Branching-Time Model Checking of One-Counter Processes and Timed Automata *

Stefan Göller[1] and Markus Lohrey[2,**]

[1] Universität Bremen, Fachbereich Mathematik und Informatik, Germany
[2] Universität Leipzig, Institut für Informatik, Germany
goeller@informatik.uni-bremen.de lohrey@informatik.uni-leipzig.de

Abstract. One-counter automata (OCA) are pushdown automata which operate only on a unary stack alphabet. We study the computational complexity of model checking computation tree logic (CTL) on transition systems induced by one-counter automata. A PSPACE upper bound is inherited from the modal μ-calculus for this problem proven by Serre. First, we analyze the periodic behaviour of CTL over OCA and derive a model checking algorithm whose running time is exponential only in the number of control locations and a syntactic notion of the formula that we call leftward until depth. In particular, model checking fixed OCA against CTL formulas with a fixed leftward until depth is in P. This generalizes a corresponding recent result of the first author, Mayr and To for the expression complexity of CTL's fragment EF. Second, we prove that already over some fixed OCA, CTL model checking is PSPACE-hard, i.e., expression complexity is PSPACE-hard. Third, we show that there already exists a fixed CTL formula for which model checking of OCA is PSPACE-hard, i.e., data complexity is PSPACE-hard as well. To obtain the latter result, we employ two results from complexity theory: (i) Converting a natural number in Chinese remainder presentation into binary presentation is in logspace-uniform NC^1 and (ii) PSPACE is AC^0-serializable.We demonstrate that our approach can be used to obtain further results. We show that model-checking CTL's fragment EF over OCA is hard for P^{NP}, thus establishing a matching lower bound. We moreover show that the following problem is hard for PSPACE: Given a one-counter Markov decision process, a set of target states with counter value zero each, and an initial state, to decide whether the probability that the initial state will eventually reach one of the target states is arbitrarily close to 1. This improves a recently proven lower bound for every level of the boolean hierarchy shown by Brazdil et al. Finally, we prove that there is a fixed CTL formula for which model checking 2-clock timed automata is PSPACE-hard, generalizing a PSPACE-hardness result for the combined complexity by Laroussinie et al.

1 Introduction

Pushdown automata (PDA) (or recursive state machines, RSMs) are a natural model for sequential programs with recursive procedure calls, and their verification problems have been studied extensively. The reachability problem for PDA can be solved in polynomial time [6, 14]. The complexity of model checking problems for PDA is quite well understood in terms of combined complexity, data complexity, and expression complexity:[3] The combined complexity of the model checking problem for the modal μ-calculus over PDA was shown to be EXPTIME-complete in [37], and the global version of the model checking problem has been considered in [29, 9, 28]. Moreover, the EXPTIME lower bound even holds for both the data and expression complexity for the simpler logic CTL and its fragment EG over PDA [36]. On the other hand, the combined complexity of the model checking problem for the logic EF (another natural fragment of CTL) over PDA is PSPACE-complete [36], and again the lower bound still holds for both the data and expression complexity [6]. Model checking problems for various fragments and extensions of PDL (propositional dynamic logic) over PDA were studied in [16].

* An extended abstract of this paper has appeared in the proceedings of STACS 2010 [17].
** The second author would like to acknowledge the support by DFG research project GELO.

[3] For a given class of systems (structures) $\mathcal{C}$ and a logic $\mathcal{L}$ one distinguishes three settings for the model-checking problem for $\mathcal{C}$ and $\mathcal{L}$, i.e., the question whether a formula $\varphi \in \mathcal{C}$ holds in a structure $A \in \mathcal{C}$: (i) the input consists of φ and A, (ii) φ is fixed, and the input only consists of A, and (iii) A is fixed, and the input only consists of φ. The combined complexity of the model-checking problem is the complexity in setting (i), whereas the data (resp. expression) complexity is the complexity in setting (ii) (resp. (iii)). The main motivation for studying the data complexity is that in many applications the formula φ is small.

1.1 One-Counter Automata

One-counter automata (OCA) are Minsky counter machines with just one counter and action labels on the transitions. They can also be seen as a special case of PDA with just one stack symbol, plus a non-removable bottom symbol which indicates an empty stack (and thus allows to test the counter for zero) and hence constitute a natural and fundamental computational model. In recent years, model checking problems for OCA received increasing attention [18, 19, 30, 32]. Clearly, all upper complexity bounds carry over from PDA. The question, whether these upper bounds can be matched by lower bounds was just recently solved for several important logics: Model checking μ-calculus on OCA is PSPACE-complete. The PSPACE upper bound was shown in [30], and a matching lower bound can easily be shown by a reduction from emptiness of alternating unary finite automata, which was shown to be PSPACE-complete in [23, 24]. This lower bound even holds if either the OCA or the formula is fixed. The situation becomes different for the fragment EF. In [18], it was shown that model checking EF over OCA is in the complexity class P^{NP} (the class of all problems that can be solved on a deterministic polynomial time machine with access to an oracle from NP). Moreover, if the input formula is represented succinctly as a DAG (directed acyclic graph), then model checking EF over OCA is also hard for P^{NP}. For the standard (and less succinct) tree representation for formulas, only hardness for the class $\mathsf{P}^{\mathsf{NP}[\log]}$ (the class of all problems that can be solved on a deterministic polynomial time machine which is allowed to make $O(\log n)$ many queries to an oracle from NP) was shown in [18]. In fact, there already exists a fixed EF formula such that model checking this formula over a given OCA is hard for $\mathsf{P}^{\mathsf{NP}[\log]}$, i.e., the data complexity is $\mathsf{P}^{\mathsf{NP}[\log]}$-hard.

In this paper we consider the model checking problem for CTL on OCA. By the known upper bound for the modal μ-calculus [30] this problem belongs to PSPACE. First, we analyze the combinatorics of CTL model checking over OCA. More precisely, we analyze the periodic behaviour of the set of natural numbers that satisfy a given CTL formula in a given control location of the OCA (Theorem 1). By making use of Theorem 1, we can derive a model checking algorithm whose running time is exponential only in the number of control locations and a syntactic measure on CTL formulas that we call leftward until depth (Theorem 2). As a corollary, we obtain that model checking a fixed OCA against CTL formulas of fixed leftward until depth lies in P (Corollary 3). This generalizes a recent result from [18], where it was shown that the expression complexity of EF over OCA lies in P. Next, we focus on lower bounds. We show that model checking CTL over OCA is PSPACE-complete, even if we fix either the OCA (Theorem 11) or the CTL formula (Theorem 18). The proof for Theorem 11 uses a subtle reduction from QBF. We have to construct a fixed OCA for which we can construct for a given unary encoded number i CTL formulas that express, when interpreted over our fixed OCA, whether the current counter value is divisible by 2^i and whether the i^{th} bit in the binary representation of the current counter value is 1, respectively. For the proof of Theorem 18 (PSPACE-hardness of data complexity for CTL) we use two techniques from complexity theory, which to our knowledge have not been applied in the context of verification so far:

- the existence of small depth circuits for converting a number from Chinese remainder representation to binary representation (see Section 6.1 for details) and
- the fact that PSPACE-computations are serializable in a certain sense (see Section 6.2 for details).

One of the main obstructions in getting lower bounds for OCA is the fact that OCA are well suited for testing divisibility properties of the counter value and hence can deal with numbers in Chinese remainder representation, but it is not clear how to deal with numbers in binary representation. Small depth circuits for converting a number from Chinese remainder representation to binary representation are the key in order to overcome this obstruction.

We are confident that our new lower bound techniques described above can be used for proving further lower bounds for OCA and related models. We present three other applications of our techniques:

- We show that model checking EF over OCA is complete for P^{NP} even if the input formula is represented by a tree (Theorem 21) and thereby solve an open problem from [18]. Figure 1 summarizes the picture on the complexity of model checking for PDA and OCA.
- We improve a lower bound on a decision problem for one-counter Markov decision processes from [8] (Theorem 25). More details on this problem are provided below.

