of OCAs under the observability semantics.

It is worth noting that revealing details from a system configuration does not always help, quite the contrary: Though timed automata and counter automata are closely related [13], the universality problem of timed automata is decidable only if time stamps are excluded from the semantics [1].

Note that it is not only for the pure fact that we obtain a robust model that we consider counter observability. Counter values usually have a *meaning*, such as energy level, value of a variable, or number of items yet to be produced (cf. Example 2). In those contexts, it is natural to include them in the semantics, just like including time stamps in timed automata.

Apart from the connection with visibly OCAs, another model closely related to ours is that of *strong automata* [9]. Strong automata operate on infinite alphabets and were introduced as an extension of *symbolic automata* [6, 10]. Essentially, a transition of a strong automaton is labeled with a formula from monadic second-order (MSO) logic over some infinite structure, say $(\mathbb{N}, +1)$. In fact, the formula has two free first-order variables so that it defines a binary relation over $\mathbb{N}$. This relation is interpreted as a constraint between successive letters from the infinite alphabet. We will show that OCAs with the observability semantics and strong automata over $(\mathbb{N}, +1)$ (extended by a component for the finite alphabet Σ) are expressively equivalent. This underpins a certain naturalness of the observability semantics. Note that the universality and the inclusion problem have been shown decidable for strong automata over $(\mathbb{N}, +1)$ [9]. However, strong automata do not allow us to derive any elementary complexity upper bounds. In fact, our model can be seen as an operational counterpart of strong automata over $(\mathbb{N}, +1)$.

The outline of the paper is as follows. Section 2 defines OCAs and their different semantics. Section 3 relates the observability semantics with visibly OCAs and shows that, under the new semantics, OCAs are closed under boolean operations and have a PSPACE-complete universality and inclusion problem. In Section 4, we show expressive equivalence of strong automata and OCAs with counter observability. We conclude in Section 5. Omitted proofs or proof details can be found in the full version of this paper, which is available at the following link: `https://arxiv.org/abs/1602.05940`

2 One-Counter Automata with Counter Observability

For $n \in \mathbb{N} = \{0, 1, 2, \ldots\}$, we set $[n] := \{1, \ldots, n\}$ and $[n]_0 := \{0, 1, \ldots, n\}$. Given an alphabet Σ, the set of finite words over Σ, including the empty word ε, is denoted by Σ^*.

2.1 One-Counter Automata and Their Semantics

We consider ordinary one-counter automata over some nonempty finite alphabet Σ. In addition to a finite-state control and transitions that produce a letter from Σ, they have a counter that can be incremented, decremented, or tested for values up to some threshold $m \in \mathbb{N}$ (as defined in [5]). Accordingly, the set of *counter operations* is $\mathsf{Op} = \{\uparrow, \downarrow\}$, where $\uparrow$ stands for "increment the counter by one" and $\downarrow$ for "decrement the counter by one". A transition is of the form (q, k, σ, q') where q, q' are states, $k \in [m]_0$ is a counter test, and $\sigma \in \Sigma \cup \mathsf{Op}$. It leads from q to q', while σ either produces a letter from Σ or modifies the

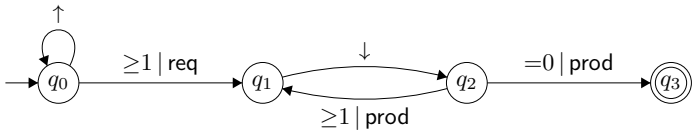

Figure 1 A one-counter automaton with threshold $m = 1$.

counter. However, the transition can only be taken if the current counter value $x \in \mathbb{N}$ satisfies $k = \min\{x, m\}$. That is, counter values can be checked against any number strictly below m or for being at least m. In particular, if $m = 1$, then we deal with the classical definition of one-counter automata, which only allows for zero and non-zero tests.

▶ **Definition 1** (OCA, cf. [5]). A *one-counter automaton* (or simply OCA) is a tuple $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ where Q is a finite set of *states*, Σ is a nonempty finite alphabet (disjoint from Op), $\iota \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *final states*, $m \in \mathbb{N}$ is the *threshold*, and $\Delta \subseteq Q \times [m]_0 \times (\Sigma \cup \mathsf{Op}) \times Q$ is the *transition relation*. We also say that $\mathcal{A}$ is an m-OCA. Its size is defined as $|Q| + |\Sigma| + m + |\Delta|$.

An OCA $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ can have several different semantics:

- $L_{oca}(\mathcal{A}) \subseteq \Sigma^*$ is the classical semantics when $\mathcal{A}$ is seen as an ordinary OCA.
- $L_{vis}(\mathcal{A}) \subseteq (\Sigma \cup \mathsf{Op})^*$ is the *visibly semantics* where, in addition to the letters from Σ, all counter movements are made apparent.
- $L_{obs}(\mathcal{A}) \subseteq (\Sigma \times \mathbb{N})^*$ is the *semantics with counter observability* where the current counter value is output each time a Σ-transition is taken.

We define all three semantics in one go. Let $\mathit{Conf}_{\mathcal{A}} := Q \times \mathbb{N}$ be the set of *configurations* of $\mathcal{A}$. In a configuration (q, x), q is the current state and x is the current counter value. The *initial* configuration is $(\iota, 0)$, and a configuration (q, x) is *final* if $q \in F$.

We determine a global transition relation $\Longrightarrow_{\mathcal{A}} \subseteq \mathit{Conf}_{\mathcal{A}} \times ((\Sigma \times \mathbb{N}) \cup \mathsf{Op}) \times \mathit{Conf}_{\mathcal{A}}$. For two configurations $(q, x), (q', x') \in \mathit{Conf}_{\mathcal{A}}$ and $\tau \in (\Sigma \times \mathbb{N}) \cup \mathsf{Op}$, we have $(q, x) \overset{\tau}{\Longrightarrow}_{\mathcal{A}} (q', x')$ if one of the following holds:

- $\tau = \uparrow$ and $x' = x + 1$ and $(q, \min\{x, m\}, \uparrow, q') \in \Delta$, or
- $\tau = \downarrow$ and $x' = x - 1$ and $(q, \min\{x, m\}, \downarrow, q') \in \Delta$, or
- $x' = x$ and there is $a \in \Sigma$ such that $\tau = (a, x)$ and $(q, \min\{x, m\}, a, q') \in \Delta$.

A *partial run* of $\mathcal{A}$ is a sequence $\rho = (q_0, x_0) \overset{\tau_1}{\Longrightarrow}_{\mathcal{A}} (q_1, x_1) \overset{\tau_2}{\Longrightarrow}_{\mathcal{A}} \dots \overset{\tau_n}{\Longrightarrow}_{\mathcal{A}} (q_n, x_n)$, with $n \geq 0$. If, in addition, (q_0, x_0) is the initial configuration, then we say that ρ is a *run*. We call ρ *accepting* if its last configuration (q_n, x_n) is final.

Now, the semantics of $\mathcal{A}$ that we consider depends on what we would like to extract from $\mathit{trace}(\rho) := \tau_1 \dots \tau_n \in ((\Sigma \times \mathbb{N}) \cup \mathsf{Op})^*$. We let (given $(a, x) \in \Sigma \times \mathbb{N}$):

- $\mathit{oca}((a, x)) = a$ and $\mathit{oca}(\uparrow) = \mathit{oca}(\downarrow) = \varepsilon$
- $\mathit{vis}((a, x)) = a$ and $\mathit{vis}(\uparrow) = \uparrow$ and $\mathit{vis}(\downarrow) = \downarrow$
- $\mathit{obs}((a, x)) = (a, x)$ and $\mathit{obs}(\uparrow) = \mathit{obs}(\downarrow) = \varepsilon$

Moreover, we extend each such mapping $\eta \in \{\mathit{oca}, \mathit{vis}, \mathit{obs}\}$ to $\tau_1 \dots \tau_n \in ((\Sigma \times \mathbb{N}) \cup \mathsf{Op})^*$ letting $\eta(\tau_1 \dots \tau_n) := \eta(\tau_1) \cdot \ldots \cdot \eta(\tau_n)$. Note that, hereby $u \cdot \varepsilon = \varepsilon \cdot u = u$ for any word u.

