Revisiting Synthesis for One-Counter Automata

Guillermo A. Pérez
University of Antwerp, Antwerp, Belgium
guillermoalberto.perez@uantwerpen.be

Ritam Raha
University of Antwerp, Antwerp, Belgium
ritam.raha@uantwerpen.be

Abstract

One-counter automata are obtained by extending classical finite-state automata with a counter whose value can range over non-negative integers and be tested for zero. The updates and tests applicable to the counter can further be made parametric by introducing a set of integer-valued variables. We revisit the parameter synthesis problem for such automata. That is, we ask whether there exists a valuation of the parameters such that all infinite runs of the automaton satisfy some omega-regular property. The problem has been shown to be encodable in a restricted one-alternation fragment of Presburger arithmetic with divisibility. In this work (i) we argue that said fragment of the logic is unfortunately undecidable. Nevertheless, by reduction to a class of partial observation games, (ii) we prove the synthesis problem is decidable. Finally, (iii) we give a polynomial-space algorithm for the problem if parameters can only be used in tests, and not updates, of the counter.

2012 ACM Subject Classification Theory of computation → Quantitative automata; Theory of computation → Logic and verification

Keywords and phrases Parametric one-counter automata, Synthesis, Reachability

Acknowledgements We thank Nathanaël Fijalkow for useful feedback, as well as Antonia Lechner and James Worrell for having brought the inconsistencies in [5, 19] to our attention.

1 Introduction

Counter automata are a classical model that extend finite-state automata with integer-valued registers. These have been shown to be useful in modelling complex systems, such as programs with lists and XML query evaluation algorithms [4, 7]. If the updates or tests applicable to the counter are *parameterized* by a set of integer-valued variables, one-counter automata can also model simple software programs and their parameters [16, 6]

Despite their usefulness as a modelling formalism, it is known that two counters suffice for counter automata to become Turing powerful [25]. In particular, this means that most interesting questions are undecidable. To circumvent this, several restrictions of the model have been studied in the literature, e.g. reversal-bounded counter automata [17] and one-counter automata. In this work we focus on an extension of the latter: one-counter automata with parametric updates and parametric (equality) tests. Reachability for such automata is known to be decidable [14, 21]. Model-checking algorithms for such automata also exist for some specification logics [11, 13].

In [19], Lechner studied a synthesis problem for one-counter automata with parameters which asks whether there exists some value of the parameters such that all runs of the automaton satisfy a specification given by a *linear temporal logic* (LTL) formula. By classical results, the problem is equivalent to asking whether all valuations make it so that all runs of the automaton satisfy some Büchi condition: they visit accepting states infinitely often (see, e.g. [2]). In that paper, Lechner proved that the Büchi condition can be decomposed into two reachability queries — intuitively a loop with an accepting state is reached and taken forever. She then goes on to reduce the resulting *synthesis reachability* problem to a fragment of Presburger arithmetic where a single alternation is allowed and a restricted divisibility predicate can be used. In [5], said fragment was claimed to be decidable. However, in [20] the result from [5] was stated as being under review. We clarify the latter by using developments from [5, 19] to prove the fragment is undecidable.

After invalidating the previous attempt at proving decidability of the synthesis reachability problem, we propose two alternatives. The first one based on the connection between counter automata and two-way automata; the second, on partial observation games. Specifically, assuming parameters are only present in counter tests, we provide small modifications to a recent construction due to Bollig et al. [3] in order to obtain alternating two-way automata whose language is empty if and only if our synthesis reachability problem has a negative answer. Regarding partial observation games, we prove that the complement of the problem reduces to energy games with partial observation [1] where the initial credit is fixed [9, 26]. The partial observability allows us to force Eve, the protagonist, to faithfully simulate runs of the counter automaton under all possible valuations.

2 Undecidability of a Restriction of AEPAD

Presburger arithmetic (PA) is the first-order theory of natural numbers in the structure $\langle \mathbb{N}, 0, 1, +, < \rangle$ where $+$ and $<$ are the standard addition and ordering of integers. *Presburger arithmetic with divisibility (PAD)* is the extension of PA obtained when we add the binary divisibility predicate $|$, where for all $a, b \in \mathbb{Z}$ we have $a \mid b \iff \exists c \in \mathbb{Z} : b = ac$. The resulting language in its full generality is undecidable [28]. In fact, already a single quantifier alternation allows to encode general multiplication, thus becoming undecidable [23]. However, the purely existential fragment of PAD, which we call *existential PAD (EPAD)*, has been shown to be decidable [22] and in **NEXP** [21].

Let X be a finite set of first-order variables. A *linear polynomial* over $\mathbf{x} = (x_1, \dots, x_n) \in X^n$ is given by the syntax rule: $p(\mathbf{x}) ::= \sum_{1 \leq i \leq n} a_i x_i + b$, where the a_i and b range over $\mathbb{Z}$ and the first-order variables from $\mathbf{x}$ range over $\mathbb{N}$. In general, quantifier-free PAD formulas have the following grammar: $\varphi ::= \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid f(\mathbf{x})\ P\ g(\mathbf{x})$, where P can be the order predicate $<$ or the divisibility predicate $|$, and f, g are linear polynomials. We define the standard Boolean abbreviations $\varphi_1 \vee \varphi_2 \iff \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. Moreover we introduce the abbreviations $f(x) \leq g(x) \iff f(x) < g(x) + 1$ and $f(x) = g(x) \iff f(x) \leq g(x) \wedge g(x) \leq f(x)$.

Allowing one restricted alternation. We define the language $\forall\exists_R$PAD of all PAD formulas allowing a universal quantification over some variables, followed by an existential quantification over variables which may not appear on the left-hand side of divisibility constraints. Formally, $\forall\exists_R$PAD is the set of all PAD formulas of the form

$$\forall z_1 \dots \forall z_n \exists x_1 \dots \exists x_m \varphi(\mathbf{x}, \mathbf{z}) \tag{1}$$

where every variable from φ — a quantifier-free PAD formula — is quantified, and all divisibility constraints are of the form $f(\mathbf{z}) \mid g(\mathbf{x}, \mathbf{z})$ with f and g being linear polynomials.

Positive-divisibility fragment. We denote by $\forall\exists_R\text{PAD}^+$ the subset of $\forall\exists_R$PAD formulas φ where the negation operator can only be applied to the order predicate $<$ and the only other Boolean operators allowed are conjunction and disjunction. In other words $\forall\exists_R\text{PAD}^+$ is a restricted negation normal form in which divisibility predicates cannot be negated. Lechner showed in [19] that all $\forall\exists_R$PAD formulas can be translated into a formula in $\forall\exists_R\text{PAD}^+$.

▶ **Proposition 1** (Lechner's trick). *For all φ_1 in $\forall\exists_R PAD$ one can compute φ_2 in $\forall\exists_R PAD^+$ such that φ_1 is true if and only if φ_2 is true.*

Proof. Consider a formula φ_1 in $\forall\exists_R$PAD of the form from Equation (1). We observe φ_1 can always be brought into negation normal form so that negations are applied only to predicates [27]. Hence, it suffices to argue that we can do away with negated divisibility predicates while staying within $\forall\exists_R$PAD.

The claim thus follows from the identity we state below since the newly introduced variables x', x'' are both existentially quantified and only appear on the right-hand side of divisibility constraints. For all $a, b \in \mathbb{Z}$ we have the following.

$$\begin{aligned}\neg(a \mid b) \iff\ & (a = 0 \wedge b \neq 0) \vee \exists x' \exists x'' \left((b = x' + x'') \wedge (a \mid x') \wedge (0 < x'' < b)\right) \\ & \vee \exists x' \exists x'' \left((b = -x' - x'') \wedge (a \mid x') \wedge (0 < x'' < -b)\right)\end{aligned}$$

In words, we have that if $a = 0$ and $b \neq 0$ then $\neg(a \mid b)$. Furthermore, if $a \neq 0$ then there exist unique integers $q, r \in \mathbb{Z}$ such that $b = qa + r$ and $0 < r < |b|$ if and only if $\neg(a \mid b)$. ◀

Undecidability. We will now prove that the language $\forall\exists_R$PAD$^+$ is undecidable, that is, to determine whether a given formula from $\forall\exists_R$PAD$^+$ is true is an undecidable problem.

▶ **Theorem 2.** *The language $\forall\exists_R PAD^+$ is undecidable.*

From Proposition 1 it follows that arguing $\forall\exists_R$PAD is undecidable suffices to prove the theorem. In [5], the authors show that $\exists\forall_R$PAD is undecidable. The set $\exists\forall_R$PAD contains all formulas of the form $\neg\varphi$ such that $\varphi \in \forall\exists_R$PAD. Their argument consists in defining the least common multiple predicate, the squaring predicate, and subsequently integer multiplication.

▶ **Proposition 3** (From [5]). *The language $\exists\forall_R PAD$ is undecidable.*

3 One-Counter Automata Models

A *one-counter automaton* is a finite automaton with a single counter that gets updated or tested along every transition of the automaton. In this work we focus on a *parametric* version of such automata. Formally, a *parametric one-counter automaton* is a tuple $\mathcal{A} = (Q, T, \delta, q_{in}, X)$, where Q is the finite set of states, $q_{in} \in Q$ is the start state, X is a finite set of parameters, $T \subseteq Q \times Q$ is a finite set of transitions and $\delta : T \to Op$ is a function that associates every transition to an operation from the set Op. The set $Op = CU \uplus PU \uplus ZT \uplus PT$ is a union of:

- *Constant Updates* or $CU := \{+a : a \in \mathbb{Z}\}$
- *Parametric Updates* or $PU := \{Sx : S \in \{+1, -1\}, x \in X\}$
- *Zero Tests* or $ZT := \{= 0\}$
- *Parametric Tests* or $PT := \{= x, \geq x : x \in X\}$

We denote by "$= 0$" or "$= x$" an *equality test* between the counter and zero or the value of a parameter respectively; by "$\geq x$" a *lower-bound test* between the counter and the value of x.

A *valuation* of the parameter set X is a function $V : X \to \mathbb{N}$ that assigns every parameter a non-zero natural number. We often denote $\delta(q, q') = op$ by (q, op, q') or $q \xrightarrow{op} q'$.

3.1 Paths, runs, and reachability

A *path* starting from a given state q is a sequence $\pi = (q_0, op_1, q_1)(q_1, op_2, q_2)\ldots$ where $(q_i, op_{i+1}, q_{i+1}) \in \delta$ for all i and $q_0 = q$. A *configuration* is a pair (q, c) where $q \in Q$ and $c \in \mathbb{N}$ is the *counter value*. Note that, by definition, $c \geq 0$ always holds.

Given a valuation $V : X \to \mathbb{N}$ and a configuration (q_0, c_0), a *V-run from* (q_0, c_0) is a sequence $\rho = (q_0, c_0)(q_1, c_1)\ldots$ such that:

1. The sequence of transitions $\tau = (q_0, \delta(q_0, q_1), q_1)(q_1, \delta(q_1, q_2), q_2)\ldots$ is a path. We say that ρ *induces* the path τ and that τ *lifts* to a V-run (that is, ρ) from (q_0, c_0).
2. For all $i \geq 0$ we have that $c_i = 0$, $c_i = V(x)$, or $c_i \geq V(x)$, if $\delta(q_i, q_{i+1})$ is "$= 0$", "$= x$", "$\geq x$", respectively.
3. Lastly, the following holds for all $i \geq 0$: $c_{i+1} = c_i$ if $\delta(q_i, q_{i+1}) \in (ZT \cup PT)$; $c_{i+1} = c_i + a$ if $\delta(q_i, q_{i+1}) = +a$; and $c_{i+1} = c_i + SV(x)$ if $\delta(q_i, q_{i+1}) = Sx$.