Logic	PDA	OCA
μ-calculus	EXPTIME	PSPACE
μ-calculus, fixed formula	EXPTIME	PSPACE
μ-calculus, fixed system	EXPTIME	PSPACE
CTL, fixed formula	EXPTIME	PSPACE (*)
CTL, fixed system	EXPTIME	PSPACE (*)
CTL, fixed system, fixed leftward until depth	EXPTIME	in P (*)
EF	PSPACE	P^{NP} (*)
EF, fixed formula	PSPACE	$\mathsf{P}^{\mathsf{NP}[\log]}$ hard
EF, fixed system	PSPACE	in P

Table 1. Model checking over PDA and OCA; our new results are marked with (*).

– We prove that there is a fixed CTL formula for which model checking 2-clock timed automata is PSPACE-complete. This improves a PSPACE lower bound for the combined complexity of model checking CTL on 2-clock timed automata from [25]. Furthermore, we show that reachability for very restricted 2-clock timed automata that allow modulo tests in the transitions is already PSPACE-hard, improving a

1.2 Markov Decision Processes

Markov decision processes (MDPs) extend classical Markov chains by allowing so called *nondeterministic vertices*. In these vertices, no probability distribution on the outgoing transitions is specified. The other vertices are called *probabilistic vertices*; in these vertices a probability distribution on the outgoing transitions is given. The idea is that in an MDP a player Eve plays against nature (represented by the probabilistic vertices). In each nondeterministic vertex v, Eve chooses a probability distribution on the outgoing transitions of v; this choice may depend on the past of the play (which is a path in the underlying graph ending in v) and is formally represented by a strategy for Eve. An MDP together with a strategy for Eve defines an ordinary Markov chain, whose state space is the unfolding of the graph underlying the MDP. In Section 9, we consider infinite MDPs, which are finitely represented by one-counter automata; this formalism was introduced in [8] under the name *one-counter Markov decision process* (OC-MDP). For a given OC-MDP $\mathcal{M}$ and a set R of control locations of $\mathcal{M}$ (a so called *reachability constraint*) the following two sets $\mathsf{ValOne}(R)$ and $\mathsf{OptValOne}(R)$ were considered in [8]: $\mathsf{ValOne}(R)$ is the set of all states s of the MDP defined by $\mathcal{M}$ such that for every $\varepsilon > 0$ there exists a strategy σ for Eve under which the probability of finally reaching from s a control location in R and at the same time having counter value 0 is at least $1 - \varepsilon$. $\mathsf{OptValOne}(R)$ is the set of all states s of the MDP defined by $\mathcal{M}$ for which there exists a specific strategy for Eve under which this probability becomes 1. It was shown in [8] that for a given OC-MDP $\mathcal{M}$, a set of control locations R, and a state s of the MDP defined by $\mathcal{M}$,

– the question whether $s \in \mathsf{OptValOne}(R)$ is PSPACE-hard and in EXPTIME, and
– the question whether $s \in \mathsf{ValOne}(R)$ is hard for every level of the boolean hierarchy BH.

The boolean hierarchy is a hierarchy of complexity classes between NP and $\mathsf{P}^{\mathsf{NP}[\log]}$, see Section 6 for a definition. We use our lower bound techniques (based on the serializability of PSPACE + small depth circuits for converting numbers from Chinese remainder representation to binary representation) in order to improve the second hardness result for the levels of BH to PSPACE-hardness. As a byproduct, we also reprove PSPACE-hardness for $\mathsf{OptValOne}(R)$. Currently, it is open, whether $\mathsf{ValOne}(R)$ is decidable; the corresponding problem for MDPs defined by pushdown automata is undecidable [15].

1.3 Timed Automata

Timed automata were introduced by Alur and Dill [1] and can be seen as an extension of finite automata by allowing the usage of real-time clocks. Timed automata are one of the most important formalisms for

modeling real-time systems. In [1] it was shown that the reachability (i.e. emptiness) problem for timed automata is PSPACE-complete. PSPACE-hardness already holds when only three clocks are present as shown by Courcoubetis and Yannakakis [13]. The precise computational complexity of reachability for 2-clock timed automata is still a major open problem. The best-known lower bound is NP-hardness [25], whereas PSPACE the the best-known upper bound for this problem. It is interesting to note that concerning the reachability problem, there is a close connection between bounded counter automata and timed automata as recently shown by Haase et al. [20]: the reachability problem of n-clock timed automata is equivalent to the the reachability problem of bounded $(n-1)$-counter automata with respect to logarithmic space reductions.

It was shown in [27] that the reachability problem for 2-clock timed automata with modulo tests on counter values is PSPACE-hard. For the lower bound proof in [27] it is crucial that the numerical constants that appear in the transitions of the timed automaton are encoded in binary. We improve the lower bound from [27] by showing that the reachability problem for 2-clock timed automata with modulo tests is already PSPACE-hard when the occuring numbers are encoded in unary. It shows that very simple extensions of the reachability problem of timed automata with two clocks are PSPACE-hard. In [25] it has been shown that model checking CTL on timed automata with two clocks (but without modulo tests) is PSPACE-hard (and PSPACE-complete). We prove that already the data complexity of this problem is PSPACE-hard.

1.4 Organization

The paper is organized as follows. In Section 2 we introduce general notation. In Section 3 we define one-counter automata and the branching-time logic CTL. Periodicity of CTL on OCA and a derived model checking algorithm is the content of Section 4. In Section 5 we give a fixed zero-test-free one-counter automaton (which is basically a one-counter automaton that cannot test if the counter is zero) for which CTL model checking is PSPACE-hard. Section 6 recalls tools from complexity theory that we need in subsequent sections. We show that there already exists a fixed CTL formula for which model checking over zero-test-free one-counter automata is PSPACE-hard in Section 7. The proof technique for this result is applied in the subsequent section and yields the following further results: (i) Model checking the CTL fragment EF over zero-test-free one-counter automata is P^{NP}-hard (Section 8), (ii) membership in $\mathsf{ValOne}(R)$ over one-counter Markov decision processes is PSPACE-hard. (Section 9), (iii) model checking CTL over timed automata with only two clocks is PSPACE-hard already for a fixed CTL formula (Section 10.1), and (iv) reachability for 2-clock timed automata that allow modulo tests in the transitions is PSPACE-hard (Section 10.2). Finally, we reformulate AC^0-serializability of PSPACE in the appendix.

An extended abstract of this paper has appeared as [17].

2 Preliminaries

We denote the naturals by $\mathbb{N} = \{0, 1, 2, \ldots\}$ and the rational numbers by $\mathbb{Q}$. Let $\mathbb{R}_+$ be the positive real numbers (0 is included). For each $i, j \in \mathbb{N}$ we define $[i, j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$ and $[j] = [1, j]$. In particular $[0] = \emptyset$. For each $n \in \mathbb{N}$ and each position $i \geq 1$, let $\mathsf{bit}_i(n)$ denote the i^{th} least significant bit of the binary representation of n, i.e., $n = \sum_{i \geq 1} 2^{i-1} \cdot \mathsf{bit}_i(n)$. For every finite and non-empty subset $M \subseteq \mathbb{N} \setminus \{0\}$, define $\mathsf{LCM}(M)$ to be the *least common multiple* of all numbers in M. Due to a result of Nair [26] it is known that $2^k \leq \mathsf{LCM}([k]) \leq 4^k$ for all $k \geq 9$. As usual, for (a possibly infinite) alphabet A, A^* denotes the set of all finite words over A, A^+ denotes the set of all finite non-empty words over A, and A^ω denotes the set of all infinite words over A. Let $A^\infty = A^* \cup A^\omega$. The length of a finite word w is denoted by $|w|$. For a word $w = a_1 a_2 \cdots a_n \in A^*$ (resp. $w = a_1 a_2 \cdots \in A^\omega$) with $a_i \in A$ and $i \in [n]$ (resp. $i \geq 1$), we denote by w_i the i^{th} letter a_i. A (possibly infinite) directed graph $G = (V, E)$ (with $E \subseteq V \times V$) is called *deadlock-free* if for all $v \in V$ there exists $v' \in V$ with $(v, v') \in E$. If for all $v \in V$ there are only finitely many $v' \in V$ with $(v, v') \in E$, then G is called *image-finite*. The set of all *finite paths in* G is the set $\mathsf{path}_+(G) = \{\pi \in V^+ \mid \forall i \in [|\pi| - 1] : (\pi_i, \pi_{i+1}) \in E\}$. The set of all *infinite paths in* G is the set $\mathsf{path}_\omega(G) = \{\pi \in V^\omega \mid \forall i \geq 1 : (\pi_i, \pi_{i+1}) \in E\}$. A nondeterministic finite automaton (NFA) is a tuple $A = (S, \Sigma, \delta, s_0, S_f)$, where S is a finite set of *states*, Σ is a *finite alphabet*, $\delta \subseteq S \times \Sigma \times S$ is the *transition relation*, $s_0 \in S$ is the *initial state*, and $S_f \subseteq S$ is a set of *final states*. We assume that the reader has some basic knowledge in complexity theory, see e.g. [2] for more details.