Finally, we let $L_\eta(\mathcal{A}) = \{\eta(\mathit{trace}(\rho)) \mid \rho \text{ is an accepting run of } \mathcal{A}\}$.

▶ **Example 2.** Consider the 1-OCA $\mathcal{A}$ from Figure 1 over $\Sigma = \{\mathsf{req}, \mathsf{prod}\}$. For readability, counter tests 0 and 1 are written as $=0$ and ≥ 1, respectively. A transition without counter test stands for two distinct transitions, one for $=0$ and one for ≥ 1 (i.e., the counter value may actually be arbitrary). When looking at the semantics $L_{obs}(\mathcal{A})$, i.e., with counter

observability, we can think of (req, n) signalizing that the production of $n \geq 1$ items is required (where n is the current counter value). Moreover, prod indicates that an item has been produced so that, along a run, the counter value represents the number of items yet to be produced. It is thus natural to include it in the semantics. Concretely, we have the following:

- $L_{oca}(\mathcal{A}) = \{\mathsf{req}\ \mathsf{prod}^n \mid n \geq 1\}$
- $L_{vis}(\mathcal{A}) = \{\uparrow^n \mathsf{req}\ (\downarrow \mathsf{prod})^n \mid n \geq 1\}$
- $L_{obs}(\mathcal{A}) = \{(\mathsf{req}, n)(\mathsf{prod}, n-1)(\mathsf{prod}, n-2)\dots(\mathsf{prod}, 0) \mid n \geq 1\}$

Apparently, $L_{vis}(\mathcal{A})$ and $L_{obs}(\mathcal{A})$ are the only meaningful semantics in the context described above.

▶ Remark. Visibly OCAs [5, 22] usually allow for general input alphabets Γ, which are partitioned into $\Gamma = \Gamma_{\text{inc}} \uplus \Gamma_{\text{dec}} \uplus \Gamma_{\text{nop}}$ so that every $\gamma \in \Gamma$ is associated with a unique counter operation (or "no counter operation" if $\gamma \in \Gamma_{\text{nop}}$). In fact, we consider here (wrt. the visibly semantics) a particular case where $\Gamma = \Sigma \cup \mathsf{Op}$ with $\Gamma_{\text{inc}} = \{\uparrow\}$, $\Gamma_{\text{dec}} = \{\downarrow\}$, and $\Gamma_{\text{nop}} = \Sigma$.

2.2 Standard Results for OCAs

Let us recall some well-known results for classical OCAs and visibly OCAs. For $\eta \in \{oca, vis, obs\}$, the *nonemptiness problem for OCAs* wrt. the η-semantics is defined as follows: Given an OCA $\mathcal{A}$, do we have $L_\eta(\mathcal{A}) \neq \emptyset$? Of course, this reduces to a reachability problem that is independent of the actual choice of the semantics:

▶ **Theorem 3** ([24]). *The nonemptiness problem for OCAs is NL-complete, wrt. any of the three semantics.*

However, the universality (and, therefore, inclusion) problem for classical OCAs is undecidable:

▶ **Theorem 4** ([12, 14]). *The following problem is undecidable: Given an OCA $\mathcal{A}$ with alphabet Σ, do we have $L_{oca}(\mathcal{A}) = \Sigma^*$?*

In this paper, we show that universality and inclusion are decidable when considering counter observability. To do so, we make use of the theory of the visibly semantics. Concretely, we exploit determinizability as well as closure under complementation and intersection. In fact, the following definition of determinism only makes sense for the visibly semantics, but we will see later that a subclass of deterministic OCAs gives a natural notion of determinism for the observability semantics as well.

▶ **Definition 5** (deterministic OCA). An OCA $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ is called *deterministic* (dOCA or m-dOCA) if, for all $(q, k, \sigma) \in Q \times [m]_0 \times (\Sigma \cup \mathsf{Op})$, there is exactly one $q' \in Q$ such that $(q, k, \sigma, q') \in \Delta$. In that case, Δ represents a (total) function $\delta : Q \times [m]_0 \times (\Sigma \cup \mathsf{Op}) \to Q$ so that we rather consider $\mathcal{A}$ to be the tuple $(Q, \Sigma, \iota, F, m, \delta)$.

A powerset construction like for finite automata can be used to determinize OCAs wrt. the visibly semantics [5]. That construction also preserves the two other semantics. However, Definition 5 only guarantees uniqueness of runs for words from $(\Sigma \cup \mathsf{Op})^*$. That is, for complementation, we have to restrict to the visibly semantics (cf. Lemma 7 below).

▶ **Lemma 6** (cf. [5]). *Let $\mathcal{A}$ be an m-OCA. There is an m-dOCA $\mathcal{A}^{det}$ of exponential size such that $L_\eta(\mathcal{A}^{det}) = L_\eta(\mathcal{A})$ for all $\eta \in \{oca, vis, obs\}$.*

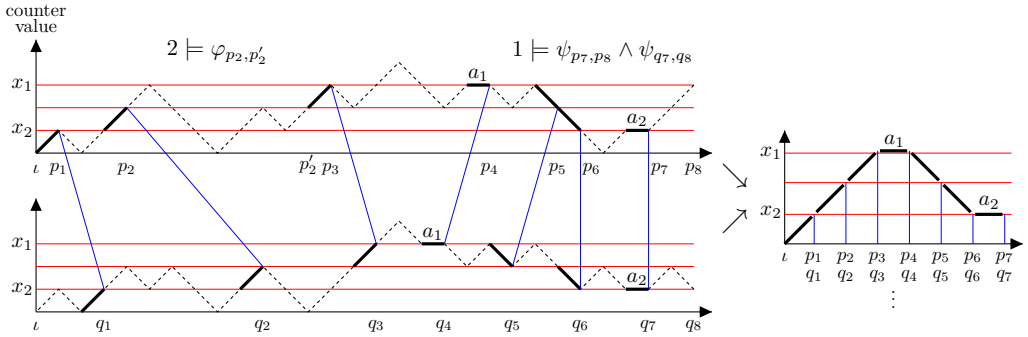

Figure 2 Decompositions of two runs on $(a_1, 3)(a_2, 1)$, and corresponding runs in normal form.

A (visibly) dOCA can be easily complemented wrt. the set of *well-formed words* $\mathsf{WF}_\Sigma := \{w \in (\Sigma \cup \mathsf{Op})^* \mid \text{no prefix of } w \text{ contains more } \downarrow\text{'s than } \uparrow\text{'s}\}$. In fact, for all OCAs $\mathcal{A}$ with alphabet Σ, we have $L_{vis}(\mathcal{A}) \subseteq \mathsf{WF}_\Sigma$.

▶ **Lemma 7.** *Let $\mathcal{A} = (Q, \Sigma, \iota, F, m, \delta)$ be a dOCA and define $\bar{\mathcal{A}}$ as the dOCA $(Q, \Sigma, \iota, Q \setminus F, m, \delta)$. Then, $L_{vis}(\bar{\mathcal{A}}) = \mathsf{WF}_\Sigma \setminus L_{vis}(\mathcal{A})$.*

Finally, visibility of counter operations allows us to simulate two OCAs in sync by a straightforward product construction:

▶ **Lemma 8** (cf. [3])**.** *Let $\mathcal{A}_1$ be an m_1-OCA and $\mathcal{A}_2$ be an m_2-OCA over the same alphabet. There is a $\max\{m_1, m_2\}$-OCA $\mathcal{A}_1 \times \mathcal{A}_2$ of polynomial size such that $L_{vis}(\mathcal{A}_1 \times \mathcal{A}_2) = L_{vis}(\mathcal{A}_1) \cap L_{vis}(\mathcal{A}_2)$. Moreover, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic, then so is $\mathcal{A}_1 \times \mathcal{A}_2$.*

3 Determinizing and Complementing OCAs

In this section, we will show that, under the *observability semantics*, OCAs are effectively closed under all boolean operations. The main ingredient of the proof is a determinization procedure, which we first describe informally.