We say that ρ *reaches* a state $q_f \in Q$ if there exists j, such that $q_j = q_f$. When V is clear from the context, e.g. when $X = \emptyset$, we omit V and just write run instead of V-run.

3.2 Model taxonomy

Historically, counter machines have been defined with constant updates $\{-1, 0, +1\}$. In the literature, models obtained when general binary-encoded integer updates are allowed are sometimes referred to as *succinct* counter machines (see, for instance, [14, 12, 15, 3]). We focus on the following one-counter automata models.

- Succinct parametric one-counter automata (SOCAP, for short) allow for constant and parametric updates as well as zero and parametric tests.
- Succinct one-counter automata (SOCA) only allow for constant updates and zero tests, i.e. $X = \emptyset$ and therefore $PT = PU = \emptyset$.
- One-counter automata with parametric tests (OCAPT) allow for constant updates of the form $\{+a : a \in \{-1, 0, 1\}\}$ as well as zero and parametric tests. However, $PU = \emptyset$.

	CU	PU	ZT	PT
	$\{+a : a \in \mathbb{Z}\}$	$\{Sx : S \in \{+1, -1\}, x \in X\}$	$\{= 0\}$	$\{= x, \geq x : x \in X\}$
SOCAP	✓	✓	✓	✓
OCAPT	$\{-1, 0, 1\}$	✘	✓	✓
SOCA	✓	✘	✓	✘

Table 1 Summary of operations allowed by SOCAP, OCAPT, and SOCA with parameter set X

3.3 Decision problems

A fundamental decision problem we can ask in non-parametric one-counter automata is *the (state-)reachability problem*. Given a model $\mathcal{A}$ and a state q_f, the problem asks whether there exists a run of $\mathcal{A}$ that starts from a given configuration and reaches q_f. Alternatively, one may want all runs to reach q_f. If one focuses on infinite runs, this is defined as follows.

▶ **Definition 4** (UnivReach). The universal reachability problem *asks, given a model $\mathcal{A}$, a configuration χ, and a state q_f, whether all infinite runs from χ reach q_f.*

For parametric one-counter automata one may be interested in whether there exist values for the parameters such that the reachability problem has a positive answer.

▶ **Definition 5** (SynthReach). The reachability synthesis problem *asks, given a model $\mathcal{A}$, a configuration χ, and a state q_f, whether there exists a valuation such that all infinite runs from χ reach q_f.*

4 One-Counter Automata with Parametric Tests

The following complexity upper bound for the reachability synthesis problem for OCAPT is our main result in this section. We obtain it by following an idea from [3] to encode parameter valuations of OCAPT into words accepted by an alternating two-way automata.

▶ **Theorem 6.** *The* SYNTHREACH *problem for OCAPT is in* $\mathbf{NP}^{\mathbf{coNP}} = \mathbf{NP}^{\mathbf{NP}}$.

Proof. In Lemma 15 we will prove that if there is a valuation V of the parameters such that all infinite V-runs reach q_f then we can assume that V assigns to each $x \in X$ a value at most exponential. Hence, we can guess them and store them in binary using a polynomial number of bits. Once we have guessed V and replaced all the x_i by $V(x_i)$, we obtain a SOCA $\mathcal{A}'$ and the problem we ask is now UNIVREACH for $\mathcal{A}'$. We will see in Proposition 16 that, for SOCA, this problem is in **coNP**. It follows that SYNTHREACH for OCAPT is in $\mathbf{NP}^{\mathbf{coNP}} = \mathbf{NP}^{\mathbf{NP}}$. ◀

4.1 Alternating two-way automata

Given a finite set Y, we denote by $\mathbb{B}^+(Y)$ the set of positive Boolean formulas over Y, including *true* and *false*. A subset $Y' \subseteq Y$ satisfies $\beta \in \mathbb{B}^+(Y)$, written $Y \models \beta$, if β is evaluated to *true* when substituting *true* for every element in Y', and *false* for every element in $Y \setminus Y'$. In particular, we have $\emptyset \models$ *true*.

We can now define an *alternating two-way automata* (A2A, for short) as a tuple $\mathcal{T} = (S, \Sigma, s_{in}, \Delta, S_f)$, where S is a finite set of states, Σ is a finite alphabet, $s_{in} \in S$ is the initial state, $S_f \subseteq S$ is the set of accepting states, and $\Delta \subseteq S \times (\Sigma \cup \{\mathit{first?}\}) \times \mathbb{B}^+(S \times \{+1, 0, -1\})$ is the finite transition relation.

In an A2A, $+1$ intuitively means that the head moves to the right; -1, that the head moves to the left; 0, that it stays at the current position. Furthermore, transitions are labelled by Boolean formulas over successors which determine whether the current run branches off in a non-deterministic or a universal fashion, or a combination thereof.

Run trees. A run (tree!) γ of $\mathcal{T}$ on an infinite word $w = a_0 a_1 a_2 \cdots \in \Sigma^w$ from $n \in \mathbb{N}$ is a (possibly infinite) rooted tree whose vertices are labelled with elements in $S \times \mathbb{N}$ and such that it satisfies the following properties. The root of γ is labelled by (s_{in}, n). Moreover, for every vertex labelled by (s, m) with $k \in \mathbb{N}$ children labelled by $(s_1, n_1), \ldots, (s_k, n_k)$, there is a transition $(s, \sigma, \beta) \in \Delta$ such that,

- the set $\{(s_1, n_1 - m), \ldots, (s_k, n_k - m)\} \subseteq S \times \{+1, 0, -1\}$ satisfies β,
- $\sigma = \mathit{first?}$ implies $m = 0$, and
- $\sigma \in \Sigma$ implies $a_m = \sigma$.

In particular, all vertices (s, m) must have children unless there is a transition $(s, \cdot, \mathit{true}) \in \Delta$. We write $(s, m) \rightsquigarrow (s', m')$ to denote a branch of a run from a vertex labelled with (s, m) to one labelled with (s', m').

A run is accepting if all of its infinite branches contain infinitely many vertices with labels from $S_f \times \mathbb{N}$. This condition is vacuously fulfilled by finite trees.

Language and emptiness problem. The language of $\mathcal{T}$ is defined as $L(\mathcal{T}) := \{w \in \Sigma^w \mid$ there exists an accepting run of $\mathcal{T}$ on w from $0\}$. The *non-emptiness problem for A2As* asks to determine, given an A2A $\mathcal{T}$ and $n \in \mathbb{N}$, whether $L(\mathcal{T}) \neq \emptyset$. It is known that the problem can be decided using polynomial space in the size of the A2A.

▶ **Proposition 7** (From [29]). *The non-emptiness problem for A2As is in* **PSPACE***.*

In what follows from a given OCAPT $\mathcal{A}$ we will build an A2A $\mathcal{T}$ such that $\mathcal{T}$ accepts precisely those words which correspond to a valuation V of X under which all infinite runs reach q_f. Hence, checking SynthReach of $\mathcal{A}$ reduces to checking non-emptiness of $\mathcal{T}$.

4.2 From parametric one-counter to alternating two-way automata

Following [3], we encode a valuation $V : X \to \mathbb{N}$ as an infinite word $w = a_0a_1a_2\dots$ over the alphabet $\Sigma = X \cup \{\Box\}$ such that $a_0 = \Box$ and, for every $x \in X$, there is exactly one position $i \in \mathbb{N}$ such that $a_i = x$. We refer to such words w as *parameter words* and write $w(i)$ to denote its prefix $a_0a_1\dots a_i$ up to the letter a_i. By $|w(i)|_\Box$, we denote the number of occurrences of $\Box$ in $a_1\dots a_i$. (Note that we ignore a_0.) Then, a parameter word w determines a valuation $V_w : x \mapsto |w(i)|_\Box$ where $a_i = x$.

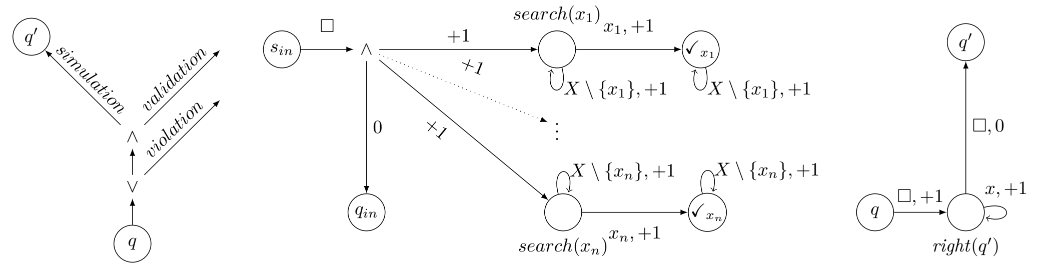
▶ **Example 8.** Note that $\Box x_1 \Box x_2 \Box^\omega$ is a parameter word, whereas $x_1x_2\Box^\omega$ and $\Box x_1 x_2 x_1 \Box^\omega$ are not. The word $w = \Box\Box\Box x_1 x_2 \Box^\omega$ determines the valuation $V_w = \{x_1 = 2, x_2 = 2\}$.

Observe that for every valuation V there is at least one parameter word w such that $V_w = V$. We denote the set of all parameter words over X by W_X.

From a given OCAPT $\mathcal{A} = (Q, T, \delta, q_{in}, X)$ and a state q_f, we will now construct an A2A $\mathcal{T} = (S, \Sigma, s_{in}, \Delta, S_f)$ that accepts the set of parameter words w such that, under the valuation $V = V_w$, all infinite runs reach q_f.

▶ **Proposition 9.** *There is a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for all OCAPT $\mathcal{A}$ there is an A2A $\mathcal{T}$ with $|\mathcal{T}| = p(|\mathcal{A}|)$ and $L(\mathcal{T}) = \{w \in W_X \mid$ all infinite V_w-runs of $\mathcal{A}$ reach $q_f\}$.*

The construction is based on the A2A built in [3], although we make more extensive use of the alternating semantics of the automaton. The main idea is to encode runs of $\mathcal{A}$ as branches of run trees of $\mathcal{T}$ on parameter words w by letting sub-trees t whose root is labelled with (q, i) correspond to the configuration $(q, |w(i)|_\Box)$ of $\mathcal{A}$. Every such t will serve as a witness that all runs of $\mathcal{A}$ from $(q, |w(i)|_\Box)$ reach q_f.



(a) Sub-A2A structure **(b)** $\mathcal{T}^{inp}_{sub}$ checking if the input is a valid parameter word **(c)** $\mathcal{T}^{inc}_{sub}$ encoding an increment

Figure 1 Sub-A2As for the word-validity check and to simulate increments; we use $search(x)$, $\checkmark_x$, and $right(q)$ as state names to make their function explicit

We sketch the construction below and elaborate on the sub-A2As in the sequel.