3 One-counter automata and computation tree logic

Fix some countable set $\mathcal{P}$ of *atomic propositions*. A *transition system* is a triple $\mathcal{T} = (S, \{S_p \mid p \in \mathcal{P}\}, \to)$, where $(S, \to)$ is a directed graph and $S_p \subseteq S$ for all $p \in \mathcal{P}$ with $S_p = \emptyset$ for all but finitely many $p \in \mathcal{P}$. Elements of S (resp. $\to$) are also called *states* (resp. *transitions*). In case $s \in S_p$, we also say that s is *p-labeled*. We prefer to use the infix notation $s_1 \to s_2$ instead of $(s_1, s_2) \in \to$. For $x \in \{+, \omega\}$ let $\mathsf{path}_x(\mathcal{T}) = \mathsf{path}_x(S, \to)$. For a subset $U \subseteq S$ of states, a (finite or infinite) path π is called a *U-path* if $\pi \in U^\infty$.

A *one-counter automaton* (OCA) is a tuple $\mathcal{O} = (Q, \{Q_p \mid p \in \mathcal{P}\}, \delta_0, \delta_{>0})$, where Q is a finite set of *control locations*, $Q_p \subseteq Q$ for each $p \in \mathcal{P}$ but $Q_p = \emptyset$ for all but finitely many $p \in \mathcal{P}$, $\delta_0 \subseteq Q \times \{0,1\} \times Q$ is a finite set of *zero transitions*, and $\delta_{>0} \subseteq Q \times \{-1, 0, 1\} \times Q$ is a finite set of *positive transitions*. The *size* of this OCA is defined as $|\mathcal{O}| = |Q| + \sum_{p \in \mathcal{P}} |Q_p| + |\delta_0| + |\delta_{>0}|$. We say that $\mathcal{O}$ is *zero-test-free* if $\delta_0 = \delta_{>0} \cap (Q \times \{0,1\} \times Q)$. Hence, for a zero-test-free OCA $\mathcal{O}$ the set δ_0 is implicitly defined by the set $\delta_{>0}$. Therefore, we will write a zero-test-free OCA as a tuple $(Q, \{Q_p \mid p \in \mathcal{P}\}, \delta)$ and identify this tuple with the OCA $(Q, \{Q_p \mid p \in \mathcal{P}\}, \delta \cap (Q \times \{0,1\} \times Q), \delta)$. A one-counter automaton $\mathcal{O} = (Q, \{Q_p \mid p \in \mathcal{P}\}, \delta_0, \delta_{>0})$ defines a (one-counter) transition system $\mathcal{T}(\mathcal{O}) = (Q \times \mathbb{N}, \{Q_p \times \mathbb{N} \mid p \in \mathcal{P}\}, \to)$, where $(q, n) \to (q', n + k)$ if and only if either $n = 0$ and $(q, k, q') \in \delta_0$, or $n > 0$ and $(q, k, q') \in \delta_{>0}$.

More details on CTL and EF can be found for instance in [3]. *Formulas* φ of the logic CTL are given by the following grammar, where $p \in \mathcal{P}$:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{EX}\varphi \mid \mathsf{E}\varphi\mathsf{U}\varphi \mid \mathsf{E}\varphi\mathsf{WU}\varphi.$$

Given a transition system $\mathcal{T} = (S, \{S_p \mid p \in \mathcal{P}\}, \to)$ and a CTL formula φ, we define the semantics $[\![\varphi]\!]_\mathcal{T} \subseteq S$ by induction on the structure of φ as follows:

$$\begin{aligned}
[\![p]\!]_\mathcal{T} &= S_p \quad \text{for each } p \in \mathcal{P} \\
[\![\neg\varphi]\!]_\mathcal{T} &= S \setminus [\![\varphi]\!]_\mathcal{T} \\
[\![\varphi_1 \wedge \varphi_2]\!]_\mathcal{T} &= [\![\varphi_1]\!]_\mathcal{T} \cap [\![\varphi_2]\!]_\mathcal{T} \\
[\![\mathsf{EX}\varphi]\!]_\mathcal{T} &= \{s \in S \mid \exists s' \in [\![\varphi]\!]_\mathcal{T} : s \to s'\} \\
[\![\mathsf{E}\varphi_1\mathsf{U}\varphi_2]\!]_\mathcal{T} &= \{s \in S \mid \exists \pi \in \mathsf{path}_+(\mathcal{T}) : \pi_1 = s, \pi_{|\pi|} \in [\![\varphi_2]\!]_\mathcal{T}, \forall i \in [|\pi| - 1] : \pi_i \in [\![\varphi_1]\!]_\mathcal{T}\} \\
[\![\mathsf{E}\varphi_1\mathsf{WU}\varphi_2]\!]_\mathcal{T} &= [\![\mathsf{E}\varphi_1\mathsf{U}\varphi_2]\!]_\mathcal{T} \cup \{s \in S \mid \exists \pi \in \mathsf{path}_\omega(\mathcal{T}) : \pi_1 = s, \forall i \geq 1 : \pi_i \in [\![\varphi_1]\!]_\mathcal{T}\}
\end{aligned}$$

We write $(\mathcal{T}, s) \models \varphi$ as an abbreviation for $s \in [\![\varphi]\!]_\mathcal{T}$. When additionally $\mathcal{T}$ is clear from the context, we just write $s \models \varphi$. We introduce the usual abbreviations $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\mathtt{true} = p \vee \neg p$ for some $p \in \mathcal{P}$, $\mathsf{AX}\varphi = \neg\mathsf{EX}\neg\varphi$, $\mathsf{EF}\varphi = \mathsf{E}\mathtt{true}\mathsf{U}\varphi$, and $\mathsf{EG}\varphi = \mathsf{E}\varphi\mathsf{WU}\mathtt{false}$. Formulas of the CTL-fragment EF are given by the following grammar, where $p \in \mathcal{P}$:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{EX}\varphi \mid \mathsf{EF}\varphi.$$

Define the *size* $|\varphi|$ of CTL formulas φ inductively as follows: $|p| = 1$, $|\neg\varphi| = |\varphi| + 1$, $|\varphi_1 \wedge \varphi_2| = |\varphi_1| + |\varphi_2| + 1$, $|\mathsf{EX}\varphi| = |\varphi| + 1$, and $|\mathsf{E}\varphi_1\mathsf{U}\varphi_2| = |\mathsf{E}\varphi_1\mathsf{WU}\varphi_2| = |\varphi_1| + |\varphi_2| + 1$.