Let $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ be the OCA to be determinized (wrt. the observability semantics). Moreover, let $w = (a_1, x_1) \dots (a_n, x_n) \in (\Sigma \times \mathbb{N})^*$. Every run ρ of $\mathcal{A}$ such that $obs(trace(\rho)) = w$ has to have reached the counter value x_1 by the time it reads the first letter a_1. In particular, it has to perform at least x_1 counter increments. In other words, we can identify x_1 transitions that lift the counter value from 0 to 1, from 1 to 2, and, finally, from $x_1 - 1$ to x_1, respectively, and that are separated by partial runs that "oscillate" around the current counter value but, at the end, return to their original level. Similarly, before reading the second letter a_2, $\mathcal{A}$ will perform $|x_2 - x_1|$-many *identical* counter operations to reach x_2, again separated by some oscillation phases, and so on. This is illustrated on the left hand side of Figure 2 for two runs on the word $(a_1, 3)(a_2, 1)$.

We will transform $\mathcal{A}$ into another automaton that decomposes a run into oscillations and increment/decrement/letter transitions, but, in fact, abstracts away oscillations. Thus, the automaton starts in an increasing mode and goes straight to the value x_1. Once it reads letter a_1, it may go into an increasing or decreasing mode, and so on. Observe that a run ρ of this new automaton is in a sort of normal form as illustrated on the right hand side of Figure 2. The crux is that $vis(trace(\rho))$ is a *unique* encoding of w: Of course, it determines the counter values output when a letter is read; and it is unique, since it continues incrementing (decrementing, respectively) until a letter is read. This normalization and encoding finally

allows us to apply known results on visibly one-counter automata for determinization and complementation.

There is a little issue here, since the possibility of performing an oscillation leading from p_2 to p_2' (cf. left hand side of Figure 2) depends on the current counter value. However, it was shown in [11] that the set of counter values allowing for such a shortcut can be described as a boolean combination of arithmetic progressions that can be computed in polynomial time. We will, therefore, work with an extended version of OCAs that includes arithmetic-progression tests (but is no more expressive, as we show afterwards).

The outline of this section is as follows: We present extended OCAs in Section 3.1 and the link between the observability and the visibly semantics in Section 3.2. In Section 3.3, we solve the universality and inclusion problem for OCAs wrt. the observability semantics.

3.1 Extended One-Counter Automata

While OCAs can only test a counter value up to some threshold, *extended OCAs* have access to boolean combinations of modulo constraints. The set $\mathsf{Guards}^{\mathrm{mod}}$ is given by the grammar $\varphi ::= c + d\mathbb{N} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$ where $c, d \in \mathbb{N}$. We call $c + d\mathbb{N}$ an *arithmetic-progression formula* and assume that c and d are encoded in unary. For $x \in \mathbb{N}$ (a counter value), we define $x \models c + d\mathbb{N}$ if $x = c + d \cdot i$ for some $i \in \mathbb{N}$. Thus, we may use *true* as an abbreviation for $0 + 1\mathbb{N}$. The other formulas are interpreted as expected. Moreover, given $\varphi \in \mathsf{Guards}^{\mathrm{mod}}$, we set $[\![\varphi]\!] := \{x \in \mathbb{N} \mid x \models \varphi\}$.

Before we introduce extended OCAs, we will state a lemma saying that the "possibility" of a shortcut in terms of an oscillation (see above) is definable in $\mathsf{Guards}^{\mathrm{mod}}$. Let $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ be an OCA and $p, q \in Q$. By $X^{\mathcal{A}}_{p,q}$, we denote the set of natural numbers $x \in \mathbb{N}$ such that $(p, x)\ (\overset{\uparrow}{\Longrightarrow}_{\mathcal{A}} \cup \overset{\downarrow}{\Longrightarrow}_{\mathcal{A}})^*\ (q, x)$, i.e., there is a partial run from (p, x) to (q, x) that performs only counter operations. Moreover, we define $Y^{\mathcal{A}}_{p,q}$ to be the set of natural numbers $x \in \mathbb{N}$ such that $(p, x)\ (\overset{\uparrow}{\Longrightarrow}_{\mathcal{A}} \cup \overset{\downarrow}{\Longrightarrow}_{\mathcal{A}})^*\ (q, x')$ for some $x' \in \mathbb{N}$. Note that $X^{\mathcal{A}}_{p,q} \subseteq Y^{\mathcal{A}}_{p,q}$. The following result is due to [11, Lemmas 6–9]:

▶ **Lemma 9** ([11]). *Let $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ be an OCA and $p, q \in Q$. We can compute, in polynomial time, guards $\varphi_{p,q}, \psi_{p,q} \in \mathsf{Guards}^{\mathrm{mod}}$ such that $[\![\varphi_{p,q}]\!] = X^{\mathcal{A}}_{p,q}$ and $[\![\psi_{p,q}]\!] = Y^{\mathcal{A}}_{p,q}$. In particular, the constants in $\varphi_{p,q}$ and $\psi_{p,q}$ are all polynomially bounded.*

▶ **Definition 10** (extended OCA). An *extended OCA (eOCA)* is a tuple $\mathcal{A} = (Q, \Sigma, \iota, f, \Delta)$ where Q, Σ, ι are like in an OCA, $f : Q \to \mathsf{Guards}^{\mathrm{mod}}$ is the *acceptance condition*, and Δ is the *transition relation*: a *finite* subset of $Q \times \mathsf{Guards}^{\mathrm{mod}} \times (\Sigma \cup \mathsf{Op}) \times Q$

Runs and the languages $L_\eta(\mathcal{A})$, with $\eta \in \{oca, vis, obs\}$, of an eOCA $\mathcal{A} = (Q, \Sigma, \iota, f, \Delta)$ are defined very similarly to OCAs. In fact, there are only two (slight) changes:

1. The definition of $\Longrightarrow_{\mathcal{A}} \subseteq \mathit{Conf}_{\mathcal{A}} \times ((\Sigma \times \mathbb{N}) \cup \mathsf{Op}) \times \mathit{Conf}_{\mathcal{A}}$ is now as follows: For $(q, x), (q', x') \in \mathit{Conf}_{\mathcal{A}}$ and $\tau \in (\Sigma \times \mathbb{N}) \cup \mathsf{Op}$, we have $(q, x) \overset{\tau}{\Longrightarrow}_{\mathcal{A}} (q', x')$ if there is $\varphi \in \mathsf{Guards}^{\mathrm{mod}}$ such that $x \models \varphi$ and one of the following holds:
   - $\tau = \uparrow$ and $x' = x + 1$ and $(q, \varphi, \uparrow, q') \in \Delta$, or
   - $\tau = \downarrow$ and $x' = x - 1$ and $(q, \varphi, \downarrow, q') \in \Delta$, or
   - $x' = x$ and there is $a \in \Sigma$ such that $\tau = (a, x)$ and $(q, \varphi, a, q') \in \Delta$.
2. A run is now *accepting* if its last configuration (q, x) is such that $x \models f(q)$.

Apart from these modifications, the languages $L_{oca}(\mathcal{A})$, $L_{vis}(\mathcal{A})$, and $L_{obs}(\mathcal{A})$ are defined in exactly the same way as for OCAs.

3.2 From OCAs with Counter Observability to Visibly OCAs

To establish a link between the observability and the visibly semantics, we will encode a word $w = (a_1, x_1)(a_2, x_2)\dots(a_n, x_n) \in (\Sigma \times \mathbb{N})^*$ as a word $\mathsf{enc}(w) \in (\Sigma \cup \mathsf{Op})^*$ as follows:

$$\mathsf{enc}(w) := \uparrow^{x_1} a_1 \, \mathrm{sign}(x_2 - x_1)^{|x_2 - x_1|} \, a_2 \, \dots \, \mathrm{sign}(x_n - x_{n-1})^{|x_n - x_{n-1}|} \, a_n$$

where, for an integer $z \in \mathbb{Z}$, we let $\mathrm{sign}(z) = \uparrow$ if $z \geq 0$, and $\mathrm{sign}(z) = \downarrow$ if $z < 0$. For example, $\mathsf{enc}(\varepsilon) = \varepsilon$ and $\mathsf{enc}((a,5)(b,2)(c,4)) = \uparrow^5 a \downarrow^3 b \uparrow^2 c$. The mapping enc is extended to sets $L \subseteq (\Sigma \times \mathbb{N})^*$ by $\mathsf{enc}(L) = \{\mathsf{enc}(w) \mid w \in L\}$. Let $\mathsf{Enc}_\Sigma := \mathsf{enc}((\Sigma \times \mathbb{N})^*)$ denote the set of *valid encodings*. Note that enc is a *bijection* between $(\Sigma \times \mathbb{N})^*$ and Enc_Σ, and that Enc_Σ is the set of *well-formed* words of the form $u_1 a_1 u_2 a_2 \dots u_n a_n$ where $a_i \in \Sigma$ and $u_i \in \{\uparrow\}^* \cup \{\downarrow\}^*$ for all $i \in \{1, \dots, n\}$.