- The constructed A2A $\mathcal{T}$ for the given $\mathcal{A}$ is such that $Q \subseteq S$.
- The A2A includes a sub-A2A that verifies that the input word is a valid parameter word.

- Recall that from a run sub-tree whose root is labelled with (q, i), the A2A verifies all runs of $\mathcal{A}$ from $(q, |w(i)|_\Box)$ reach q_f. To do this, for all transitions $\delta = (q, op, q')$, we create a sub-A2A $\mathcal{T}^{\delta}_{sub}$ using copies of sub-A2As we describe in the sequel. For each such δ, one of two cases should hold: either the transition cannot be simulated (because of a zero test or a decrement from zero), or the transition can indeed be simulated. For the former, we add a *violation branch* to check that is indeed the case; for the latter, a *validation branch* checks the transition can be simulated and a *simulation branch* reaches $(q', \cdot)$.
- We obtain the global A2A $\mathcal{T}$ by connecting sub-A2As as follows. To ensure that all the runs of $\mathcal{A}$ are simulated, we have the global transition relation Δ be a conjunction of that of the sub-A2As which start at the same state $q \in Q$. For instance, let $\delta_1 = (q, op_1, q_1), \delta_2 = (q, op_2, q_2)$ be transitions of $\mathcal{A}$. The constructed sub-A2As $\mathcal{T}^{\delta_1}_{sub}, \mathcal{T}^{\delta_2}_{sub}$ will contain transitions $(q, \Box, \beta_1) \in \Delta_1, (q, \Box, \beta_2) \in \Delta_2$ respectively. In $\mathcal{T}$, we instead have $(q, \Box, \beta_1 \wedge \beta_2) \in \Delta$.
- Finally, we add transitions $(q_f, \sigma, \mathit{true}) \in \Delta$ for all $\sigma \in \Sigma$ to allow accepting runs of $\mathcal{T}$ to stop simulating runs of $\mathcal{A}$ that have already reached q_f.

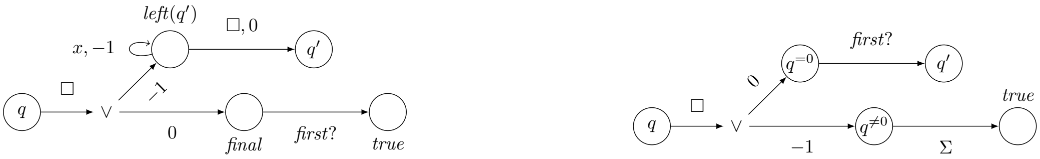
Verifying the input word. The sub-A2A $\mathcal{T}^{inp}_{sub}$ depicted in Figure 1b checks whether the given input is a valid parameter word. We let S_f consist of states $\checkmark_{x_i}$, one per $x_i \in X$.

▶ **Lemma 10.** *It holds that $L(\mathcal{T}^{inp}_{sub}) = W_X$.*

Proof. The A2A $\mathcal{T}^{inp}_{sub}$ consists of one deterministic one-way automata, per $x \in X$, whose language clearly corresponds to the set of words where x occurs exactly once. In $\mathcal{T}^{inp}_{sub}$, from the initial state and on the first letter $\Box$, a transition with a conjunction formula leads to all sub-automata for each x. The result follows. ◀

Increments. For every transition $\delta = (q, +1, q')$ we construct $\mathcal{T}^{inc}_{sub}$ (see Figure 1c). A run of this sub-A2A starts from q and some position c on the input word. Recall that c uniquely determines the current counter value in the simulated run of $\mathcal{A}$ (although, it should be noted c itself is not the counter value). Then, the run of $\mathcal{T}^{inc}_{sub}$ moves to the next occurrence of $\Box$ to the right of the current position and then goes to q'.

Decrements. For transitions $\delta = (q, -1, q')$ we construct $\mathcal{T}^{dec}_{sub}$ (see Figure 2a). In contrast to the increment sub-A2A, it also includes a *violation* branch in case the decrement would result in a negative counter value: On this branch, $\mathcal{T}^{dec}_{sub}$ attempts to read *first*? to determine if the position of the head corresponds to the first letter of the word.



(a) $\mathcal{T}^{dec}_{sub}$ encoding an decrement

(b) $\mathcal{T}^{zero}_{sub}$ encoding a zero test

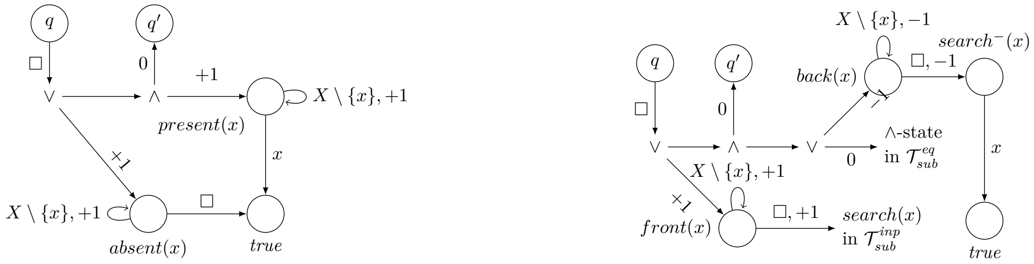
Figure 2 Sub-A2As to simulate decrements and zero tests

▶ **Lemma 11.** *Let $i, j \in \mathbb{N}$ and $w \in W_X$ with $\Box$ the $(i+1)$-th letter of w. A run tree γ of $\mathcal{T}^{dec}_{sub}$ on w from i is accepting if and only if either $(q, i) \rightsquigarrow (q', j)$ is a part of γ and $|w(i)|_\Box - 1 = |w(j)|_\Box$, or $(q, 0) \rightsquigarrow (\mathit{final}, 0)$ is a part of γ and $i = 0$.*

Zero tests. For every transition $\delta = (q, =0, q')$ we construct $\mathcal{T}_{sub}^{zero}$ (see Figure 2b) similarly to how we did for decrements. For the *validation* branch, it reads *first*? to confirm the position of the head is at the beginning of the word. For the *violation* branch, it moves the head to the left to confirm that the head is not at the beginning.

▶ **Lemma 12.** *Let $i \in \mathbb{N}$ and $w \in W_X$ with $\square$ the $(i+1)$-th letter of w. A run tree γ of $\mathcal{T}_{sub}^{zero}$ on w from i is accepting if and only if either $(q, 0) \rightsquigarrow (q', 0)$ is a part of γ and $i = 0$, or $(q, i) \rightsquigarrow (q^{\neq 0}, i-1)$ is a part of γ and $|w(i)|_\square > 0$.*

Parametric equality tests. For every transition $\delta = (q, = x, q')$ we construct $\mathcal{T}_{sub}^{eq}$ (see Figure 3a). For the *validation* branch, it moves the head right, skipping over other variable symbols $X \setminus \{x\}$, while looking for x. For the *violation* branch it skips over other variable symbols while looking for the next $\square$.



(a) $\mathcal{T}_{sub}^{eq}$ for parametric equality tests

(b) $\mathcal{T}_{sub}^{lb}$ for parametric lower-bound tests

Figure 3 Sub-A2As to simulate parametric tests

▶ **Lemma 13.** *Let $i \in \mathbb{N}$ and $w \in W_X$ with $\square$ the $(i+1)$-th letter of w. A run tree γ of $\mathcal{T}_{sub}^{eq}$ on w from i is accepting if and only if either $(q, i) \rightsquigarrow (q', i)$ is part of γ and $V_w(x) = |w(i)|_\square$, or $(q, i) \rightsquigarrow (absent(x), i+1)$ is a part of γ and $V_w(x) \neq |w(i)|_\square$.*

Parametric lower-bound tests. For every transition $\delta = (q, \geq x, q')$ we construct $\mathcal{T}_{sub}^{lb}$ (see Figure 3b). For the *validation* branch, we check for equality to x or we check whether $> x$. We also create the corresponding *violation* branches.

▶ **Lemma 14.** *Let $i \in \mathbb{N}$ and $w \in W_X$ with $\square$ the $(i+1)$-th letter of w. A run tree γ of $\mathcal{T}_{sub}^{lb}$ on w from i is accepting if and only if either $(q, i) \rightsquigarrow (q', i)$ is part of γ and $|w(i)|_\square \geq V_w(x)$, or $(q, i) \rightsquigarrow (front(x), i+1)$ is a part of γ and $|w(i)|_\square < V_w(x)$.*

Using the previous lemmas, it is straightforward to prove Proposition 9.

4.3 An upper bound for reachability synthesis

Again following developments from [3], we will now sketch a guess-and-check procedure using the fact that Proposition 9 implies a sufficient bound on valuations satisfying SynthReach.

▶ **Lemma 15.** *There exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that if there is a valuation V such that all infinite V-runs of $\mathcal{A}$ reach q_f then there is a valuation V' such that $V'(x) \in \mathcal{O}(2^{p(|\mathcal{A}|)})$ for all $x \in X$ and all infinite V'-runs of $\mathcal{A}$ reach q_f.*

Proof. Using Proposition 9 for OCAPT, there is an A2A $\mathcal{T}$ such that, $L(\mathcal{T})$ is precisely the subset of W_X such that all infinite V_w-runs of $\mathcal{A}$ reach q_f. Additionally, we have that every

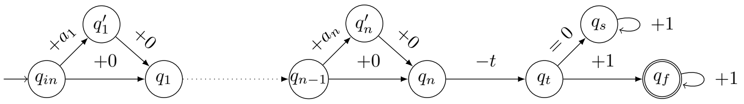


Figure 4 Reduction from the complement of the SubsetSum problem to UnivReach of SOCA

sub-A2A is linear in the size of $|\mathcal{A}|$, hence $|\mathcal{T}|$ is polynomial w.r.t. $|\mathcal{A}|$. We then use that there is a non-deterministic Büchi automaton $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{T})$ and $|\mathcal{B}| \in 2^{\mathcal{O}(|\mathcal{T}|^2)}$ [30, 8].

Suppose $\mathcal{A}$ is a positive instance of the SynthReach problem, i.e. $L(\mathcal{B}) \neq \emptyset$. We know the language of Büchi automata is non-empty only if there is "lasso" word which witnesses this. For all parameter words w accepted by a lasso there is a word $u \in \Sigma^*$ such that $|u| \leq |\mathcal{B}|$ and $w = u\square^\omega \in L(\mathcal{B})$. The result thus follows from our encoding of valuations. ◀

It remains to give an algorithm for verifying that the resulting SOCA, after substituting parameters with their values, is such that all infinite runs reach q_f.

▶ **Proposition 16.** *The* UnivReach *problem for SOCA is* **coNP***-complete.*

Before proving the claim above, we first establish an auxiliary lemma.

A *path* $\pi = (q_0, op_1, q_1) \dots (q_{n-1}, op_n, q_n)$ in $\mathcal{A}$ is a *cycle* if $q_0 = q_n$. We say the cycle is *simple* if no state (besides q_0) is repeated, i.e., for all $0 \leq i < j < n$ we have $q_i \neq q_j$. A cycle *starts from a zero test* if op_1 is "$= 0$". A *zero-test-free cycle* is a cycle where no op in it is a zero test. We define a *pumpable cycle* as being a zero-test-free cycle such that for all runs $\rho = (q_0, c_0) \dots (q_n, c_n)$ lifted from π we have $c_n \geq c_0$, i.e., the effect of the cycle is non-negative. The following result was argued for in [19, Section 3.4].