4 CTL on OCA: Periodic behaviour and upper bounds

The goal of this section is to prove a periodicity property of CTL over one-counter transition systems. We will use this property in order to establish an upper bound for CTL on OCA, see Theorem 2. As a corollary, we show that for a fixed one-counter automaton, CTL model checking restricted to formulas of fixed leftward until depth (see the definition below) can be done in polynomial time, see Corollary 3. For

this, let us define the *leftward until depth* lud of CTL formulas inductively as follows:

$$\begin{aligned}
\mathsf{lud}(p) &= 0 \text{ for each } p \in \mathcal{P} \\
\mathsf{lud}(\neg\varphi) &= \mathsf{lud}(\varphi) \\
\mathsf{lud}(\varphi_1 \wedge \varphi_2) &= \max\{\mathsf{lud}(\varphi_1), \mathsf{lud}(\varphi_2)\} \\
\mathsf{lud}(\mathsf{EX}\varphi) &= \mathsf{lud}(\varphi) \\
\mathsf{lud}(\mathsf{E}\varphi_1\mathsf{U}\varphi_2) &= \max\{\mathsf{lud}(\varphi_1)+1, \mathsf{lud}(\varphi_2)\} \\
\mathsf{lud}(\mathsf{E}\varphi_1\mathsf{W}\mathsf{U}\varphi_2) &= \max\{\mathsf{lud}(\varphi_1)+1, \mathsf{lud}(\varphi_2)\}
\end{aligned}$$

A similar definition of the until depth can be found in [31], but there the until depth of $\mathsf{E}\varphi_1\mathsf{U}\varphi_2$ is 1 plus the maximum of the until depths of φ_1 and φ_2. Note that $\mathsf{lud}(\varphi) \leq 1$ for each EF formula φ.

Let us fix some one-counter automaton $\mathcal{O} = (Q, \{Q_p \mid p \in \mathcal{P}\}, \delta_0, \delta_{>0})$ for the rest of this section. Let us introduce a bit more notation. Let $\odot \in \{+, -\}$, let $\xi \in \mathbb{N}$, and let $\pi = (q_1, n_1) \to (q_2, n_2) \cdots \to (q_k, n_k)$ (resp. $\pi = (q_1, n_1) \to (q_2, n_2) \to \cdots$) be a finite (resp. infinite) path in $\mathcal{T}(\mathcal{O})$ such that moreover $n_i, n_i \odot \xi > 0$ for all i. Define $\pi \odot \xi$ to be the path that emerges from π by replacing each n_i by $n_i \odot \xi$. For each position i and j of π with $i \leq j$, define $\pi[i,j]$ to be the subpath of π that begins in (q_i, n_i) and that ends in (q_j, n_j).

We aim to prove the following: For each CTL formula φ we can compute some threshold $t(\varphi)$ and some period K_φ, where $t(\varphi), K_\varphi \leq 2^{(|\mathcal{O}|\cdot|\varphi|)^{O(1)}}$, such that for all $n \in \mathbb{N}$ with $n > t(\varphi)$ only n's residue class modulo K_φ determines whether $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$ or not, where $q \in Q$ is an arbitrary control location. The goal of this section is to give rather precise bounds on the size of the threshold $t(\varphi)$ and the period K_φ embracing the notion of leftward until depth from above.

Let us assume that $|Q| = k$. Define $K = \mathsf{LCM}([k])$ and $K_\varphi = K^{\mathsf{lud}(\varphi)}$ for each CTL formula φ.

Theorem 1. *Let φ be a* CTL *formula. Then we can compute in polynomial time a threshold*

$$t(\varphi) \leq 2 \cdot |\varphi| \cdot k^2 \cdot K_\varphi$$

such that for all $n, n' > t(\varphi)$ that satisfy $n \equiv n' \bmod K_\varphi$ we have

$$(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})} \quad \textit{if and only if} \quad (q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})} \tag{1}$$

for each control location $q \in Q$.

Proof. We prove the theorem by induction on the structure of φ. That $t(\varphi)$ can be computed in polynomial time will be obvious.

Assume $\varphi \in \mathcal{P}$. Then we put $t(\varphi) = 0$. Recall that $K_\varphi = K^{\mathsf{lud}(\varphi)} = 1$. Trivially, (1) holds.

Assume $\varphi = \neg\psi$. Then we put $t(\varphi) = t(\psi)$. Note that $K_\varphi = K_\psi$. Equation (1) follows immediately by induction hypothesis.

Assume $\varphi = \psi_1 \wedge \psi_2$. Then we put $t(\varphi) = \max\{t(\psi_1), t(\psi_2)\}$. We have

$$\begin{aligned}
t(\varphi) &= \max\{t(\psi_1), t(\psi_2)\} \\
&\overset{\text{IH}}{\leq} \max\{2 \cdot |\psi_i| \cdot k^2 \cdot K_{\psi_i} \mid i \in \{1,2\}\} \\
&\leq 2 \cdot |\varphi| \cdot k^2 \cdot K_\varphi
\end{aligned}$$

and hence $t(\varphi)$ satisfies the requirement of the theorem. Note that $K_\varphi = \mathsf{LCM}\{K_{\psi_1}, K_{\psi_2}\}$ by definition. By choice of $t(\varphi)$, Equation (1) holds immediately due to induction hypothesis.

Assume $\varphi = \mathsf{EX}\psi$. Then we put $t(\varphi) = t(\psi) + K_\psi$. Thus we get

$$\begin{aligned}
t(\varphi) &= t(\psi) + K_\psi \\
&\overset{\text{IH}}{\leq} 2 \cdot |\psi| \cdot k^2 \cdot K_\psi + K_\psi \\
&\leq 2 \cdot (|\psi| + 1) \cdot k^2 \cdot K_\psi \\
&= 2 \cdot |\varphi| \cdot k^2 \cdot K_\varphi
\end{aligned}$$

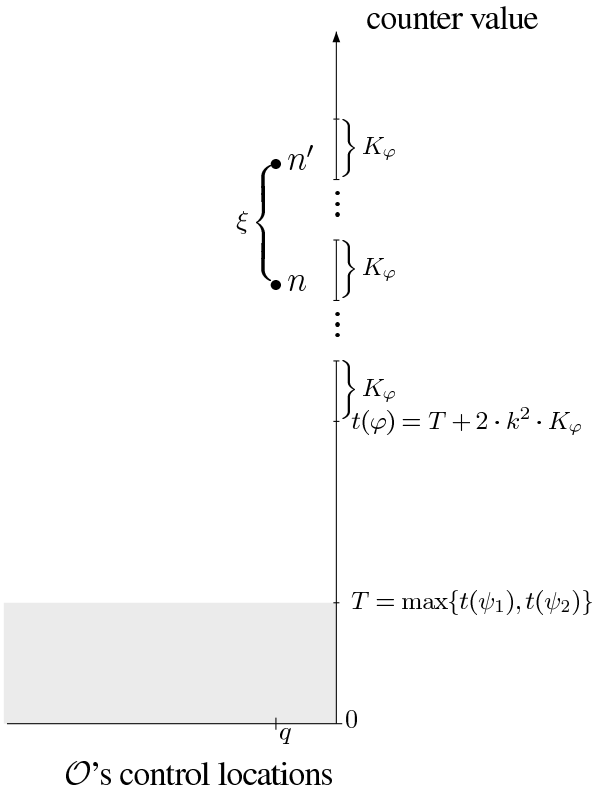


Fig. 1. The until case.

and hence $t(\varphi)$ satisfies the requirement of the theorem. Since $t(\varphi) - t(\psi) = K_\psi \geq 1$, we have that (1) follows immediately by induction hypothesis.