Obviously, there is a small dOCA whose visibly semantics is Enc_Σ. It will be needed later for complementation of OCAs wrt. the observability semantics.

▶ **Lemma 11.** *There is a 0-dOCA $\mathcal{B}_{\mathsf{enc}}$ with only four states such that $L_{vis}(\mathcal{B}_{\mathsf{enc}}) = \mathsf{Enc}_\Sigma$.*

The idea is that $\mathcal{B}_{\mathsf{enc}}$ enters an "increasing" or "decreasing" mode as soon as it performs $\uparrow$ or, respectively, $\downarrow$. Such a mode can only be quit by reading a letter from Σ or entering a sink state. This avoids forbidden reversals between $\uparrow$ and $\downarrow$. Finally, it is easy to ensure that any nonempty accepted word ends in a letter from Σ.

In fact, there is a tight link between the visibly and the observability semantics of OCAs provided the visibly semantics contains only valid encodings:

▶ **Lemma 12.** *Let $\mathcal{A}$ be an OCA with alphabet Σ such that $L_{vis}(\mathcal{A}) \subseteq \mathsf{Enc}_\Sigma$. Then, we have $L_{vis}(\mathcal{A}) = \mathsf{enc}(L_{obs}(\mathcal{A}))$ and, equivalently, $L_{obs}(\mathcal{A}) = \mathsf{enc}^{-1}(L_{vis}(\mathcal{A}))$.*

Lemmas 8 and 12 imply the following closure property, which will later be exploited to solve the inclusion problem:

▶ **Proposition 13.** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be OCAs over Σ such that $L_{vis}(\mathcal{A}_1) \subseteq \mathsf{Enc}_\Sigma$ and $L_{vis}(\mathcal{A}_2) \subseteq \mathsf{Enc}_\Sigma$. Then, $L_{obs}(\mathcal{A}_1 \times \mathcal{A}_2) = L_{obs}(\mathcal{A}_1) \cap L_{obs}(\mathcal{A}_2)$ (where $\mathcal{A}_1 \times \mathcal{A}_2$ is due to Lemma 8).*

The next lemma constitutes the main ingredient of the determinization procedure. It will eventually allow us to rely on OCAs whose visibly semantics consists only of valid encodings.

▶ **Lemma 14.** *Let $\mathcal{A}$ be an OCA. We can compute, in polynomial time, an eOCA $\mathcal{A}^{ext}$ such that $L_{obs}(\mathcal{A}^{ext}) = L_{obs}(\mathcal{A})$ and, for all $w \in L_{obs}(\mathcal{A}^{ext})$, we have $\mathsf{enc}(w) \in L_{vis}(\mathcal{A}^{ext})$.*

Proof. Suppose $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ is the given OCA. We first translate a simple "threshold constraint" into an arithmetic expression that can be used as a guard in the eOCA $\mathcal{A}^{ext}$: Let $\pi_m = m + 1\mathbb{N}$, and $\pi_k = k + 0\mathbb{N}$ for all $k \in \{0, \dots, m-1\}$.

We define $\mathcal{A}^{ext} = (Q, \Sigma, \iota, f, \Delta')$ as follows: Essentially, $\mathcal{A}^{ext}$ simulates $\mathcal{A}$ so that it has the same state space. However, when $\mathcal{A}$ allows for a shortcut (oscillation) from state p to state q (which will be checked in terms of $\varphi_{p,q}$ from Lemma 9) and there is a transition (q, k, σ, q') of $\mathcal{A}$, then $\mathcal{A}^{ext}$ may perform σ and go directly from p to q', provided π_k is satisfied as well. Formally, the transition relation is given as

$$\Delta' = \{(p, \varphi_{p,q} \wedge \pi_k, \sigma, q') \mid p \in Q \text{ and } (q, k, \sigma, q') \in \Delta\}.$$

Moreover, a configuration (p, x) is "final" in $\mathcal{A}^{ext}$ if the current counter value x satisfies $\psi_{p,q}$ for some $q \in F$ (cf. Lemma 9). That is, for all $p \in Q$, we let $f(p) = \bigvee_{q \in F} \psi_{p,q}$. ◀

To transform an eOCA back into an ordinary OCA while determinizing it and preserving its observability semantics, we will need a dOCA that takes care of the modulo constraints:

▶ **Lemma 15.** *Let $\Omega \subseteq \mathsf{Guards}^{\mathrm{mod}}$ be a nonempty finite set. Set $m_\Omega := \max\{c \mid c + d\mathbb{N} \text{ is an atomic subformula of some } \varphi \in \Omega\} + 2$. There are a dOCA $\mathcal{B}_\Omega = (Q, \Sigma, \iota, Q, m_\Omega, \delta)$ of exponential size and $\lambda : Q \to 2^\Omega$ such that, for all $(q, x) \in Conf_{\mathcal{B}_\Omega}$ and all runs of $\mathcal{B}_\Omega$ ending in (q, x), we have $\lambda(q) = \{\varphi \in \Omega \mid x \models \varphi\}$.*

Proof. We sketch the idea. For every arithmetic-progression formula $c + d\mathbb{N}$ that occurs in Ω (for simplicity, let us assume $d \geq 1$), we introduce a state component $\{0, 1, \ldots, c\} \times \{0, 1, \ldots, d-1\}$. Increasing the counter, we increment the first component until c and then count modulo d in the second. We proceed similarly when decreasing the counter. The current state $(x, y) \in [c]_0 \times [d-1]_0$ will then tell us whether $c + d\mathbb{N}$ holds, namely iff $x = c$ and $y = 0$. Finally, the mapping λ evaluates a formula based on the outcome for its atomic subformulas. Note that $\mathcal{B}_\Omega$ can be computed in exponential time. Its size is exponential in the number of arithmetic-progression formulas that occur in Ω. ◀

We will now apply Lemma 15 to transform an eOCA into a dOCA (cf. also Lemma 6).

▶ **Lemma 16.** *Let $\mathcal{A}$ be an eOCA. We can compute, in exponential time, a dOCA $\mathcal{A}'$ (deterministic according to Definition 5) such that $L_\eta(\mathcal{A}') = L_\eta(\mathcal{A})$ for all $\eta \in \{oca, vis, obs\}$.*

Proof. Suppose $\mathcal{A} = (Q, \Sigma, \iota, f, \Delta)$ is the given eOCA. Let $\Omega \subseteq \mathsf{Guards}^{\mathrm{mod}}$ be the set of formulas that occur in Δ or f, and let $\mathcal{B}_\Omega = (\hat{Q}, \Sigma, \hat{\iota}, \hat{Q}, m_\Omega, \hat{\delta})$ be the dOCA along with the function λ according to Lemma 15.

We build the dOCA $\mathcal{A}' = (Q', \Sigma, \iota', F', m_\Omega, \delta')$ as follows. Essentially, we perform a simple powerset construction for $\mathcal{A}$. Moreover, to eliminate modulo guards, we run $\mathcal{B}_\Omega$ in parallel. Thus, the set of states is $Q' = 2^Q \times \hat{Q}$, with initial state $\iota' = (\{\iota\}, \hat{\iota})$ and set of final states $F' = \{(P, q) \in Q' \mid f(p) \in \lambda(q) \text{ for some } p \in P\}$. Finally, the transition function is given by $\delta'((P, q), k, \sigma) = (P', \hat{\delta}(q, k, \sigma))$ where $P' = \big\{p' \mid (p, \varphi, \sigma, p') \in \Delta \cap (P \times \lambda(q) \times \{\sigma\} \times Q)\big\}$. ◀

There is a "nondeterministic version" of Lemma 16, which does not perform a powerset construction but rather computes a nondeterministic OCA. The latter is then still of exponential size, but only wrt. to the number of arithmetic-progression formulas in $\mathcal{A}$.