▶ **Lemma 17.** *Let $\mathcal{A}$ be a SOCA with an infinite run which does not reach q_f. Then there exists an infinite run of $\mathcal{A}$ which does not reach q_f and such that it induces a path of the form $\pi_0 \cdot \pi_1^\omega$, where π_1 either starts from a zero test or it is a simple pumpable cycle.*

Now, we can move to the proof of Proposition 16.

Proof of Proposition 16. The UnivReach problem for SOCA asks whether all infinite runs starting from $(q_{in}, 0)$ reach q_f or not. Lemma 17 shows two conditions, one of which must hold if there exists an infinite run that does not reach q_f. Note that both conditions are in fact reachability properties: to a cycle that starts from a zero test or to a simple pumpable cycle. Since the reachability problem for SOCA is in **NP** [14], we can guess which condition will hold and guess the polynomial-time verifiable certificates. This implies that the UnivReach problem for SOCA is in **coNP**.

For the lower bound, we give a reduction from the complement of the SubsetSum problem, which is known to be **NP**-complete [10]. The idea is similar to reductions used in the literature to prove **NP**-hardness for reachability in SOCA. Given an instance of the SubsetSum problem with a set $\{a_1, \dots, a_n\} \subseteq \mathbb{N}$ and a target sum t, we create a SOCA $\mathcal{A}$ as depicted in Figure 4. Note that, the UnivReach instance is positive if and only if the answer to the SubsetSum problem is negative (a full proof is given in appendix). Hence, the UnivReach problem for SOCA is **coNP**-hard. ◀

5 Succinct and Parametric One-Counter Automata

In this section we establish the main result of this paper:

▶ **Theorem 18.** *The* SynthReach *problem for SOCAP is decidable.*

We actually focus on the following problem: is there a run that reaches $(q_f, 0)$ from $(q_{in}, 0)$ for all valuations? At the end of the section, we comment on how the construction can be modified to prove the same about SynthReach.

For convenience, we assume $PT = \emptyset$. This is no loss of generality since parametric tests can be implemented using parametric updates and zero tests.

5.1 Partial-observation energy games

A *Partial-Observation Energy Game* (or POEG) $\mathcal{G}$ is a tuple $(Q, q_0, \Sigma, \Delta, w, \mathrm{Obs})$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is a finite alphabet of actions, $\Delta \subseteq Q \times \Sigma \times Q$ is a total transition relation, $w : \Delta \to \mathbb{Z}$ is a weight function, and $\mathrm{Obs} \subseteq \mathcal{P}(Q)$ is a partition of Q into observations. We say $\mathcal{G}$ is blind if $\mathrm{Obs} = \{Q\}$ and write $w_{max} := \max\{|w(\delta)| : \delta \in \Delta\}$.

Plays & observation sequences. A *play* in $\mathcal{G}$ is an infinite path $\pi = (q_0, \sigma_1, q_1)(q_1, \sigma_2, q_2) \ldots$ We say π visits a state q if there exists $i \geq 0$ such that $q_i = q$. The unique observation containing state q is denoted by $\mathrm{obs}(q)$. We extend $\mathrm{obs}(\cdot)$ to plays and prefixes in the natural way. For instance, $\mathrm{obs}(\pi)$ denotes $(\mathrm{obs}(q_0), \sigma_1, \mathrm{obs}(q_1))(\mathrm{obs}(q_1), \sigma_1, \mathrm{obs}(q_2)) \ldots$

The *energy level* of a play prefix $\rho = \ldots (q_{n-1}, \sigma_n, q_n)$ is $\mathrm{EL}(\rho) := \sum_{i=0}^{n-1} w(q_i, \sigma_{i+1}, q_{i+1})$. The set of plays satisfying the *(fixed-initial-credit) energy objective* is defined as follows: $\mathrm{PosEn} := \{\pi \mid \forall i \geq 0 : \mathrm{EL}(\pi[0..i]) \geq 0\}$, where $\pi[0..i]$ is the prefix of π up to i-th transition. In words, the *energy objective* asks for the energy level of a play never to drop below 0.

Strategies and winning. An *(observation-based) strategy for Eve* is a function λ from play prefixes to actions such that for all play prefixes ρ, ρ' we have $\mathrm{obs}(\rho) = \mathrm{obs}(\rho') \implies \lambda(\rho) = \lambda(\rho')$. A path $\pi = (q_0, \sigma_1, q_1) \ldots$ is consistent with λ if $\lambda(\pi[0..i]) = \sigma_{i+1}$ for all $i \geq 0$. We say a strategy λ for Eve is a *winning strategy* if all plays consistent with λ are in PosEn.

The adversary, Adam, intuitively plays as follows. Given a play prefix and an action $\sigma \in \Sigma$, he selects a σ-successor q of the current state and reveals $\mathrm{obs}(q)$ to Eve.

▶ **Proposition 19** (From [26]). *Given a POEG, determining whether there exists a winning strategy for Eve is decidable in Ackermannian time.*

5.2 From parametric counter automata to energy games

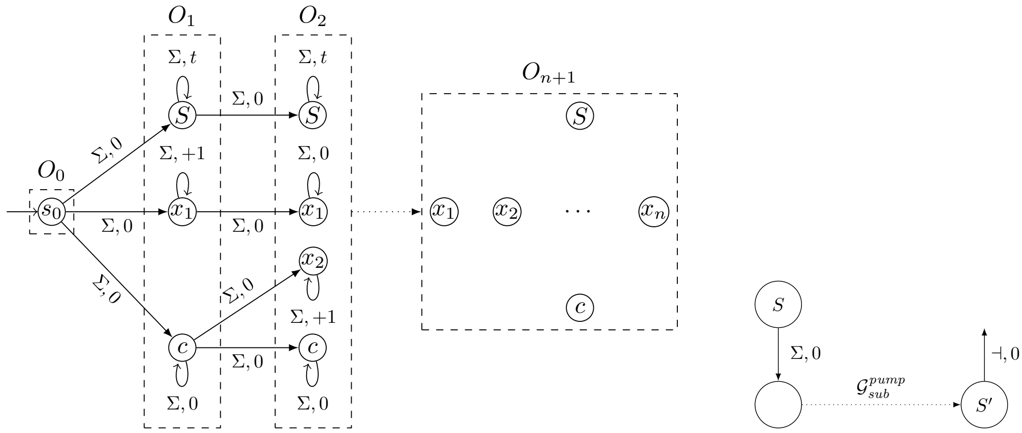
Fix $\mathcal{A} = (Q, T, \delta, q_{in}, X)$ where $X = \{x_1, x_2, \ldots x_n\}$. We call a V-run of $\mathcal{A}$ from $(q_{in}, 0)$ *accepting* if it reaches $(q_f, 0)$. We construct a POEG $\mathcal{G} = (S, s_0, \Sigma, \Delta, w, \mathrm{Obs})$ such that:

▶ **Lemma 20.** *For all valuations V there exists an accepting V-run of $\mathcal{A}$ if and only if Eve has a winning strategy in $\mathcal{G}$.*

The main idea is to allow Adam to choose a valuation V of the parameters. These are encoded as the energy levels of plays that are observation-equivalent to Eve. From there, the partial observability of the game allows us to set up sub-games which force Eve to faithfully simulate all updates and tests, parametric or not, of the run (cf. [9, 26]).

- Our construction consists of two phases and $n + 4$ observations. During the initial phase, Adam chooses a parameter valuation and Eve sets up "possible plays" with high energy levels. In the second phase, Eve perpetually simulates accepting runs of $\mathcal{A}$.
- We set the alphabet to be $\Sigma := \Sigma_T \uplus \{\dashv, \$\} \uplus \Sigma_{pump} \uplus \Sigma_{PU}$ where $\Sigma_T := \{\#\} \cup \{(q, op, q') : (q, q') \in T, \delta(q, q') = op\}$, and Σ_{pump} and Σ_{PU} contain actions used in sub-games.

- The sub-game $\mathcal{G}_{sub}^{val}$ allows Adam to choose a valuation of the parameters. The sub-game $\mathcal{G}_{sub}^{pump}$ creates a bound on the length of the accepting run of $\mathcal{A}$. Finally, $\mathcal{G}$ also contains a family of sub-games $\mathcal{G}_{sub}^{PU}$ that force Eve to faithfully simulate parametric updates.



(a) $\mathcal{G}_{sub}^{val}$ allowing Adam to choose the valuations

(b) $\mathcal{G}_{sub}^{pump}$ for run-length bound

Figure 5 Initial sub-games used to choose a valuation and bound on the length of a run

Allowing Adam to choose valuations. The sub-game $\mathcal{G}_{sub}^{val}$ depicted in Figure 5a allows Adam to choose a valuation V of the parameters. It consists of $n+2$ observations denoted by dashed boxes. The game starts from $O_0 = \{s_0\}$ and, on any action from Σ, the play can move to any state in O_1 with weight 0. Observations O_i, for all $1 \leq i \leq n$, contain $i+2$ states. All states have self-loops on all Σ as well as transitions to their corresponding copies in the next observation. The $i+2$ states are:

- c with a zero-loop to maintain a play with energy level of 0;
- x_i with a $+1$ loop for Adam to fix the value of $x_i \in X$ to $V(x_i)$ by staying in O_i;
- $i-1$ states with zero-loops to preserve the values of $x_1, \ldots, x_{i-1}$;
- S with a $+t$ self-loop to obtain a value of $t \cdot \sum_{i=1}^{n} V(x_i)$, where $t := |Q| \cdot a_{\max}$ and $a_{\max} := \max\{|a| : +a \in CU\}$.

For notational convenience, we use the same name for the states in the different observations.

The last observation O_{n+1} contains n states that keep track of the valuation of the n parameters, a c state and an S state. From all states in O_{n+1} there is an outgoing transition on Σ and with weight 0 leading to the next sub-game.

▶ **Lemma 21.** *For all x_i and all $\sigma_1 \ldots \sigma_k \in \Sigma^k$, with $k \geq n+1$, there is a unique play prefix ρ of the form $(s_0, \sigma_1, s_1) \ldots (s_{k-1}, \sigma_k, x_i)$ such that $x_i \in O_{n+1}$. Furthermore, any play π staying in $\mathcal{G}_{sub}^{val}$ forever is such that $\pi \in \mathrm{PosEn}$.*

Note that for any such play prefix ρ_i, Eve can infer the value $V(\rho_i) := \mathrm{EL}(\rho_i)$ based on how many times she observed O_i along the way. However, due to partial observability, she cannot distinguish between ρ and other play prefixes ρ' of the form $(s_0, \sigma_1, s'_1) \ldots (s'_{k-1}, \sigma_k, s'_k)$ with $s'_k \in O_{n+1}$. We write $V_\forall$ to denote the valuation $x_i \mapsto V(\rho_i)$ which, from the above arguments, Eve can infer for any play prefix reaching O_{n+1}. We extend $V_\forall$ to S and c.

Upper-bounding the length of the accepting run. A key element of our reduction to POEGs is that we give Eve a finite number of transitions she can use to simulate an accepting run of $\mathcal{A}$. It will be useful to prove a sufficient bound on the length of such runs. Below, let $x^V_{\max} := \max\{V(x) \mid x \in X\}$. We omit V and write $x_{\max}$ if it is clear from the context.