Assume $\varphi = \mathsf{E}\psi_1\mathsf{U}\psi_2$. Let us first define the threshold. Let $T = \max\{t(\psi_1), t(\psi_2)\}$. We put $t(\varphi) = T + 2 \cdot k^2 \cdot K_\varphi$. Hence we have

$$\begin{array}{rcl} t(\varphi) & = & T + 2 \cdot k^2 \cdot K_\varphi \\ & \overset{\text{IH}}{\leq} & \max\{2 \cdot |\psi_i| \cdot k^2 \cdot K_{\psi_i} \mid i \in \{1,2\}\} + 2 \cdot k^2 \cdot K_\varphi \\ & \leq & 2 \cdot ((|\varphi| - 1) + 1) \cdot k^2 \cdot K_\varphi \\ & = & 2 \cdot |\varphi| \cdot k^2 \cdot K_\varphi \end{array}$$

and thus $t(\varphi)$ satisfies the requirement of the theorem. It remains to prove (1).

Recall that $K_\varphi = \mathsf{LCM}\{K \cdot K_{\psi_1}, K_{\psi_2}\}$ by definition. Let us fix an arbitrary control location $q \in Q$ and naturals $n, n' \in \mathbb{N}$ such that $t(\varphi) < n < n'$ and $n \equiv n'$ mod K_φ. We have to prove that (1) holds, i.e., $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$ if and only if $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$. For this, let $\xi = n' - n$, which is a multiple of K_φ. The current situation is shown in Figure 1.

'Only-if': Let us assume that $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$. Hence, there exists a finite path

$$\pi \; = \; (q_1, n_1) \to (q_2, n_2) \; \cdots \to (q_l, n_l),$$

where $l \geq 1$, $\pi[1, l-1]$ is a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path, $(q, n) = (q_1, n_1)$, and $(q_l, n_l) \in [\![\psi_2]\!]_{\mathcal{T}(\mathcal{O})}$. Now we make a case distinction.

Case A: $n_j > T$ for each $j \in [l]$. Since $K_{\psi_1} | \xi$ and $K_{\psi_2} | \xi$ we obtain that the path $\pi + \xi$ witnesses $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$ by induction hypothesis. This is depicted in Figure 2.

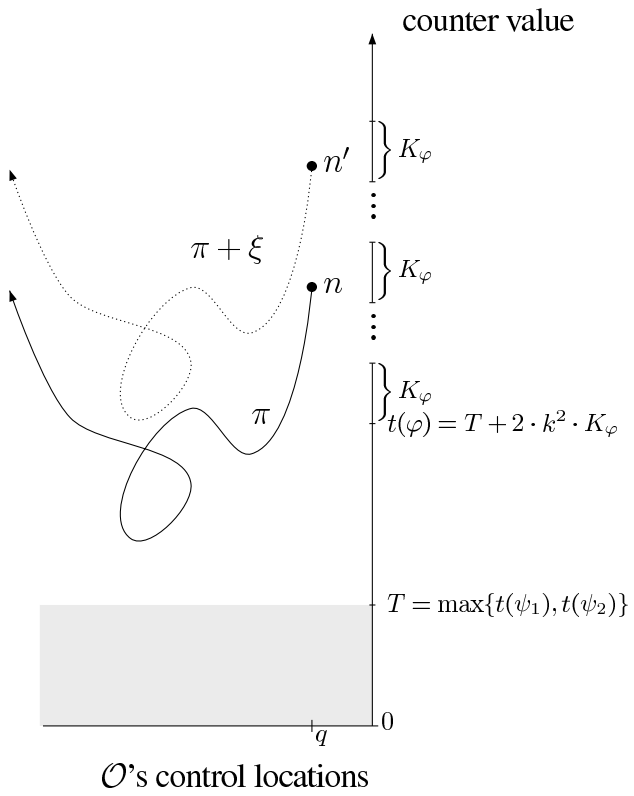


Fig. 2. The path $\pi + \xi$ witnesses $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$.

Case B: $n_j \leq T$ for some $j \in [l]$. For each of π's counter values $h \in \{n_i \mid i \in [l]\}$, define

$$\mu(h) \quad = \quad \min\{i \in [l] \mid n_i = h\}$$

to be the minimal position in π whose corresponding state has counter value h. We are interested in π's first states of counter value $n, n - K_{\psi_1}, n - 2 \cdot K_{\psi_1}$, and so on. For this, define $m(i) = \mu(n - i \cdot K_{\psi_1})$ for every appropriate $i \in \mathbb{N}$. By the pigeonhole principle, there are distinct $i_1, i_2 \in [0, k]$ such that $i_1 < i_2$ and $q_{m(i_1)} = q_{m(i_2)}$. Note that i_1 and i_2 are well-defined since

$$n - i_1 \cdot K_{\psi_1} \; > \; n - i_2 \cdot K_{\psi_1} \; \geq \; n - k \cdot K_{\psi_1} \; \geq \; T + 2 \cdot k^2 \cdot K_\varphi - k \cdot K_{\psi_1} \; > \; T.$$

Let $p = q_{m(i_1)} = q_{m(i_2)}$ and $d = i_1 - i_2 \in [k]$. Hence, d divides K. Moreover, let σ denote π's subpath from $(q_{m(i_1)}, n_{m(i_1)}) = (p, n - i_1 \cdot K_{\psi_1})$ down to $(q_{m(i_2)}, n_{m(i_2)}) = (p, n - i_2 \cdot K_{\psi_1}) = (p, n - i_1 \cdot K_{\psi_1} - d \cdot K_{\psi_1})$, i.e., formally $\sigma = \pi[m(i_1), m(i_2)]$. Note that σ is a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path. The current situation is depicted in Figure 3. The path σ is indicated thick.

We have to prove $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$. For this, we show that there exists a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path $\pi_\downarrow$ from (q, n') down to $(q_{m(i_1)}, n_{m(i_1)}) = (p, n - i_1 \cdot K_{\psi_1})$. Thus, since $\pi_\downarrow$ meets π in $(p, n - i_1 \cdot K_{\psi_1})$, it follows $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$. The path $\pi_\downarrow$ is indicated by a dashed curve in Figure 3. Our path $\pi_\downarrow$ consists of two concatenated paths. First recall that the path σ loses a counter height of precisely $d \cdot K_{\psi_1}$. The first part of $\pi_\downarrow$ is the $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path $\pi[1, m(i_1)]$ shifted upwards by the offset ξ (i.e., $\pi[1, m(i_1)] + \xi$). The second part of $\pi_\downarrow$ is the path from $(q_{m(i_1)}, n_{m(i_1)} + \xi) = (p, n - i_1 \cdot K_{\psi_1} + \xi)$ down to $(q_{m(i_1)}, n_{m(i_1)}) = (p, n - i_1 \cdot K_{\psi_1})$ that we can obtain by first shifting σ up by the offset ξ and then downward pumping it precisely $\frac{\xi}{d \cdot K_{\psi_1}}$ many times. Formally, this is the path

$$(\sigma + \xi)(\sigma + \xi - d \cdot K_{\psi_1})(\sigma + \xi - 2d \cdot K_{\psi_1}) \cdots (\sigma + d \cdot K_{\psi_1}) = \prod_{i=0}^{\frac{\xi}{d \cdot K_{\psi_1}} - 1} (\sigma + \xi - i \cdot d \cdot K_{\psi_1}).$$