With Theorem 3, it follows that nonemptiness for eOCAs can be solved in PSPACE. We do not know if this upper bound is tight.

Let $\mathcal{A}$ be a dOCA with alphabet Σ and let $w \in (\Sigma \times \mathbb{N})^*$. By $\rho_\mathcal{A}(w)$, we denote the unique run of $\mathcal{A}$ such that $vis(trace(\rho_\mathcal{A}(w))) = \mathsf{enc}(w)$.

By the following observation, which follows directly from Lemma 12, it is justified to call any dOCA $\mathcal{A}$ with $L_{vis}(\mathcal{A}) \subseteq \mathsf{Enc}_\Sigma$ *deterministic wrt. the observability semantics*:

▶ **Lemma 17.** *Let $\mathcal{A}$ be a dOCA such that $L_{vis}(\mathcal{A}) \subseteq \mathsf{Enc}_\Sigma$. For every word $w \in (\Sigma \times \mathbb{N})^*$, we have $w \in L_{obs}(\mathcal{A})$ iff $\rho_\mathcal{A}(w)$ is accepting.*

Altogether, we obtain that OCAs are determinizable wrt. the observability semantics.

▶ **Theorem 18** (determinizability)**.** *Let $\mathcal{A}$ be an OCA over Σ. We can compute, in exponential time, an m-dOCA $\mathcal{A}'$ (with m only polynomial) such that $L_{obs}(\mathcal{A}') = L_{obs}(\mathcal{A})$ and $L_{vis}(\mathcal{A}') \subseteq \mathsf{Enc}_\Sigma$.*

Proof. Let $\mathcal{A}$ be the given OCA. We apply Lemmas 14 and 16 to obtain a dOCA $\widetilde{\mathcal{A}}$ of exponential size such that $L_{obs}(\widetilde{\mathcal{A}}) = L_{obs}(\mathcal{A})$ and, for all $w \in L_{obs}(\widetilde{\mathcal{A}})$, we have $\mathsf{enc}(w) \in L_{vis}(\widetilde{\mathcal{A}})$. We set $\mathcal{A}' = \widetilde{\mathcal{A}} \times \mathcal{B}_{\mathsf{enc}}$ (cf. Lemmas 8 and 11) and obtain $L_{obs}(\mathcal{A}') = L_{obs}(\mathcal{A})$ and $L_{vis}(\mathcal{A}') \subseteq \mathsf{Enc}_\Sigma$. ◀

We conclude that OCAs are complementable wrt. the observability semantics:

▶ **Theorem 19** (complementability). *Let $\mathcal{A}$ be an OCA with alphabet Σ. We can compute, in exponential time, a dOCA $\bar{\mathcal{A}}$ such that $L_{obs}(\bar{\mathcal{A}}) = (\Sigma \times \mathbb{N})^* \setminus L_{obs}(\mathcal{A})$.*

Proof. We first transform $\mathcal{A}$ into the dOCA $\mathcal{A}' = \widetilde{\mathcal{A}} \times \mathcal{B}_{\mathsf{enc}}$ according to (the proof of) Theorem 18. Suppose $\widetilde{\mathcal{A}} = (Q, \Sigma, \iota, F, m, \delta)$. Then, we set $\bar{\mathcal{A}} = (Q, \Sigma, \iota, Q \setminus F, m, \delta) \times \mathcal{B}_{\mathsf{enc}}$. Note that $\bar{\mathcal{A}}$ is indeed a dOCA and that $L_{vis}(\bar{\mathcal{A}}) \subseteq \mathsf{Enc}_\Sigma$. For $w \in (\Sigma \times \mathbb{N})^*$, we have:

$$w \in L_{obs}(\bar{\mathcal{A}}) \overset{\text{Lem. } 17}{\iff} \rho_{\bar{\mathcal{A}}}(w) \text{ is accepting} \overset{(*)}{\iff} \rho_{\mathcal{A}'}(w) \text{ is not accepting} \overset{\text{Lem. } 17}{\iff} w \notin L_{obs}(\mathcal{A}')$$

Equivalence $(*)$ holds as $\rho_{\bar{\mathcal{A}}}(w)$ and $\rho_{\mathcal{A}'}(w)$ have the same projection to the Q-component. ◀

Determinization and complementation of *extended* OCAs are a priori more expensive: Lemmas 9 and 14 only apply to OCAs so that one has to go through Lemma 16 first.

3.3 Universality and Inclusion Problem wrt. Observability Semantics

We are now ready to solve the universality and the inclusion problem for OCAs wrt. the observability semantics. The *universality problem* is defined as follows: Given an OCA $\mathcal{A}$ over some alphabet Σ, do we have $L_{obs}(\mathcal{A}) = (\Sigma \times \mathbb{N})^*$? The *inclusion problem* asks whether, given OCAs $\mathcal{A}_1$ and $\mathcal{A}_2$, we have $L_{obs}(\mathcal{A}_1) \subseteq L_{obs}(\mathcal{A}_2)$.

▶ **Theorem 20.** *The universality problem and the inclusion problem for OCAs wrt. the observability semantics are PSPACE-complete. In both cases, PSPACE-hardness already holds when $|\Sigma| = 1$.*

Proof. To solve the universality problem for a given OCA $\mathcal{A} = (Q, \Sigma, \iota, F, m, \Delta)$ in (nondeterministic) polynomial space, we apply the construction from Theorem 19 (and, in particular, Theorem 18) on the fly to obtain a dOCA $\bar{\mathcal{A}}$ such that $L_{obs}(\bar{\mathcal{A}}) = (\Sigma \times \mathbb{N})^* \setminus L_{obs}(\mathcal{A})$. That is, we have to keep in memory a state of the form (P, q, r), where $P \subseteq Q$, q is the modulo-counting component (Lemma 15), and r is a state of $\mathcal{B}_{\mathsf{enc}}$ (Lemma 11). In addition, we will maintain a component for the current counter value. In fact, the latter can be supposed to be polynomially bounded (cf. [8] for a tight upper bound) in the size of $\bar{\mathcal{A}}$. The size of $\bar{\mathcal{A}}$ is exponential in the size of $\mathcal{A}$, and so the required information can be stored in polynomial space. To compute a successor state of (P, q, r), we first guess an operation $\sigma \in \Sigma \cup \mathsf{Op}$. We then compute (P', q') according to the proof of Lemma 16 and update r to r' according to the type of σ. Note that this takes polynomial time only, since the function λ as required in Lemma 15 can be computed on the fly. Finally, the algorithm outputs "non-universal" when we find a final state of $\bar{\mathcal{A}}$.

For the inclusion problem, we rely on Proposition 13 and perform the determinization procedure on-the-fly for *both* of the given OCAs.

For the lower bound, we will restrict to the universality problem, since it is a special case of the inclusion problem. We reduce from the universality problem for ordinary finite automata, which is known to be PSPACE-complete [17]. If we suppose that Σ is part of the input, then there is a straightforward reduction, which essentially takes the (ordinary) finite automaton and adds self-looping increment/decrement transitions to each state. Assuming $|\Sigma| = 1$, the reduction is as follows. Let $\mathcal{A}$ be a finite automaton over some finite alphabet $\Gamma = \{a_0, \ldots, a_{n-1}\}$. We construct an OCA $\mathcal{A}'$ over the singleton alphabet Σ such that $L(\mathcal{A}) = \Gamma^*$ iff $L(\mathcal{A}') = (\Sigma \times \mathbb{N})^*$. The idea is to represent letter a_i by (counter) value i. To obtain $\mathcal{A}'$, an a_i-transition in $\mathcal{A}$ is replaced with a gadget that nondeterministically outputs i or any other natural number strictly greater than $n - 1$. ◀

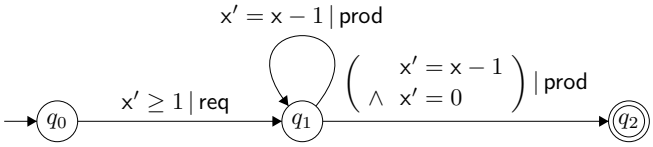

Figure 3 A strong automaton over $(\mathbb{N}, +1)$.