▶ **Lemma 22.** *Let V be a parameter valuation. If there exists an accepting V-run of $\mathcal{A}$ then then there exists an accepting V-run ρ of $\mathcal{A}$ whose length is at most $\mathcal{O}(|Q|^4 \cdot a^3_{\max} \cdot x^3_{max})$.*

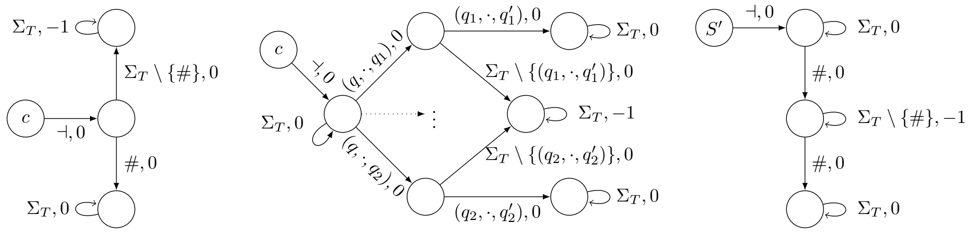
The lemma follows from its well-known analogue for (non-parametric) SOCA [18].

To allow Eve to simulate a run of bounded length according to Lemma 22 we will later use the sub-game from Figure 6c. However, we still need to reach that sub-game with a run whose energy level is at least the upper bound. After the initial sub-game in which Adam chooses parameter valuations, we already have $V_\forall(S) = |Q| \cdot a_{\max} \cdot (\sum_{i=1}^n x_i) \geq |Q| \cdot a_{\max} \cdot x_{\max}$. Conveniently, it is known that a POEG can be constructed which allows Eve to build a run that exponentiates its initial energy level [26].

We will make use of the family of *the fast-growing functions* $(F_i)_{i \geq 0}$. These are number-theoretic functions defined inductively as $F_{i+1}(y) := F_i^{y+1}(y)$, that is the $(y+1)$-composition of F_i, and $F_0(y) := y + 1$. Presently, we will only be needing F_2, which can be shown to be exponential. Hence, $F_2(V_\forall(S))$ suffices for our bound.

▶ **Proposition 23** (From [26]). *For all $i, y \in \mathbb{N}$ we can construct a blind POEG $\mathcal{G}'$ where Eve has no winning strategy with initial energy level y but there is a state s_f and a strategy λ for her such that all play prefixes π consistent with λ and ending at s_f have $\mathrm{EL}(\rho) = F_i(y)$.*

Let $\mathcal{G}^{pump}_{sub}$ denote the POEG for F_2 and Σ_{pump} its set of actions. Eve enters this sub-game from $S \in O_{n+1}$ after playing any action. All other states in O_{n+1} go to states in which there is a 0-weight self-loop on Σ_{pump}. We add transitions from S' on $\dashv \in \Sigma_{pump}$; and from all other states in $\mathcal{G}^{pump}_{sub}$, to a state with a 0-weight self-loop on Σ, i.e. Eve wins immediately. Since Eve does not have a winning strategy in $\mathcal{G}^{pump}_{sub}$, we assume she plays to reach S' with $F_2(V_\forall(S))$, see Figure 5b. Note that if the play does not reach S', she wins.



(a) $\mathcal{G}^{start}_{sub}$ ensuring first letter is #

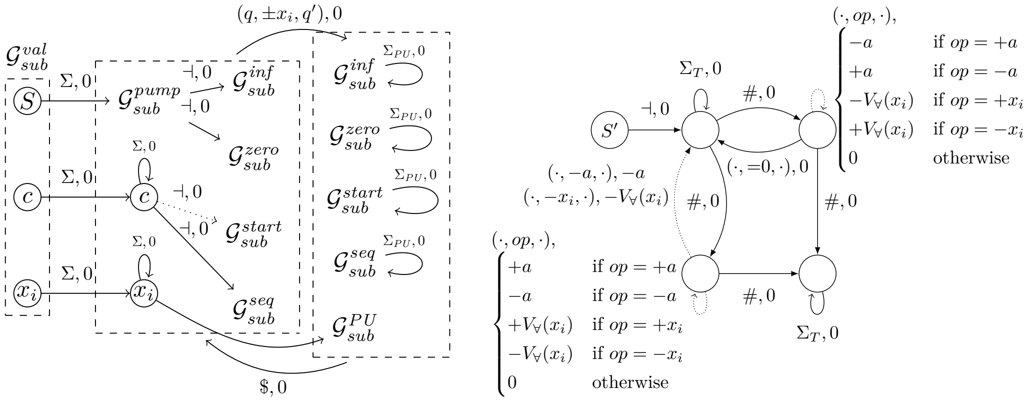
(b) $\mathcal{G}^{seq}_{sub}$ ensuring the sequence of transitions is correct starting from q

(c) $\mathcal{G}^{inf}_{sub}$ ensuring # is played often

Figure 6 Sub-games used to enforce conditions for the correct simulation of accepting runs

After ⊣, Eve must play #. After playing ⊣, the play can move to one of several sub-games in the same observation, see Figure 7a. Our goal is to have Eve perpetually simulate accepting runs separated by #. Her playing blindly in the sub-games means: (i) we can think of her strategy as a word from Σ^ω, (ii) she must win in all sub-games with the same word.

The sub-game $\mathcal{G}^{start}_{sub}$ depicted in Figure 6a forces Eve to start by playing # after ⊣. If Eve does not play #, the play can go to the upper state. There the energy level eventually drops below zero and she loses. Otherwise, it reaches the lower state and Eve wins in $\mathcal{G}^{start}_{sub}$.



$\mathcal{G}^{val}_{sub}$
S
c
x_i
$\Sigma, 0$
$\Sigma, 0$
$\Sigma, 0$
$\mathcal{G}^{pump}_{sub}$
$\dashv, 0$
$\dashv, 0$
$\mathcal{G}^{inf}_{sub}$
$\mathcal{G}^{zero}_{sub}$
$\mathcal{G}^{start}_{sub}$
$\mathcal{G}^{seq}_{sub}$
$(q, \pm x_i, q'), 0$
$\Sigma_{PU}, 0$
$\mathcal{G}^{PU}_{sub}$
$\$, 0$
S'
$\Sigma_T, 0$
$\#, 0$
$(\cdot, =0, \cdot), 0$
$(\cdot, -a, \cdot), -a$
$(\cdot, -x_i, \cdot), -V_\forall(x_i)$
$(\cdot, op, \cdot),$
$-a$ if $op = +a$
$+a$ if $op = -a$
$-V_\forall(x_i)$ if $op = +x_i$
$+V_\forall(x_i)$ if $op = -x_i$
0 otherwise
$+a$ if $op = +a$
$-a$ if $op = -a$
$+V_\forall(x_i)$ if $op = +x_i$
$-V_\forall(x_i)$ if $op = -x_i$
0 otherwise

(a) $\mathcal{G}$ after initial phase **(b)** $\mathcal{G}^{zero}_{sub}$ verifying counter updates and tests

Figure 7 Global game structure and update-test sub-game

Parametric updates. We now describe the most intricate sub-games: The family $\mathcal{G}^{PU}_{sub} = \{\mathcal{G}^{+x}_{sub}, \mathcal{G}^{-x}_{sub} : x \in X\}$ simulates transitions with parametric updates in a separate observation which contains copies of all sub-games (see Fig. 7a). We do so by "calling a procedure" implemented by a *deterministic* counter automaton: one transition is active from every state.

▶ **Proposition 24** (Adapted from [26]). *For all non-parametric deterministic k-counter automata M realizing a function $f : \mathbb{N}^k \to \mathbb{N}^k$, there is a blind POEG $\mathcal{G}_M$ such that M has a run $(q_{in}, \mathbf{c_0}) \dots (q_f, \mathbf{c_m})$ with $f(\mathbf{c_0}) = \mathbf{c_m}$ if and only if Eve has a winning strategy $\mathcal{G}_M$.*

We can construct a 3-counter automaton which receives $V_\forall(x)$ and the current counter value v as input and uses a temporary counter to increase (or decrease) the value of the latter by the former. Furthermore, this can be done while keeping the counter values below $v + V_\forall(x)$ and thus, in our construction, below $F_2(V_\forall(S))$. (See appendix for such an automaton.) For every $x \in X$, we construct $\mathcal{G}^{\pm x}_{sub}$ from the corresponding 3-counter automaton. We return from $\mathcal{G}^{\pm x}_{sub}$ with the affected counter value, or reset the sub-simulation, on \$ (Adam chooses which, to keep Eve from cheating). Henceforth, we assume Eve plays optimally in $\mathcal{G}^{PU}_{sub}$. Thus, for correctness, we may assume the play returns to the global simulation.

Eve plays # often. The sub-game $\mathcal{G}^{inf}_{sub}$ depicted in Figure 6c ensures Eve plays # within $F_2(V_\forall(S))$ actions from Σ_T of each other. For a word w over Σ, write $w|_T$ for the maximal subword over $\Sigma_T \setminus \{\#\}$ of w. Formally, the sub-game ensures Eve plays as follows.

▶ **Lemma 25.** *A strategy of Eve is winning in $\mathcal{G}^{inf}_{sub}$ if and only if it is a word $\#w_0\#w_1\#\dots$ such that for all $\rho_i = w_i|_T$ we have $|\rho_i| \leq F_2(V_\forall(S))$.*

The sequence of transitions is respected. Copies of the sub-game $\mathcal{G}^{seq}_{sub}$, one per state $q \in Q$, ensure that Eve plays a valid run of $\mathcal{A}$ (see Figure 6b). That is, she respects the sequence of transitions. Importantly, after having played a transition that ends at q_f, i.e. $(\cdot, \cdot, q_f)$, the sub-game allows Eve to play # and reach q_{in} again: to reset the simulation. If Eve plays a transition (q, op, q') that is not valid from q, a play can go to the sink where its energy level will become negative; otherwise Eve wins in $\mathcal{G}^{seq}_{sub}$.

▶ **Lemma 26.** *A strategy of Eve is winning in $\mathcal{G}^{seq}_{sub}$ if and only if it is a word $\#w_0\#w_1\#\dots$ where all $w_i|_T = (q_0, op_1, q_1)\dots$ are such that $(q_j, q_{j+1}) \in T$ for all $j \geq 0$ and all ρ_i start at q_{in} and end at q_f for $i \geq 1$.*

Zero tests. The sub-game $\mathcal{G}_{sub}^{zero}$ (see Figure 7b) ensures Eve (eventually) simulates all zero tests properly and the counter value does not go negative: Either (i) Eve executes a zero test when the counter value is positive (*a zero cheat*); or (ii) Eve executes a $-a$ or $-x_i$, and the counter value becomes negative (*a positive cheat*).

Note the sub-game has an incoming transition from the state S', so the initial energy level is $F_2(V_\forall(S))$ — and we will keep that as an invariant. If Eve commits a zero cheat, there is a play that goes to the top-right state, simulates the inverse operations and comes back to the initial state on the zero cheat. If she commits a positive cheat, there is a play that goes to the bottom-left state, simulates the operations and comes back to the initial state on the positive cheat. Both plays have an energy level of at most $F_2(V_\forall(S)) - 1$. Hence Eve can cheat at most $F_2(V_\forall(S))$ times.