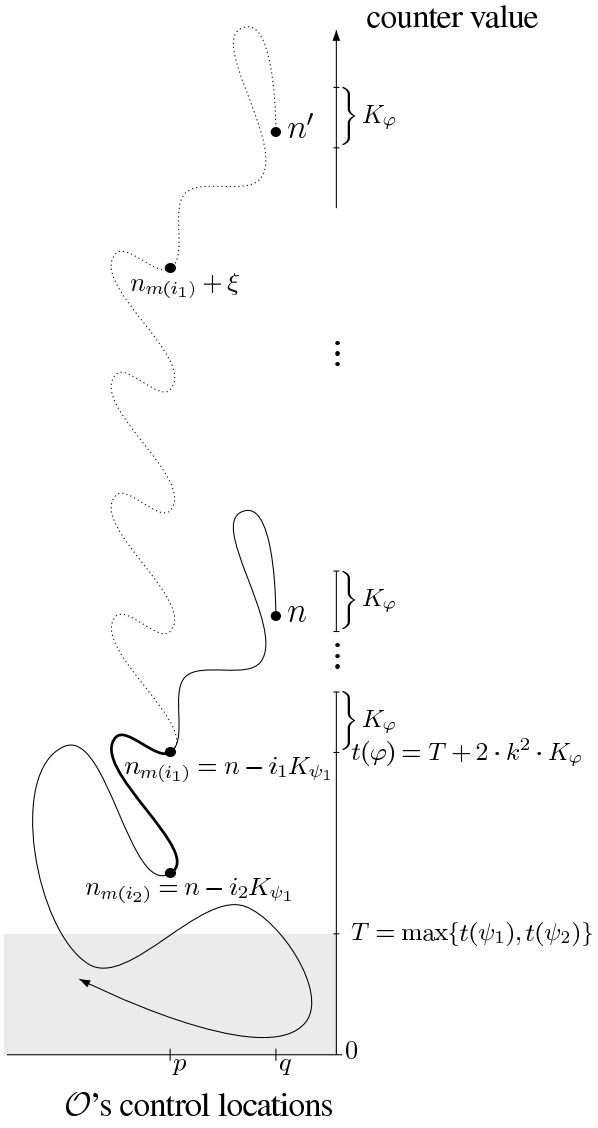


Fig. 3. The path from (q, n) can be merged from (q, n').

Note that ξ is a multiple of K_φ, which is in turn a multiple of $K \cdot K_{\psi_1}$, hence $\frac{\xi}{d \cdot K_{\psi_1}} \in \mathbb{N}$.

'If': Assume that $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$. To prove that $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$, we will use the following claim.

Claim: Assume some $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path $(q_1, n_1) \to (q_2, n_2) \to \cdots \to (q_l, n_l)$ whose counter values are all strictly above T and where $n_1 - n_l \geq k^2 \cdot K \cdot K_{\psi_1}$. Then there exists a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path from (q_1, n_1) to $(q_l, n_l + K \cdot K_{\psi_1})$ strictly above $T + K \cdot K_{\psi_1}$. The statement of the claim is depicted in Figure 4.

Thus, the claim tells us that paths that lose height at least $k^2 \cdot K \cdot K_{\psi_1}$ and whose states all have counter values strictly above T can be lifted by a height precisely $K \cdot K_{\psi_1}$.

Let us postpone the proof of the claim and first finish the proof of the if-direction. Since by assumption $(q, n') \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$, there exists a finite path

$$\pi \;=\; (q_1, n_1) \to (q_2, n_2) \cdots \to (q_l, n_l),$$

where $\pi[1, l-1]$ is a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path, $(q, n') = (q_1, n_1)$, and where $(q_l, n_l) \in [\![\psi_2]\!]_{\mathcal{T}(\mathcal{O})}$. To prove $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$, we make a case distinction.

Case A: $n_j > T$ for each $j \in [l]$. Assume that the path $\pi[1, l-1]$ contains two states whose counter difference is at least $k^2 \cdot K \cdot K_{\psi_1} + K_\varphi$ which is (strictly) greater than $k^2 \cdot K \cdot K_{\psi_1}$. Since K_φ is a multiple

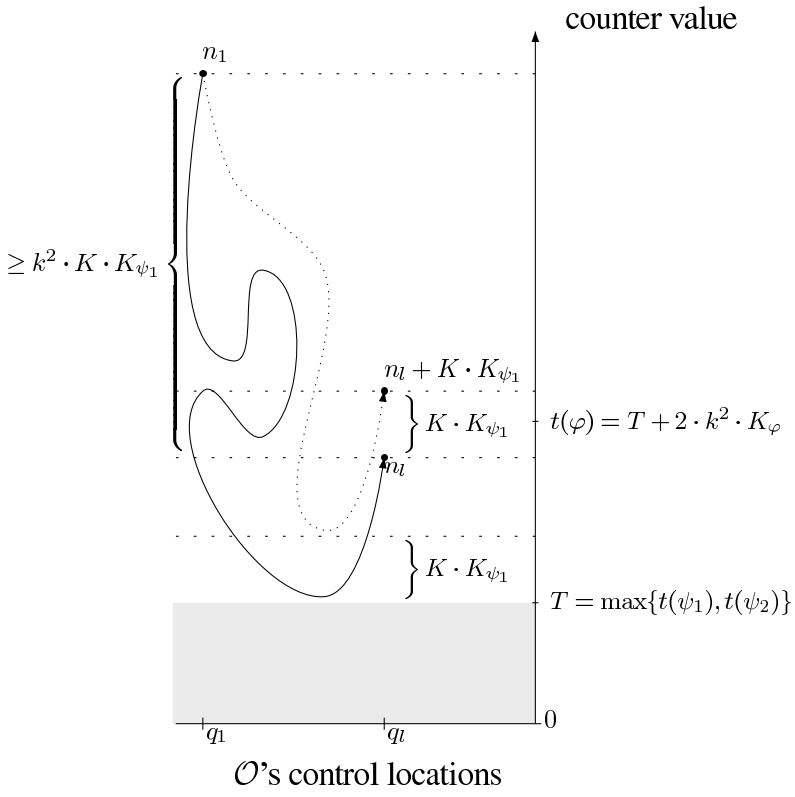


Fig. 4. Shortening paths above T of height difference at least $k^2 \cdot K \cdot K_{\psi_1}$ by height $K \cdot K_{\psi_1}$.

of $K \cdot K_{\psi_1}$ by definition, we can shorten $\pi[1, l-1]$ by a height precisely K_φ by applying the above claim $\frac{K_\varphi}{K \cdot K_{\psi_1}} \in \mathbb{N}$ many times. We repeat this shortening process of $\pi[1, l-1]$ by height K_φ as long as this is no longer possible, i.e., until there are no two states whose counter difference is at least $k^2 \cdot K \cdot K_{\psi_1} + K_\varphi$. Let σ denote the $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path starting in (q, n') that we obtain from $\pi[1, l-1]$ until the before mentioned shortening is no longer possible. Thus, σ ends in some state with a counter value that is congruent n_{l-1} modulo K_φ (since we shortened $\pi[1, l-1]$ by a multiple of K_φ). Since K_φ is in turn a multiple of K_{ψ_2}, we can build a path σ' which extends the path σ by a single transition to some state that satisfies ψ_2 by induction hypothesis. Moreover, by our shortening process, the counter difference between any two states in σ' is at most

$$k^2 \cdot K \cdot K_{\psi_1} + K_\varphi \quad \leq \quad 2 \cdot k^2 \cdot K_\varphi.$$

Since $n > T + 2 \cdot k^2 \cdot K_\varphi$, it follows that the path $\sigma' - \xi$ (which starts in (q, n)) is strictly above T. Moreover, since ξ is a multiple of K_{ψ_1} and K_{ψ_2}, this path witnesses $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$ by induction hypothesis.

Case B: $n_j = T$ for some $j \in [l]$. Let $j_0 \in [l]$ be minimal such that $n_{j_0} = T$. Note that $\pi[1, j_0 - 1]$ is a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path whose counter values are all strictly above T. Moreover, we have

$$n' - n_j = n' - T = (n' - n) + (n - T) = \xi + n - T > \xi + t(\varphi) - T = \xi + 2k^2 K_\varphi.$$

Hence, the maximal counter difference between two states of $\pi[1, j_0 - 1]$ is at least

$$2 \cdot k^2 \cdot K_\varphi + \xi \quad \geq \quad k^2 \cdot K \cdot K_{\psi_1} + \xi.$$

Hence, in analogy to case A, we can shorten $\pi[1, j_0 - 1]$ precisely by height ξ. Let σ denote the resulting path. Then $\sigma - \xi$ is a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path that ends in (q_{j_0-1}, n_{j_0-1}) and starts in (q, n). We can append $\pi[j_0 - 1, l]$ to this path. The resulting path witnesses $(q, n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$.