4 Relation with Strong Automata

In this section, we show that OCAs with counter observability are expressively equivalent to strong automata over $(\mathbb{N}, +1)$ [9]. As the latter are descriptive in spirit, OCAs can thus be seen as their operational counterpart.

Let us first give a short account of monadic second-order (MSO) logic over $(\mathbb{N}, +1)$ (see [23] for more details). We have infinite supplies of first-order variables, ranging over $\mathbb{N}$, and second-order variables, ranging over subsets of $\mathbb{N}$. The atomic formulas are *true*, $\mathsf{x}' = \mathsf{x} + 1$, $\mathsf{x}' = \mathsf{x}$, and $\mathsf{x} \in \mathsf{X}$ where x and x' are first-order variables and X is a second-order variable. Those formulas have the expected meaning. Further, MSO logic includes all boolean combinations, first-order quantification $\exists \mathsf{x} \Phi$, and second-order quantification $\exists \mathsf{X} \Phi$ (with Φ an MSO formula). The latter requires that there is a (possibly infinite) subset of $\mathbb{N}$ satisfying Φ. As abbreviations, we may also employ $\mathsf{x}' = \mathsf{x} - 1$ and formulas of the form $\mathsf{x}' \in (\mathsf{x} + c + d\mathbb{N})$, where $c, d \in \mathbb{N}$. This does not change the expressive power of MSO logic.

In the following, we assume that x and x' are two distinguished first-order variables. We write $\Phi(\mathsf{x}, \mathsf{x}')$ to indicate that the free variables of Φ are among x and x'. If $\Phi(\mathsf{x}, \mathsf{x}')$ is evaluated to true when x is interpreted as $x \in \mathbb{N}$ and x' is interpreted as $x' \in \mathbb{N}$, then we write $(x, x') \models \Phi$. In fact, a transition of a strong automaton is labeled with a formula $\Phi(\mathsf{x}, \mathsf{x}')$ and can only be executed if $(x, x') \models \Phi$ where x and x' are the natural numbers read at the previous and the current position, respectively. Thus, two successive natural numbers in a word can be related explicitly in terms of an MSO formula.

▶ **Definition 21** ([9]). A *strong automaton* is a tuple $\mathcal{S} = (Q, \Sigma, \iota, F, \Delta)$ where Q is the finite set of *states*, Σ is a nonempty finite alphabet, $\iota \in Q$ is the *initial state*, and $F \subseteq Q$ is the set of *final states*. Further, Δ is a *finite* set of *transitions*, which are of the form (q, Φ, a, q') where $q, q' \in Q$ are the source and the target state, $a \in \Sigma$, and $\Phi(\mathsf{x}, \mathsf{x}')$ is an MSO formula.

Similarly to an OCA, $\mathcal{S}$ induces a relation $\Longrightarrow_{\mathcal{S}} \subseteq \mathit{Conf}_{\mathcal{S}} \times (\Sigma \times \mathbb{N}) \times \mathit{Conf}_{\mathcal{S}}$, where $\mathit{Conf}_{\mathcal{S}} = Q \times \mathbb{N}$. For $(q, x), (q', x') \in \mathit{Conf}_{\mathcal{S}}$ and $(a, y) \in \Sigma \times \mathbb{N}$, we have $(q, x) \xRightarrow{(a,y)}_{\mathcal{S}} (q', x')$ if $y = x'$ and there is an MSO formula $\Phi(\mathsf{x}, \mathsf{x}')$ such that $(q, \Phi, a, q') \in \Delta$ and $(x, x') \models \Phi$. A *run* of $\mathcal{S}$ on $w = (a_1, x_1) \dots (a_n, x_n) \in (\Sigma \times \mathbb{N})^*$ is a sequence $\rho = (q_0, x_0) \xRightarrow{(a_1,x_1)}_{\mathcal{S}} (q_1, x_1) \xRightarrow{(a_2,x_2)}_{\mathcal{S}} \dots \xRightarrow{(a_n,x_n)}_{\mathcal{S}} (q_n, x_n)$ such that $q_0 = \iota$ and $x_0 = 0$. It is *accepting* if $q_n \in F$.

The language $L(\mathcal{S}) \subseteq (\Sigma \times \mathbb{N})^*$ of $\mathcal{S}$ is defined as the set of words $w \in (\Sigma \times \mathbb{N})^*$ such that there is an accepting run of $\mathcal{S}$ on w.

▶ **Example 22.** We refer to the OCA $\mathcal{A}$ from Example 2. Figure 3 depicts a strong automaton $\mathcal{S}$ such that $L(\mathcal{S}) = L_{obs}(\mathcal{A}) = \{(\mathsf{req}, n)(\mathsf{prod}, n-1)(\mathsf{prod}, n-2) \dots (\mathsf{prod}, 0) \mid n \geq 1\}$.

In fact, we can transform any OCA into an equivalent strong automaton preserving the observability semantics, and vice versa:

▶ **Theorem 23.** *Let $L \subseteq (\Sigma \times \mathbb{N})^*$. There is an OCA $\mathcal{A}$ such that $L_{obs}(\mathcal{A}) = L$ iff there is a strong automaton $\mathcal{S}$ such that $L(\mathcal{S}) = L$.*

Proof. "$\Longrightarrow$": Using the following observation, we can directly transform an OCA into a strong automaton: For all states q and q' of the given OCA $\mathcal{A}$, there is an MSO formula $\Phi_{q,q'}(\mathsf{x},\mathsf{x}')$ such that, for all $x, x' \in \mathbb{N}$, we have $(x,x') \models \Phi_{q,q'}$ iff $(q,x)\ \big(\overset{\uparrow}{\Longrightarrow}_{\mathcal{A}} \cup \overset{\downarrow}{\Longrightarrow}_{\mathcal{A}}\big)^*\ (q',x')$. The existence of $\Phi_{q,q'}$ can be shown using a two-way automaton over infinite words [21], which simulates $\mathcal{A}$ and can be translated into an MSO formula [21, 23].

More precisely, given q, q', we build a two-way automaton $\mathcal{T}_{q,q'}$ over the alphabet $2^{\{\$,\$'\}}$. The idea is that word positions represent counter values (the first position marking 0, the second 1, and so on), and $\$$ and $\$'$ represent x and x', respectively. Thus, we are only interested in words in which $\$$ and $\$'$ each occur exactly once. Clearly, this is a regular property. At the beginning, $\mathcal{T}_{q,q'}$ goes to the position carrying $\$$. It then simulates $\mathcal{A}$ starting in q, and it accepts if it is on the position carrying $\$'$ and in state q'. The simulation itself is straightforward: Counter increments and decrements of an OCA are simulated by going one step to the left or to the right, respectively, and a zero test simply checks whether the automaton is at the first position of the word. Note that $\mathcal{T}_{q,q'}$ checks for the markers $\$$ and $\$'$ only at the beginning and at the end of an execution, but not during the actual simulation of $\mathcal{A}$. Let $w = z_0 z_1 z_2 \ldots \in \big(2^{\{\$,\$'\}}\big)^\omega$ and $x, x' \in \mathbb{N}$ be unique positions such that $\$ \in z_x$ and $\$' \in z_{x'}$. Then, w is accepted by $\mathcal{T}_{q,q'}$ iff $(q,x)\ (\overset{\uparrow}{\Longrightarrow}_{\mathcal{A}} \cup \overset{\downarrow}{\Longrightarrow}_{\mathcal{A}})^*\ (q',x')$.