▶ **Lemma 27.** *A strategy of Eve is winning in $\mathcal{G}_{sub}^{zero}$ if and only if it is a word $\#w_0\#w_1\#\ldots$ where all but finitely many $\rho_i = w_i|_T$ are accepting runs.*

Using the previous lemmas it is easy to show that Eve can win in $\mathcal{G}$ if and only if there exists an accepting run of $\mathcal{A}$ for all valuations. The full proof is given in appendix.

Sketch of proof for Theorem 18. For Theorem 18 we again let Adam choose valuations and then have Eve simulate the finite path $\pi_0\pi_1$ implied by Lemma 17. More formally, Eve simulates a run in $\mathcal{A}^2$ and chooses when to end π_0 and start π_1. We then need to add sub-games to verify π_1 starts at a zero test or is pumpable. ◀

6 Conclusion

We clarified the decidability status of synthesis problems for one-counter automata with parametric tests and updates. Our result using alternating two-way automata is perhaps more interesting for applications. We have unfortunately been unable to reduce SynthReach for full SOCAP to A2As. However, we conjecture the POEG we construct for them can actually be realized via multi-energy games with "resets and transfers". The latter may yield a primitive-recursive upper bound, unlike our approach. Lower bounds for SynthReach for OCAPT and SOCAP would be interesting as future work.

References

1 Krzysztof R. Apt and Erich Grädel. *Lectures in game theory for computer scientists.* Cambridge University Press, 2011.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

3 Benedikt Bollig, Karin Quaas, and Arnaud Sangnier. The complexity of flat freeze LTL. *Logical Methods in Computer Science*, 15(3), 2019. URL: https://doi.org/10.23638/LMCS-15(3:33)2019, doi:10.23638/LMCS-15(3:33)2019.

4 Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomás Vojnar. Programs with lists are counter automata. In *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, pages 517–531, 2006. URL: https://doi.org/10.1007/11817963_47, doi:10.1007/11817963_47.

5 Marius Bozga and Radu Iosif. On decidability within the arithmetic of addition and divisibility. In Vladimiro Sassone, editor, *Foundations of Software Science and Computational Structures, 8th International Conference, FOSSACS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3441 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 2005. URL: https://doi.org/10.1007/978-3-540-31982-5_27, doi:10.1007/978-3-540-31982-5_27.

6 Daniel Bundala and Joël Ouaknine. On parametric timed automata and one-counter machines. *Inf. Comput.*, 253:272–303, 2017. URL: https://doi.org/10.1016/j.ic.2016.07.011, doi:10.1016/j.ic.2016.07.011.

7 Cristiana Chitic and Daniela Rosu. On validation of XML streams using finite state machines. In *Proceedings of the Seventh International Workshop on the Web and Databases, WebDB 2004, June 17-18, 2004, Maison de la Chimie, Paris, France, Colocated with ACM SIGMOD/PODS 2004*, pages 85–90, 2004. URL: https://doi.org/10.1145/1017074.1017096, doi:10.1145/1017074.1017096.

8 Christian Dax and Felix Klaedtke. Alternation elimination by complementation (extended abstract). In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, pages 214–229, 2008. URL: https://doi.org/10.1007/978-3-540-89439-1_16, doi:10.1007/978-3-540-89439-1_16.

9 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toru'nczyk. Energy and mean-payoff games with imperfect information. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-15205-4_22, doi:10.1007/978-3-642-15205-4_22.

10 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

11 Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Model checking succinct and parametric one-counter automata. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-14162-1_48, doi:10.1007/978-3-642-14162-1_48.

12 Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Model checking succinct and parametric one-counter automata. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-14162-1_48, doi:10.1007/978-3-642-14162-1_48.

13 Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Branching-time model checking of parametric one-counter automata. In *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, pages 406–420, 2012. URL: https://doi.org/10.1007/978-3-642-28729-9_27, doi:10.1007/978-3-642-28729-9_27.

14 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, pages 369–383, 2009. URL: https://doi.org/10.1007/978-3-642-04081-8_25, doi:10.1007/978-3-642-04081-8_25.

15 Paul Hunter. Reachability in succinct one-counter games. In Mikolaj Bojanczyk, Slawomir Lasota, and Igor Potapov, editors, *Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings*, volume 9328 of *Lecture Notes in Computer Science*, pages 37–49. Springer, 2015. URL: https://doi.org/10.1007/978-3-319-24537-9_5, doi:10.1007/978-3-319-24537-9_5.

16 Oscar H. Ibarra, Tao Jiang, Nicholas Q. Trân, and Hui Wang. New decidability results concerning two-way counter machines and applications. In *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*, pages 313–324, 1993. URL: https://doi.org/10.1007/3-540-56939-1_82, doi:10.1007/3-540-56939-1_82.

17 Oscar H. Ibarra, Jianwen Su, Zhe Dang, Tevfik Bultan, and Richard A. Kemmerer. Counter machines and verification problems. *Theor. Comput. Sci.*, 289(1):165–189, 2002. URL: https://doi.org/10.1016/S0304-3975(01)00268-7, doi:10.1016/S0304-3975(01)00268-7.

18 Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, pages 308–322, 2005. URL: https://doi.org/10.1007/978-3-540-32033-3_23, doi:10.1007/978-3-540-32033-3_23.

19 Antonia Lechner. Synthesis problems for one-counter automata. In Mikolaj Bojanczyk, Slawomir Lasota, and Igor Potapov, editors, *Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings*, volume 9328 of *Lecture Notes in Computer Science*, pages 89–100. Springer, 2015. URL: https://doi.org/10.1007/978-3-319-24537-9_9, doi:10.1007/978-3-319-24537-9_9.

20 Antonia Lechner. *Extensions of Presburger arithmetic and model checking one-counter automata.* PhD thesis, University of Oxford, 2016.

21 Antonia Lechner, Joël Ouaknine, and James Worrell. On the complexity of linear arithmetic with divisibility. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 667–676, 2015. URL: https://doi.org/10.1109/LICS.2015.67, doi:10.1109/LICS.2015.67.

22 Leonard Lipshitz. The diophantine problem for addition and divisibility. *Transactions of the American Mathematical Society*, pages 271–283, 1978.

23 Leonard Lipshitz. Some remarks on the diophantine problem for addition and divisibility. *Bull. Soc. Math. Belg. Sér. B*, 33(1):41–52, 1981.

24 Ju V Matijasevic. Enumerable sets are diophantine. In *Soviet Math. Dokl.*, volume 11, pages 354–358, 1970.

25 Marvin L. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics*, 74(3):437–455, 1961. URL: http://www.jstor.org/stable/1970290.

26 Guillermo A. Pérez. The fixed initial credit problem for partial-observation energy games is ack-complete. *Inf. Process. Lett.*, 118:91–99, 2017. URL: https://doi.org/10.1016/j.ipl.2016.10.005, doi:10.1016/j.ipl.2016.10.005.

27 Alan J.A. Robinson and Andrei Voronkov. *Handbook of automated reasoning*, volume 1. Gulf Professional Publishing, 2001.

28 Julia Robinson. Definability and decision problems in arithmetic. *The Journal of Symbolic Logic*, 14(2):98–114, 1949.

29 Olivier Serre. Parity games played on transition graphs of one-counter processes. In Luca Aceto and Anna Ingólfsdóttir, editors, *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings*, volume 3921 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2006. URL: https://doi.org/10.1007/11690634_23, doi:10.1007/11690634_23.

30 Moshe Y. Vardi. Reasoning about the past with two-way automata. In *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, pages 628–641, 1998. doi:10.1007/BFb0055090.

A Undecidability of EPAD with a Restricted Alternation

For completeness, we give a proof of Proposition 3 below.

Proof of Proposition 3. We begin by recalling the definition of the $\mathrm{LCM}(\cdot,\cdot,\cdot)$ predicate. A common multiple of $a, b \in \mathbb{Z}$ is an integer $m \in \mathbb{Z}$ such that $a \mid m$ and $b \mid m$. Their least common multiple m is minimal, that is $m \mid m'$ for all common multiples m'. This leads to the following definition of $\mathrm{LCM}(a, b, m)$ for all $a, b, m \in \mathbb{Z}$.

$$\mathrm{LCM}(a, b, m) \iff \forall m' \left((a \mid m') \wedge (b \mid m')\right) \longleftrightarrow (m \mid m'))$$

Observe that the universally-quantified m' appears only on the right-hand side of the divisibility constraints. We thus have that $\exists\forall_R$PAD can be assumed to include a least-common-multiple predicate.[1] For convenience, we will write $\mathrm{LCM}(a, b) = m$ instead of $\mathrm{LCM}(a, b, m)$.

Now, once we have defined the $\mathrm{LCM}(\cdot,\cdot,\cdot)$ predicate, we can define the perfect square relation using the identity:

$$x > 0 \wedge x^2 = y \iff \mathrm{LCM}(x, x+1) = y + x$$

and multiplication via:

$$4xy = (x+y)^2 - (x-y)^2.$$

Observe that we are now able to state Diophantine equations. Undecidability thus follows from the MRDP theorem [24] which states that satisfiability for such equations (i.e. Hilbert's 10th problem) is undecidable. ◀

B Missing proofs from Section 4.2

Proof of Lemma 11. Note that any accepting run γ of the sub-A2A must include at least one of the two finite branches from the claim. We further argue that each branch enforces the corresponding constraints if they appear in γ. Since these are mutually exclusive, it follows that γ includes exactly one of the branches.

If γ includes $(q, i) \rightsquigarrow (q', j)$ then $|w(i)|_\square - 1 = |w(j)|_\square$. The latter implies $i > j >= 0$ since otherwise the position of the head cannot be moved to the left. On the other hand, if γ includes $(q, n) \rightsquigarrow (\mathit{final}, n)$ then γ can only be accepting if $n = 0$. Hence, γ includes $(q, 0) \rightsquigarrow (\mathit{final}, 0)$. ◀

Proof of Lemma 12. We proceed as in the proof of Lemma 11.

If γ includes a branch with the state $q^{=0}$ then γ is accepting if and only if it reaches q'. It can only reach q' with the *first*? transition, i.e. when $i = 0$. Otherwise, it has to include a branch with $q^{\neq 0}$ and reading any letter it reaches *true*. This is only possible if $i > 0$. Since the $(i+1)$-th letter of w is $\square$, the latter means $|w(i)|_\square > 0$. ◀

Proof of Lemma 13. Fix a word $w \in W_X$ with $\square$ as $(i+1)$-th letter. Consider any run tree γ of $\mathcal{T}^{eq}_{sub}$ on w. After reading the first $\square$, suppose γ has a branch leading to the state q'.

[1] We remark that this definition of the least common multiple is oblivious to the sign of m, e.g. $\mathrm{LCM}(2, 3, -6)$ is true and $\mathrm{LCM}(a, b, m) \iff \mathrm{LCM}(a, b, -m)$ in general. This is not a problem since we can add $m \geq 0$ if desired.

It must therefore also have a branch containing $present(x)$. Since, from there, it can only move to the state *true* if it reads x before reading another $\Box$ symbol to the right, we have $V(x) = |w(i)|_\Box$.