It remains to prove the above claim.

Proof of the claim. For each counter value $h \in \{n_i \mid i \in [l]\}$ that appears in π, let

$$\mu(h) = \min\{i \in [l] \mid n_i = h\}$$

denote the minimal position in π whose corresponding state has counter value h. Define $\Delta = k \cdot K_{\psi_1}$. We will be interested in $k \cdot K$ many consecutive intervals (of counter values) each of size Δ – we will call these intervals blocks. Define the bottom $b = n_1 - (k \cdot K) \cdot \Delta$. A *block* is an interval $B_i = [b + (i-1) \cdot \Delta, b + i \cdot \Delta]$ for some $i \in [k \cdot K]$. Since each block has size $\Delta = k \cdot K_{\psi_1}$, we can think of each block B_i to consist of k consecutive *subblocks* of size K_{ψ_1} each. Note that each subblock has two extremal elements, namely its *upper* and *lower boundary*. Thus all k subblocks have $k+1$ boundaries in total. Hence, by the pigeonhole principle, for each block B_i, there exists some distance $d_i \in [k]$ and two distinct boundaries $\beta(i,1)$ and $\beta(i,2)$ of distance $d_i \cdot K_{\psi_1}$ such that the control location of π's earliest state of counter value $\beta(i,1)$ agrees with the control location of π's earliest state of counter value $\beta(i,2)$, i.e., formally

$$q_{\mu(\beta(i,1))} \;=\; q_{\mu(\beta(i,2))}.$$

The situation is depicted in Figure 5. Observe that shortening the path π by gluing together π's states at position $\mu(\beta(i,1))$ and $\mu(\beta(i,2))$ still results in a $[\![\psi_1]\!]_{\mathcal{T}(\mathcal{O})}$-path by induction hypothesis, since we shorten the height of π by a multiple of K_{ψ_1}. Our overall goal is to shorten π by gluing together states only of certain blocks such that we obtain a path whose height is in total precisely $K \cdot K_{\psi_1}$ smaller than π's.

Recall that there are $k \cdot K$ many blocks. By the pigeonhole principle there is some $d \in [k]$ such that $d_i = d$ for at least K many blocks B_i. By gluing together $\frac{K}{d} \in \mathbb{N}$ pairs of states of distance $d \cdot K_{\psi_1}$ each, we shorten π by a height of $\frac{K}{d} \cdot d \cdot K_{\psi_1} = K \cdot K_{\psi_1}$. This proves the claim.

Assume $\varphi = \mathsf{E}\psi_1 \mathsf{W} \mathsf{U} \psi_2$. This can easily seen to be proven analogously to the case when $\varphi = \mathsf{E}\psi_1 \mathsf{U} \psi_2$. □

Theorem 2. *The following problem can be solved in time* $O(\log(n) + |Q|^3 \cdot |\varphi|^2 \cdot 4^{|Q| \cdot \mathsf{lud}(\varphi)} \cdot |\delta_0 \cup \delta_{>0}|)$*:*
INPUT: A one-counter automaton $\mathcal{O} = (Q, \{Q_p \mid p \in \mathcal{P}\}, \delta_0, \delta_{>0})$*, a* CTL *formula* φ*, a control location* $q \in Q$ *and some natural* $n \in \mathbb{N}$ *given in binary.*
QUESTION: $(q,n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$*?*

Proof. Let $k = |Q|$. We first compute the threshold $t(\varphi) \leq 2 \cdot |\varphi| \cdot k^2 \cdot K_\varphi$ from Theorem 1. Then we have $(q,n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$ if and only if $(q,m) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$, where either $n = m \leq t(\varphi)$ or $n > t(\varphi)$ and m is the unique number in the interval $[t(\varphi)+1, t(\varphi)+K_\varphi]$, which is congruent n modulo K_φ. We can find this number in time $O(\log n)$. Now we check $(q,m) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$ using the standard algorithm for model checking CTL on finite transition systems. The only difference is that if we reach a counter value of $t(\varphi) + K_\varphi + 1$, then we replace this value by $t(\varphi)+1$. More precisely, we compute inductively for every subformula ψ of φ the set

$$S(\psi) = [\![\psi]\!]_{\mathcal{T}(\mathcal{O})} \cap (Q \times [t(\varphi) + K_\varphi]).$$

Let us sketch the case of an until formula $\psi = \mathsf{E}\psi_1 \mathsf{U} \psi_2$. By induction, we have already computed the sets $S(\psi_1)$ and $S(\psi_2)$. The set $S(\psi)$ is computed by a fixpoint iteration. Initially, we put all elements from $S(\psi_2)$ into $S(\psi)$. Then, we perform the following fixpoint iteration process as long as possible. Assume that $(p,k) \in S(\psi_1)$ is a state, which does not belong to the current $S(\psi)$. Assume that (p,k) has a $\mathcal{T}(\mathcal{O})$-successor (where a counter value of $t(\varphi) + K_\varphi + 1$ is reduced to $t(\varphi)+1$) in $S(\psi)$. Then we add (p,k) to $S(\psi)$. The correctness of this fixpoint iteration process follows from Theorem 1. The size of each set $S(\psi)$ is bounded by $O(|Q| \cdot |\varphi| \cdot k^2 \cdot K_\varphi) \subseteq O(|Q|^3 \cdot |\varphi| \cdot 4^{|Q| \cdot \mathsf{lud}(\varphi)})$. Computing $S(\psi)$ can be done in time $O(|Q|^3 \cdot |\varphi| \cdot 4^{|Q| \cdot \mathsf{lud}(\varphi)} \cdot |\delta_0 \cup \delta_{>0}|)$. Hence, the total time bound is $O(\log(n) + |Q|^3 \cdot |\varphi|^2 \cdot 4^{|Q| \cdot \mathsf{lud}(\varphi)} \cdot |\delta_0 \cup \delta_{>0}|)$. □

The following corollary generalizes a result from [18], stating that the expression complexity of EF over one-counter automata is in P.

Corollary 3. *For every fixed one-counter automaton* $\mathcal{O} = (Q, \{Q_p \mid p \in \mathcal{P}\}, \delta_0, \delta_{>0})$ *and every fixed* k *the following problem is in* P*:*
INPUT: A CTL *formula* φ *with* $\mathsf{lud}(\varphi) \leq k$*, a control location* $q \in Q$ *and some natural* $n \in \mathbb{N}$ *given in binary.*
QUESTION: $(q,n) \in [\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$*?*

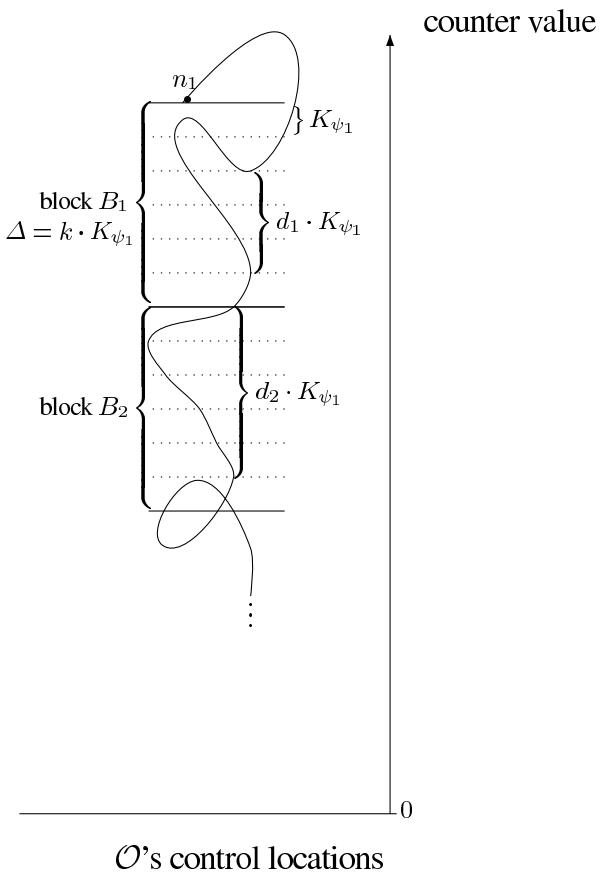


Fig. 5. Repeating control locations in blocks

5 Expression complexity for CTL is hard for PSPACE

The goal of this section is to prove that model checking CTL is PSPACE-hard already over a fixed zero-test-free one-counter automaton. We show this via a reduction from the well-known PSPACE-complete problem QBF. Our lower bound proof is separated into three steps. In step one, we define a family of CTL formulas $(\varphi_i)_{i\geq 1}$ such that over the fixed zero-test-free one-counter automaton $\mathcal{O}$ that is depicted in Figure 6 (states in $\mathcal{O}$ will be identified with atomic propositions) we can express (non-)divisibility by 2^i. In step two, we define a family of CTL formulas $(\psi_i)_{i\geq 1}$ such that over $\mathcal{O}$ we can express if the i^{th} bit in the binary representation of a natural number is set to 1. In our final step, we give the reduction from QBF.