It is well known that two-way word automata are expressively equivalent to one-way automata (cf. [21]). Therefore, by Büchi's theorem, the word language accepted by $\mathcal{T}_{q,q'}$ can be translated into a corresponding MSO formula without free variables but with subformulas of the form "position y carries $\$$" and "position y carries $\$'$" [23]. We replace the latter two by $\mathsf{y} = \mathsf{x}$ and $\mathsf{y} = \mathsf{x}'$, respectively, and finally obtain $\Phi_{q,q'}$ as required.

"$\Longleftarrow$": We will transform a strong automaton $\mathcal{S}$ into an equivalent *OCA with super transitions*. A super transition can perform counter operations of the form $+\psi$ or $-\psi$ where $\psi \in \mathsf{Guards}^{\mathrm{mod}}$. Operation $+\psi$ allows the counter value to be increased by $n \in \mathbb{N}$ provided $n \models \psi$. Similarly, $-\psi$ allows the counter value to be decreased by n if $n \models \psi$.

▶ **Definition 24** (OCA with super transitions)**.** An *OCA with super transitions* is a tuple $\mathcal{A} = (Q, \Sigma, \iota, F, \Delta)$ where Q, Σ, ι, F are like in an OCA and Δ is the finite *transition relation*. A transition is of the form $(q, \varphi, op, a, \varphi', q')$ where $q, q' \in Q$ are the source and the target state, $\varphi, \varphi' \in \mathsf{Guards}^{\mathrm{mod}}$ are guards checking the original and the modified counter value, respectively, $a \in \Sigma$ is the output letter, and op is of the form $+\psi$ or $-\psi$ where $\psi \in \mathsf{Guards}^{\mathrm{mod}}$.

We define a global transition relation $\Longrightarrow_{\mathcal{A}} \subseteq \mathit{Conf}_{\mathcal{A}} \times (\Sigma \times \mathbb{N}) \times \mathit{Conf}_{\mathcal{A}}$ where, as usual, $\mathit{Conf}_{\mathcal{A}} = Q \times \mathbb{N}$. For $(q,x), (q',x') \in \mathit{Conf}_{\mathcal{A}}$ and $(a,y) \in \Sigma \times \mathbb{N}$, we have $(q,x) \overset{(a,y)}{\Longrightarrow}_{\mathcal{A}} (q',x')$ if $y = x'$ and there are $\varphi, \varphi', \psi \in \mathsf{Guards}^{\mathrm{mod}}$ such that $x \models \varphi$, $x' \models \varphi'$, and one of the following holds:

- $x \leq x'$ and $(q, \varphi, +\psi, a, \varphi', q') \in \Delta$ and $x' - x \models \psi$, or
- $x \geq x'$ and $(q, \varphi, -\psi, a, \varphi', q') \in \Delta$ and $x - x' \models \psi$.

With this, the language $L(\mathcal{A}) \subseteq (\Sigma \times \mathbb{N})^*$ is defined in the expected way, like for strong automata.

It is easily seen that $\mathcal{A}$ can be translated into an eOCA $\mathcal{A}'$ such that $L_{obs}(\mathcal{A}') = L(\mathcal{A})$: For every $\psi \in \mathsf{Guards}^{\mathrm{mod}}$, the set $\{\square^n \mid n \in [\![\psi]\!]\}$ is a *regular* language over the unary alphabet $\{\square\}$. Thus, counter operations of the form $+\psi$ or $-\psi$ can be simulated by a finite-state gadget. Essentially, we take a finite automaton for $\{\square^n \mid n \in [\![\psi]\!]\}$ and replace $\square$ by $\uparrow$ or, respectively, $\downarrow$. It will thus be enough to translate a strong automaton into an OCA with super transitions.

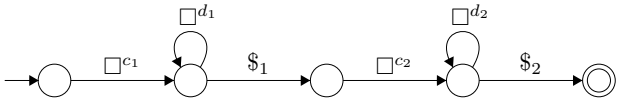

Figure 4 Finite automaton for $L^+_\Phi(c_1, d_1, c_2, d_2)$.

Next, we demonstrate why super transitions are indeed useful to emulate an MSO formula $\Phi(\mathsf{x}, \mathsf{x}')$. Using Büchi's theorem (cf. [23]), we can transform Φ into *finite automata* $\mathcal{B}^+_\Phi$ and $\mathcal{B}^-_\Phi$ recognizing the following regular languages over the alphabet $\{\square, \$_1, \$_2\}$:

- $L(\mathcal{B}^+_\Phi) = \{\square^x\$_1\square^y\$_2 \mid x, y \in \mathbb{N} \text{ such that } (x, x+y) \models \Phi\}$
- $L(\mathcal{B}^-_\Phi) = \{\square^x\$_2\square^y\$_1 \mid x, y \in \mathbb{N} \text{ such that } (x+y, x) \models \Phi\}$

Similarly to \$ and \$′ in the other proof direction, the positions of $\$_1$ and $\$_2$ in a word from $L(\mathcal{B}^+_\Phi) \cup L(\mathcal{B}^-_\Phi)$ encode an interpretation of the free variables x and, respectively, x' that makes Φ true. Note that $L(\mathcal{B}^+_\Phi)$ can be written as a finite union of sets

$$L^+_\Phi(c_1, d_1, c_2, d_2) := \{\square^x\$_1\square^y\$_2 \mid x \in [\![c_1 + d_1\mathbb{N}]\!] \text{ and } y \in [\![c_2 + d_2\mathbb{N}]\!]\}$$

with $c_1, d_1, c_2, d_2 \in \mathbb{N}$. This is achieved by determinizing $\mathcal{B}^+_\Phi$ and splitting it into components as illustrated in Figure 4 (cf. also [26] for a polynomial transformation). Similarly, $L(\mathcal{B}^-_\Phi)$ is the finite union of sets of the form

$$L^-_\Phi(c_1, d_1, c_2, d_2) := \{\square^x\$_2\square^y\$_1 \mid x \in [\![c_1 + d_1\mathbb{N}]\!] \text{ and } y \in [\![c_2 + d_2\mathbb{N}]\!]\}\,.$$

In other words, there are finite sets $D^+_\Phi, D^-_\Phi \subseteq \mathbb{N}^4$ such that

- $L(\mathcal{B}^+_\Phi) = \bigcup_{(c_1,d_1,c_2,d_2)\in D^+_\Phi} L^+_\Phi(c_1, d_1, c_2, d_2)$ and
- $L(\mathcal{B}^-_\Phi) = \bigcup_{(c_1,d_1,c_2,d_2)\in D^-_\Phi} L^-_\Phi(c_1, d_1, c_2, d_2)$.

We now turn to the actual translation of a strong automaton $\mathcal{S} = (Q, \Sigma, \iota, F, \Delta)$ into an OCA with super transitions $\mathcal{A} = (Q, \Sigma, \iota, F, \Delta')$ such that $L(\mathcal{A}) = L(\mathcal{S})$. Note that the only change is in the transition relation: For all $(q, \Phi, a, q') \in \Delta$ and $(c_1, d_1, c_2, d_2) \in D^+_\Phi$, Δ' contains $(q, c_1 + d_1\mathbb{N}, +(c_2 + d_2\mathbb{N}), a, \mathit{true}, q')$. Moreover, for all $(q, \Phi, a, q') \in \Delta$ and $(c_1, d_1, c_2, d_2) \in D^-_\Phi$, Δ' contains $(q, \mathit{true}, -(c_2 + d_2\mathbb{N}), a, c_1 + d_1\mathbb{N}, q')$. This concludes the construction of $\mathcal{A}$.