If γ has a branch containing $absent(x_i)$, then it is accepting if and only if it reaches *true* after reading another $\Box$ before ever reading x. Hence, $V(x_i) \neq |w(i)|_\Box$. ◀

Proof of Lemma 14. Fix a word $w \in W_X$ with $\Box$ as $(i+1)$-th letter and consider any run tree γ of $\mathcal{T}^{lb}_{sub}$ on w. After reading the first $\Box$, let it suppose it adds a branch checking $= x$ in $\mathcal{T}^{eq}_{sub}$. Then, γ is accepting if and only if it additionally contains a branch to (q', i) and $|w(i)|_\Box = V_w(x)$. If it has the other sub-tree, i.e. it contains $back(x)$, γ is accepting if and only if it reaches the state *true* which is possible only if it reads x after reading a $\Box$ to the left of the current position. It follows that it is accepting if and only if $|w(i)|_\Box > V_w(x)$ and $(q, i) \rightsquigarrow (q', i)$ is part of γ.

If γ instead contains the branch with $front(x_i)$, it is accepting only if it can read x from $search(x)$ after having read a $\Box$ from $front(x)$ to the right of the current position of the input. Hence, $|w(i)|_\Box < V_w(x_i)$. ◀

The proof of correctness for the main result of the section is given below.

Proof of Proposition 9. We have to show that, $L(\mathcal{T}) = \{w \in W_X \mid \text{all infinite } V_w\text{-runs of } \mathcal{A} \text{ reach } q_f\}$. We prove this in two parts:

$\supseteq$: Consider a word $w = a_0a_1a_2 \cdots \in W_X$, such that with valuation V_w all infinite V_w-runs of $\mathcal{A}$ reach q_f. We have to show, that w is accepted by $\mathcal{T}$, i.e., there exists an accepting run tree γ of w on $\mathcal{T}$. We will now grow an accepting run tree γ_{valid}. Since w is a valid parameter word, we can add to γ_{valid} a sub-tree with root labelled by $(s_{in}, 0)$ and a branch extending to $(q_{in}, 0)$ (see Lemma 10).

Consider now a valid infinite run ρ of $\mathcal{A}$ that reaches q_f. Let $\pi = (q_0, op_1, q_1)(q_1, op_2, q_2)\ldots$ be the path induced by ρ such that $\exists i : q_i = q_f$. We further extend γ_{valid} by appending to it, from the $(q_{in}, 0)$-labelled vertex, a sub-tree γ_ρ as follows: For every transition of the form (q_i, op_{i+1}, q_{i+1}) where op_i is an increment or decrement, the corresponding $\mathcal{T}^{inc}_{sub}$ and $\mathcal{T}^{dec}_{sub}$ simulate the path from q_i to q_{i+1} correctly. Also, as every transition in π is valid in ρ (i.e. does not result in negative counter values) using the first part of Lemmas 12, 13, and 14, we can take the *simulation* and *validation* sub-trees of $\mathcal{T}^{zero}_{sub}$, $\mathcal{T}^{eq}_{sub}$, and $\mathcal{T}^{lb}_{sub}$, and append them to our run tree. Since ρ reaches q_f, the simulation branch (obtained by concatenating the simulation branches of all the sub-A2As) of γ_ρ reaches q_f. As ρ was chosen arbitrarily, we have that γ_ρ, for all infinite runs ρ, are accepting. To conclude, we need to deal with run trees arising from maximal finite runs: We construct a sub-tree γ_{fin} appending $\sim$ and *validation* sub-trees for as long as possible. By definition of maximal finite runs, the every such run reaches a point where all possible transitions are disabled. There, we append a *violation* sub-tree which, using the second part of the mentioned lemmas, is accepting. Hence, γ_{valid} is accepting.

$\subseteq$: Consider a word $w \in L(\mathcal{T})$. We have to show that with valuation V_w, every infinite run of $\mathcal{A}$ reaches q_f. We will prove the contrapositive of this statement: Let there exists a valuation V such that there is an infinite run of $\mathcal{A}$ that does not reach q_f, then for all words w with $V_w = V$, $w \notin L(\mathcal{T})$.

Consider ρ to be an infinite run with valuation V_w such that it does not reach q_f. Let ρ induces the path π of the form $(q_{in}, \delta(q_{in}, q_1), q_1)\ldots\{(q_k, \delta(q_k, q_{k+1}), q_{k+1})\ldots(q_n, \delta(q_n, q_k), q_k)\}^\omega$, where none of the q_i's are q_f. Recall that for every δ_i, a run of $\mathcal{T}^{\delta_i}_{sub}$ has one *simulation* branch, one or more *validation* branches or a *violation* branch. Now, as ρ is a valid infinite run of $\mathcal{A}$, every δ_i is valid. Hence, any *violation* branch in any $\mathcal{T}^{\delta_i}_{sub}$ will be non-accepting already. Hence, for every δ_i appearing in π, let us consider the *simulation* and *val* branches. Now, consider the global *simulation* branch b in $\mathcal{T}$: $q_{in} \rightsquigarrow q_1 \rightsquigarrow \dots (q_k \rightsquigarrow \dots q_n)^\omega$. Note that, b is a valid infinite branch in a run in A2A with no final states $\checkmark_{x_i}$ are present. Branch b will be present in every run of w in $\mathcal{T}$, resulting no accepting run for w. ◀

C coNP-hardness result for the Universal Reachability Problem for SOCA

Proof of second part of Proposition 16. Here we give the full reduction from the complement of the SubsetSum problem to the UnivReach problem for SOCA.

Given a set $S = \{a_1, a_2, \dots a_n\} \subseteq \mathbb{N}$ of integers and a target value $t \in \mathbb{N}$, the SubsetSum problem asks whether there exists $S' \subseteq S$ such that $\sum_{a_i \in S'} a_i = t$. Given an instance of the SubsetSum problem with S and t, we create a SOCA $\mathcal{A}$ as depicted in Figure 4. If we let $q_{in} = q_0$, then for every $1 \leq i \leq n$ there are two ways of reaching q_i from q_{i-1}: directly, with constant update $+0$; or via q'_i with total effect $+a_i$. Hence, for every subset $S' \subseteq S$, there exists a path from q_{in} to q_n with counter value $\sum_{a_i \in S'} a_i$. Clearly, if there exists S' such that $\sum_{a_i \in S'} a_i = t$ SubsetSum then there exists an infinite run leading to q_s and the UnivReach problem for the constructed SOCA has a negative answer. If there is no such S' then all infinite runs reach q_f and the UnivReach problem has a positive answer. ◀

D Missing proofs from Section 4.3

Proof of Lemma 17. Let us call an infinite run of $\mathcal{A}$ a *safe run* if it does not reach q_f. Fix a safe run ρ. Let $\pi = (q_0, op_1, q_1)(q_1, op_2, q_2)\dots$ be the path it induces. We denote by $\pi[i,j]$ the infix $(q_i, op_{i+1}, q_{i+1})\dots(q_{j-1}, op_j, q_j)$ of π and by $\pi[i,\cdot]$ its infinite suffix $(q_i, op_{i+1}, q_{i+1})\dots$ Suppose there are $0 \leq m < n \in \mathbb{N}$ such that $\pi[m,n]$ is a cycle that starts from a zero test. Note that if a cycle that starts from a zero test can be traversed once, it can be traversed infinitely many times. Then, the run lifted from the path $\pi[0,m] \cdot \pi[m,n]^\omega$ is our desired safe run. Now, let us assume that π has no cycles which start at a zero test. This means every zero test occurs at most once in π. Since the number of zero tests in $\mathcal{A}$ is finite, we have a finite $k \in \mathbb{N}$ such that there are no zero tests at all in $\pi[k,\cdot]$.

Now, consider $\pi[k,\cdot]$. Suppose it does not witness any non-negative effect cycle, i.e., every cycle in $\pi[k,\cdot]$ is negative. But, we know π lifts to a valid infinite run which means the counter value cannot go below zero. This contradicts our assumption, hence there are $k \leq p < q$ such that $\pi[p,q]$ is a cycle with non-negative effect. It is easy to see that there must be r, s such that $p \leq r < s \leq q$ and $\pi[r,s]$ is a simple non-negative effect cycle. Also note that, $r \geq k$ which means that $\pi[r,s]$ does not have any zero tests. Hence, $\pi[r,s]$ is a simple pumpable cycle. Note that if a pumpable cycle can be traversed once then it can be traversed infinitely many times. Using this fact, the run lifted from $\pi[0,r] \cdot \pi[r,s]^\omega$ is our desired safe run. ◀

E Missing proofs from Section 5.2

E.1 Bound on the length of the accepting run

We first state the following well-known result.

▶ **Proposition 28** (From [18]). *Let $\mathcal{B} = (Q, T, \delta, q_{in})$ be a SOCA. If there exists an accepting run of $\mathcal{B}$ then there exists an accepting run $(q_0, c_0) \dots (q_k, c_k)$ of it such that for all $0 \leq i \leq k$ we have $c_i \leq (|Q| \cdot a_{\max})^3 + (|Q| \cdot a_{\max})^2 + a_{\max}$.*

Now, we prove the bound on the length of the accepting run in SOCAP.

Proof of Lemma 22. Fix a valuation V. From Proposition 28 we know that, there exists an accepting run where counter value is $\leq C_B = (|Q| \cdot a_{\max})^3 + (|Q| \cdot a_{\max})^2 + a_{\max}$. Note that, it is sufficient to prove that, there exists an accepting V-run of length at most $|Q| \cdot C_B$. Let all the accepting V-runs of $\mathcal{A}$ are of length $> |Q| \cdot C_B$. Then, there are two possibilities: the counter value goes above C_B for all the runs, which already rises a contradiction to the claim in Proposition 28. Hence, this is not true and fix the shortest run ρ whose counter value is bounded by C_B and has length $> |Q| \cdot C_B$. The total number of distinct possible configurations possible in ρ is at most $|Q| \cdot C_B$. By the pigeonhole principle, there exists a configuration that appears twice along ρ. Then, we can ignore the infix of the run between these two configurations and create a new run ρ'. Clearly, ρ' is also an accepting V-run whose counter value is still bounded by C_B but shorter in length than ρ, which gives us the contradiction to our assumption. ◀

E.2 Sub-game simulating parametric updates.