To prove $L(\mathcal{A}) = L(\mathcal{S})$, it is enough to show that, for all configurations $(q, x), (q', x') \in \mathit{Conf}_\mathcal{A}$ and all $a \in \Sigma$, the following are equivalent:

(1) $(q, x) \xRightarrow{(a,x')}_\mathcal{S} (q', x')$

(2) $(q, x) \xRightarrow{(a,x')}_\mathcal{A} (q', x')$

Suppose (1) holds. There is an MSO formula Φ such that $(q, \Phi, a, q') \in \Delta$ and $(x, x') \models \Phi$. We distinguish two (not necessarily disjoint) cases:

- Suppose $x \leq x'$. By $(x, x') \models \Phi$, we have $\square^x\$_1\square^{x'-x}\$_2 \in L(\mathcal{B}^+_\Phi)$. Thus, there is $(c_1, d_1, c_2, d_2) \in D^+_\Phi$ such that $\square^x\$_1\square^{x'-x}\$_2 \in L^+_\Phi(c_1, d_1, c_2, d_2)$. The latter implies $x \models c_1 + d_1\mathbb{N}$ and $x' - x \models c_2 + d_2\mathbb{N}$. Since we also have $(q, c_1 + d_1\mathbb{N}, +(c_2 + d_2\mathbb{N}), a, \mathit{true}, q') \in \Delta'$, (2) holds as well.
- Now, suppose $x \geq x'$. Then, by $(x, x') \models \Phi$, we have $\square^{x'}\$_2\square^{x-x'}\$_1 \in L(\mathcal{B}^-_\Phi)$. Thus, there is $(c_1, d_1, c_2, d_2) \in D^-_\Phi$ such that $\square^{x'}\$_2\square^{x-x'}\$_1 \in L^-_\Phi(c_1, d_1, c_2, d_2)$. This implies $x' \models c_1 + d_1\mathbb{N}$ and $x - x' \models c_2 + d_2\mathbb{N}$. Moreover, $(q, \mathit{true}, -(c_2 + d_2\mathbb{N}), a, c_1 + d_1\mathbb{N}, q') \in \Delta'$. We conclude that (2) holds.

Towards the other direction, suppose that (2) is true. Again, we will distinguish two (not necessarily disjoint) cases:

- Suppose $x \leq x'$ and suppose there is $(q, c_1 + d_1\mathbb{N}, +(c_2 + d_2\mathbb{N}), a, true, q') \in \Delta'$ such that $x \models c_1 + d_1\mathbb{N}$ and $x' - x \models c_2 + d_2\mathbb{N}$. There is an MSO formula Φ such that $(q, \Phi, a, q') \in \Delta$ and $(c_1, d_1, c_2, d_2) \in D^+_\Phi$. By $x \models c_1 + d_1\mathbb{N}$ and $x' - x \models c_2 + d_2\mathbb{N}$, we have $\Box^x\$_1\Box^{x'-x}\$_2 \in L^+_\Phi(c_1, d_1, c_2, d_2) \subseteq L(\mathcal{B}^+_\Phi)$. Thus, $(x, x') \models \Phi$. We deduce that (1) holds.
- Assume $x \geq x'$ and suppose there is a transition $(q, true, -(c_2+d_2\mathbb{N}), a, c_1+d_1\mathbb{N}, q') \in \Delta'$ such that $x - x' \models c_2 + d_2\mathbb{N}$ and $x' \models c_1 + d_1\mathbb{N}$. There is Φ such that $(q, \Phi, a, q') \in \Delta$ and $(c_1, d_1, c_2, d_2) \in D^-_\Phi$. Since $x - x' \models c_2 + d_2\mathbb{N}$ and $x' \models c_1 + d_1\mathbb{N}$, we have $\Box^{x'}\$_2\Box^{x-x'}\$_1 \in L^-_\Phi(c_1, d_1, c_2, d_2) \subseteq L(\mathcal{B}^-_\Phi)$. This implies $(x, x') \models \Phi$. Thus, (1) holds.

This concludes the correctness proof of $\mathcal{A}$. Finally, recall that one can easily transform $\mathcal{A}$ into an OCA whose observability semantics coincides with $L(\mathcal{A})$. ◀

5 Conclusion

The observability semantics opens several directions for follow-up work. We may carry it over to other classes of infinite-state systems such as Petri nets. Are there infinite-state restrictions of Petri nets other than 1-VASS whose observability semantics is robust?

A direct application of our results is that the language $L \subseteq (\Sigma \times \mathbb{N})^*$ of an OCA with observability semantics/strong automaton is learnable (in the sense of Angluin [4]) in terms of a visibly one-counter automaton for $\mathsf{enc}(L)$ [18]. It would be worthwhile to transfer results on visibly one-counter/pushdown automata that concern Myhill-Nerode congruences or minimization [2, 7].

Another interesting question is to which extent we can relax the requirement that the counter value be output with every letter $a \in \Sigma$. It may indeed be possible to deal with a bounded number of Σ-transitions between any two counter outputs. Note that there have been relaxations of the visibility condition in pushdown automata, albeit preserving closure under boolean operations [19].

Acknowledgments. The author is grateful to C. Aiswarya, Stefan Göller, Christoph Haase, and Arnaud Sangnier for numerous helpful discussions and pointers to the literature.

References

1 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2 R. Alur, V. Kumar, P. Madhusudan, and M. Viswanathan. Congruences for visibly pushdown languages. In *Proceedings of ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114, 2005.
3 R. Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM*, 56(3):1–43, 2009.
4 D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.
5 V. Bárány, C. Löding, and O. Serre. Regularity problems for visibly pushdown languages. In *Proceedings of STACS'06*, volume 3884 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2006.
6 A. Bès. An application of the Feferman-Vaught theorem to automata and logics for words over an infinite alphabet. *Logical Methods in Computer Science*, 4(1), 2008.
7 P. Chervet and I. Walukiewicz. Minimizing variants of visibly pushdown automata. In *Proceedings of MFCS'07*, volume 4708 of *Lecture Notes in Computer Science*, pages 135–146, 2007.

8 D. Chistikov, W. Czerwiński, P. Hofman, M. Pilipczuk, and M. Wehar. Shortest paths in one-counter systems. In *Proceedings of FoSSaCS'16*, volume 9634 of *Lecture Notes in Computer Science*, pages 462–478. Springer, 2016.

9 C. Czyba, C. Spinrath, and W. Thomas. Finite automata over infinite alphabets: Two models with transitions for local change. In *Proceedings of DLT'15*, volume 9168 of *Lecture Notes in Computer Science*, pages 203–214. Springer, 2015.

10 L. D'Antoni and M. Veanes. Minimization of symbolic automata. In *Proceedings of POPL'14*, pages 541–554. ACM, 2014.

11 S. Göller, R. Mayr, and A. Widjaja To. On the computational complexity of verifying one-counter processes. In *Proceedings of LICS'09*, pages 235–244. IEEE Computer Society Press, 2009.

12 S. A. Greibach. An infinite hierarchy of context-free languages. *Journal of the ACM*, 16(1):91–106, 1969.

13 C. Haase, J. Ouaknine, and J. Worrell. Relating reachability problems in timed and counter automata. *Fundamenta Informaticae*, 143(3-4):317–338, 2016.

14 O. H. Ibarra. Restricted one-counter machines with undecidable universe problems. *Mathematical Systems Theory*, 13:181–186, 1979.

15 P. Jančar. Decidability of bisimilarity for one-counter processes. *Information and Computation*, 158(1):1–17, 2000.

16 K. Mehlhorn. Pebbling mountain ranges and its application of DCFL-recognition. In *Proceedings of ICALP'80*, volume 85 of *Lecture Notes in Computer Science*, pages 422–435. Springer, 1980.

17 A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory*, pages 125–129, 1972.

18 D. Neider and C. Löding. Learning visibly one-counter automata in polynomial time. Technical Report AIB-2010-02, RWTH Aachen, January 2010.

19 D. Nowotka and J. Srba. Height-deterministic pushdown automata. In *Proceedings of MFCS'07*, volume 4708 of *Lecture Notes in Computer Science*, pages 125–134. Springer, 2007.

20 R. J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.

21 J.-P. Pécuchet. Automates boustrophédon et mots infinis. *Theoretical Computer Science*, 35:115–122, 1985.

22 J. Srba. Visibly pushdown automata: From language equivalence to simulation and bisimulation. In *Proceedings of CSL'06*, volume 4207 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2006.

23 W. Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.

24 L. G. Valiant and M. S. Paterson. Deterministic one-counter automata. *Journal of Computer and System Sciences*, 10(3):340–350, 1975.

25 J. van Leeuwen. Effective constructions in well-partially-ordered free monoids. *Discrete Mathematics*, 21(3):237–252, 1978.

26 A. Widjaja To. Unary finite automata vs. arithmetic progressions. *Inf. Process. Lett.*, 109(17):1010–1014, 2009.