Here we describe the family of sub-games $\mathcal{G}_{sub}^{PU}$ in a bit more detail. In Figure 8, we depict a deterministic three-counter automaton, which takes $(0, x, v)$ as input where v is the current counter value and x denotes the valuation $V_\forall(x)$ of x chosen by Adam. The automaton has an accepting run bounded by $v + x$ and thus by $F_2(V_\forall(S))$ in our construction. Hence using Proposition 24, we can construct a blind POEG $\mathcal{G}_{sub}^{+x}$ such that Eve has a winning strategy λ in it. Note that, the construction of $\mathcal{G}_{sub}^{\pm}$ will be similar to our construction of the global game but it will not have any parametric updates. Moreover, every play consistent with λ reach a state in the sub-games of $\mathcal{G}_{sub}^{+x}$ with value $v + x$ and from there reading \$ it can dump everything, reset the simulation in $\mathcal{G}_{sub}^{\pm}$ or it can go back to the corresponding state in $\mathcal{G}_{sub}^{zero}$ in the global game $\mathcal{G}$. The special action \$ takes the place of # in $\mathcal{G}$ and thus, from final states, allows Eve to signal the end of a simulation of an accepting run. Adam can then choose to reset the simulation within the sub-game or to go back to the simulation in the global game $\mathcal{G}$. As Eve has a winning strategy in $\mathcal{G}_{sub}^{\pm}$, we can assume Adam will choose to go back to the simulation in the global game $\mathcal{G}$ to argue for correctness of our construction. In that case, a play returns to $\mathcal{G}_{sub}^{zero}$ with the value $v + x$. For the sub-game $\mathcal{G}_{sub}^{-x}$, we apply the same construction *mutatis mutandis.*

E.3 Sub-games ensuring Eve is simulating valid run

Proof of Lemma 25. Note that, $\mathcal{G}_{sub}^{inf}$ starts with an incoming $\dashv$ transition of weight 0 from the state S' (see Figure 6c) and hence the initial energy level is $F_2(V_\forall(S))$. If Eve plays the action # infinitely often and within $F_2(V_\forall(S))$ actions from Σ_T of each other, all the plays in the sub-game will take the only negatively-weighted transition i.e., the -1 loop at most $F_2(V_\forall(S))$ times. Hence, all plays in $\mathcal{G}_{sub}^{inf}$ will never have a negative energy level. If, however, Eve plays in any other way (eventually stopping with the letter # or taking too long to produce it) then there will be a play which reaches the state with the -1 loop and take the negative transition enough times to drop the energy level below 0. ◀

Proof of Lemma 26. According to our construction of the sub-game $\mathcal{G}_{sub}^{seq}$, if the first action played by Eve is of the form $(q, \cdot, q')$ and second action is of the form $(q', \cdot, q'')$ for all

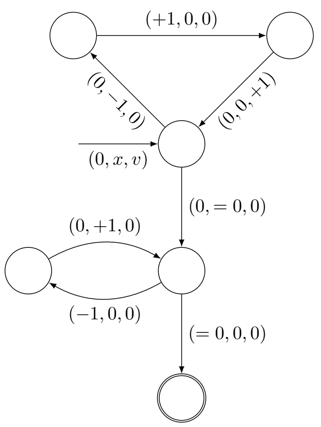


Figure 8 Deterministic 3-counter machine simulating $+x$

successors q'' of q' then that play moves to the states with the 0 loop and the energy level always remains non-negative. If not, reading anything else from Σ_T, it will move to the state with -1 loop. Note that, all copies of $\mathcal{G}_{sub}^{seq}$ are in the same observation-set and Eve does not know, which one of them has been chosen by Adam. If she violates the order of the sequence of transitions at any point, say from state q_i, then there will be a play consistent with her strategy which in the corresponding copy of $\mathcal{G}_{sub}^{seq}$ reaches the -1 loop and she loses. ◀

Proof of Lemma 27 . In $\mathcal{G}_{sub}^{zero}$, if Eve does a zero cheat then there exists a play that goes to the state at the top-right corner simulates the inverse of the operations and come back to the initial state with a a zero cheat. Note that, this play has energy level at most $F_2(V_\forall(S)) - 1$. Also, if she does a positive cheat then there exists a play that goes to the bottom-left corner of the sub-game, simulates the operations and comeback to the initial state with a positive cheat. This play also has energy level at most $F_2(V_\forall(S)) - 1$. This shows that, if Eve cheats more than $F_2(V_\forall(S))$ times, there will be a play in the sub-game with negative energy level. But, Eve has to simulate the run infinitely many times and hence, she cannot cheat forever.

If, however, she correctly simulates the zero tests, then a play can forever stay in the initial state; in which case it will never have a negative energy level or it can move to the top-right or bottom-left corner at some point. There, if Eve is simulating an accepting V-run ρ, she will play # again in at most $|\rho| + 1$ steps. If the play is still in one those corners at that moment, then we know it moves to the bottom-right state and that it must have seen a weight of -1 at most ρ times. Clearly, once there, the play can no longer have a negative energy level. If the play returned to the initial state before that then, since Eve is not cheating, the energy level of the play must have been at least $F_2(V_\forall(S)) - 1$ and it must have seen a weight of -1 at most $|\rho|$ times. ◀

E.4 Reachability Synthesis for SOCAP

Proof of Lemma 20. For all valuations V let there be accepting V-run in $\mathcal{A}$. Using Lemma 22, the accepting run has length bounded and in our game specifically by $F_2(V_\forall(S))$. Let Adam chooses a valuation $V_\forall$ in $\mathcal{G}_{sub}^{val}$. There exists an accepting $V_\forall$-run ρ in $\mathcal{A}$ where $|\rho| \leq F_2(V_\forall(S))$. We know that Eve has a winning strategy μ in $\mathcal{G}_{sub}^{PU}$ such that she will faithfully simulate all the parametric updates. Moreover if Adam stays in $\mathcal{G}_{sub}^{PU}$ forever, then Eve wins automatically with strategy μ. Hence without loss of generality, we assume that Adam leaves $\mathcal{G}_{sub}^{PU}$ whenever Eve plays \$ after faithfully simulating the parametric update and comes back to $\mathcal{G}$. Let Eve's strategy to get the energy level up to $F_2(V_\forall(S))$ be λ in $\mathcal{G}_{sub}^{pump}$. Then after the initial phase in G, Eve can play the strategy $\lambda \dashv (\#w)^\omega$ such that $\rho = w|_T$. Note that, every play consistent with this strategy, satisfies all the conditions in the sub-games. Hence, Eve will win in every sub-games and hence in the global game G.

Conversely, let there exists a valuation V such that there is no reaching run in $\mathcal{A}$. We have to show that Eve does not have a winning observation-based strategy in $\mathcal{G}$ in this case. We will prove it by contradiction. Let Eve has a winning strategy λ in $\mathcal{G}$. Adam chooses a valuation $V'_\forall$ such that $V'_\forall = V$. Note that along all plays consistent with λ, Eve has to simulate a valid run of $\mathcal{A}$ or there will be a play with negative energy level in one of the sub-games. Then the only possibility is that she is faithfully simulating a run which is not accepting resulting she will not see $\#$ within a $\leq F_2(V_\forall(S))$ actions of Σ_T of each other in $\mathcal{G}_{sub}^{inf}$ and then, negative energy level can be reached in that sub-game and she loses. Hence, she has no observation-based winning strategy. ◀

Proof of Theorem 18. Given a SOCAP $\mathcal{A} = (Q, T, \delta, q_{in}, X)$, we can construct $\mathcal{A}^2 = (Q \times Q \cup \{\cdot\}, T', \delta', (q_{in}, \cdot), X)$ such that, if $(q_1, q_2) \in T$, then for all $q \in Q$, $((q, q_1), (q, q_2)) \in T'$, $((q_1, \cdot), (q_2, \cdot)) \in T'$ such that $\delta'((q, q_1), (q, q_2)) = \delta'((q_1, \cdot), (q_2, \cdot)) = \delta(q_1, q_2)$. Also, for every state $q \in Q$, $(q, \cdot) \xrightarrow{+0} (q, q) \in T'$.

Recall that the SynthReach problem asks whether there exists a valuation such that all runs reach q_f from a given configuration $\chi = (q_0, c_0)$. From Lemma 17 we infer that, if SynthReach is not true then for all valuations there exists an infinite run in $\mathcal{A}$ that induces a path of the form $\pi_0 \cdot \pi_1^\omega$ or $\pi_0 \cdot \pi_2^\omega$ such that π_1 starts with a zero test and π_2 is a simple pumpable cycle.

Now we will create POEG $\mathcal{G}$ from $\mathcal{A}^2$ where Eve tries to simulate corresponding π_0 and π_1 or π_2. The game will have three sub-games: $\mathcal{G}_{\pi_0}$, $\mathcal{G}_{\pi_1}$, $\mathcal{G}_{\pi_2}$ where Eve tries to simulate corresponding π_0, π_1 or π_2. The main idea is as follows: From every state in $\mathcal{G}_{\pi_0}$ there is a transition of $\#_1$ to move to the similar state in the similar sub-game of $\mathcal{G}_{\pi_1}$ or to go back to the initial states of the sub-games in $\mathcal{G}_{\pi_0}$. At first Adam chooses a valuation in the first phase of the game as per our construction. Let the run that Eve tries to simulate for the valuation be of the form $\pi_0 \cdot \pi_1^\omega$, where π_0 ends at state q and π_1 starts with a zero test from q. Then, Eve has to simulate π_0 faithfully in $\mathcal{G}_{\pi_0}$ and then reading $\#_1$ from q, where Adam has two choice: he can reset the simulation of π_0 in $\mathcal{G}_{\pi_0}$ or he can move to $\mathcal{G}_{\pi_1}$, where Eve has to faithfully simulate π_1. The condition that π_1 starts with zero test can be checked easily by minor modifications to the sub-game $\mathcal{G}_{sub}^{seq}$ in $\mathcal{G}_{\pi_1}$. Note that, from every state of $\mathcal{G}_{\pi_0}$ there is a transition to the corresponding state in $\mathcal{G}_{\pi_1}$, so the choice of where to stop π_0 and start π_1 depends on the choice of Eve. Hence if Eve cheats in simulating π_0 then there is a play in $\mathcal{G}_{\pi_0}$ where Adam wins and if she cheats in π_1, then Adam wins in $\mathcal{G}_{\pi_1}$. Hence, Eve can only win if and only if she faithfully simulates a run of the required form.

The case for π_2 is similar. From every state of $\mathcal{G}_{\pi_0}$, there will be another transition reading $\#_2$ where Adam has a choice to move to $\mathcal{G}_{\pi_2}$ or resets simulation in $\mathcal{G}_{\pi_0}$. Note that, the fact that π_2 is a pumpable cycle also can be checked easily as Eve has to simulate π_2 infinitely often in $\mathcal{G}_{\pi_2}$ and this time as the start of every simulation we do not reset the counter value. Hence, π_2 can be simulated infinitely if and only if it is pumpable.

◀

Contents

1 Introduction **1**

2 Undecidability of a Restriction of AEPAD 2

3 One-Counter Automata Models 3
3.1 Paths, runs, and reachability . . . 3
3.2 Model taxonomy . . . 4
3.3 Decision problems . . . 4

4 One-Counter Automata with Parametric Tests 5
4.1 Alternating two-way automata . . . 5
4.2 From parametric one-counter to alternating two-way automata . . . 6
4.3 An upper bound for reachability synthesis . . . 8

5 Succinct and Parametric One-Counter Automata 9
5.1 Partial-observation energy games . . . 10
5.2 From parametric counter automata to energy games . . . 10

6 Conclusion 14

A Undecidability of EPAD with a Restricted Alternation 18

B Missing proofs from Section 4.2 18

C coNP-hardness result for the Universal Reachability Problem for SOCA 20

D Missing proofs from Section 4.3 20

E Missing proofs from Section 5.2 20
E.1 Bound on the length of the accepting run . . . 20
E.2 Sub-game simulating parametric updates. . . . 21
E.3 Sub-games ensuring Eve is simulating valid run . . . 21
E.4 Reachability Synthesis for SOCAP . . . 22