For step one, we need the following simple fact which characterizes divisibility by powers of two. Recall that $[n] = \{1, \ldots, n\}$, in particular $[0] = \emptyset$.

Fact 4 *Let $n \geq 0$ and $i \geq 1$. Then the following two statements are equivalent:*

- *2^i divides n.*
- *2^{i-1} divides n and $|\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}|$ is even.*

The set of atomic propositions of $\mathcal{O}$ in Figure 6 coincides with its control locations. Note that both t and $\bar{t}$ are control locations of $\mathcal{O}$. Now we define a family of CTL formulas $(\varphi_i)_{i\geq 1}$ such that for each $n \in \mathbb{N}$ we have that

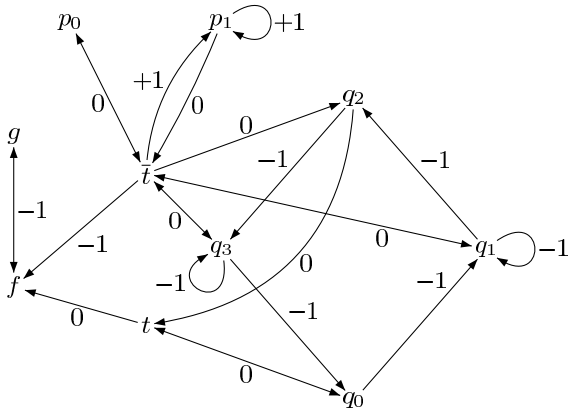


Fig. 6. The zero-test-free one-counter automaton $\mathcal{O}$ for which CTL model checking is PSPACE-hard

- $(t, n) \models \varphi_i$ if and only if 2^i divides n and
- $(\bar{t}, n) \models \varphi_i$ if and only if 2^i does *not* divide n.

On first sight, it might seem superfluous to let the control location t represent divisibility by powers of two and the control location $\bar{t}$ to represent non-divisibility by powers of two since CTL allows negation. However the fact that we have *only one* family of formulas $(\varphi_i)_{i \geq 1}$ to express both divisibility and non-divisibility is a crucial technical subtlety that is necessary in order to avoid an exponential blowup in formula size. By making use of Fact 4, we construct the formulas φ_i inductively. First, let us define the auxiliary formulas test $= t \vee \bar{t}$ and $\varphi_\diamond = q_0 \vee q_1 \vee q_2 \vee q_3$. Think of $\varphi_\diamond$ to hold in those control locations that altogether are situated in the "diamond" in Figure 6. We define

$$\varphi_1 \quad = \quad \text{test} \ \wedge \ \mathsf{EX}\,(f \wedge \mathsf{EF}(f \wedge \neg \mathsf{EX} g))\,.$$

Now assume $i > 1$. Then we define

$$\begin{aligned} \varphi_i &= \text{test} \ \wedge \ \mathsf{EX}\mu_i, \text{ where} \\ \mu_i &= \mathsf{E}(\varphi_\diamond \wedge \mathsf{EX}\varphi_{i-1})\mathsf{U}(q_0 \wedge \neg \mathsf{EX} q_1). \end{aligned}$$

Before we formally prove that φ_i indeed expresses (non-)divisibility in Lemma 6, let us provide some informal explanation. Observe that φ_i can only be true either in control location t or $\bar{t}$. Note that the formula right to the until symbol expresses that we are in q_0 and that the current counter value is zero. Also note that the formula left to the until symbol requires that $\varphi_\diamond$ holds, i.e., we are always in one of the four "diamond control locations". In other words, we decrement the counter by moving along the diamond control locations (by possibly looping) and always check if $\mathsf{EX}\varphi_{i-1}$ holds, just until we are in q_0 and the counter value is zero. Since φ_{i-1} is only used once in φ_i, we get:

Fact 5 $|\varphi_i| \in O(i)$.

The following lemma shows the correctness of the construction.

Lemma 6. *Let $n \geq 0$ and $i \geq 1$. Then*

(1) $(t, n) \models \varphi_i$ if and only if 2^i divides n.
(2) $(\bar{t}, n) \models \varphi_i$ if and only if 2^i does not divide n.

Proof. We prove statements (1) and (2) simultaneously by induction on i. For the induction base, assume $i = 1$. We only show (2), i.e. $(\bar{t}, n) \in [\![\varphi_1]\!]_{\mathcal{T}(\mathcal{O})}$ if and only if n is odd. We have the following equivalences:

$$\begin{aligned} (\bar{t}, n) \models \varphi_1 \quad &\iff \quad n \geq 1 \text{ and } (f, n-1) \models \mathsf{EF}(f \wedge \neg \mathsf{EX} g) \\ &\iff \quad n \geq 1 \text{ and } (f, n-1) \to^* (f, 0) \\ &\iff \quad n \geq 1 \text{ and } n - 1 \text{ is even} \\ &\iff \quad n \text{ is odd} \end{aligned}$$

Point (1) can be shown analogously for $i = 1$.

For the induction step, assume $i \geq 2$ and that the statement in the lemma holds for $i-1$. It is easy to verify by the construction of $\mathcal{O}$ and by induction hypothesis that the following claim holds.

Claim A: For every $n \geq 1$ the following equivalences hold:

$$\begin{array}{lcll} (q_0,n) \models \varphi_\diamond \wedge \mathsf{EX}\varphi_{i-1} & \iff & (q_2,n) \models \varphi_\diamond \wedge \mathsf{EX}\varphi_{i-1} & \iff & 2^{i-1} \text{ divides } n \\ (q_1,n) \models \varphi_\diamond \wedge \mathsf{EX}\varphi_{i-1} & \iff & (q_3,n) \models \varphi_\diamond \wedge \mathsf{EX}\varphi_{i-1} & \iff & 2^{i-1} \text{ does not divide } n \end{array}$$

Using Claim A, one can easily show the following (recall that $\mu_i = \mathsf{E}(\varphi_\diamond \wedge \mathsf{EX}\varphi_{i-1})\mathsf{U}(q_0 \wedge \neg\mathsf{EX}q_1)$):

Claim B: For every $n \geq 0$ the following equivalences hold:

$$\begin{array}{lcl} (q_0,n) \models \mu_i & \iff & 2^{i-1} \text{ divides } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is even} \\ (q_1,n) \models \mu_i & \iff & 2^{i-1} \text{ does not divide } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd} \\ (q_2,n) \models \mu_i & \iff & 2^{i-1} \text{ divides } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd} \\ (q_3,n) \models \mu_i & \iff & 2^{i-1} \text{ does not divide } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is even} \end{array}$$

Let us now prove Point (1) from the lemma for $i \geq 2$. We have the following equivalences:

$$\begin{array}{lcl} (t,n) \models \varphi_i & \iff & (q_0,n) \models \mu_i \\ & \overset{\text{Claim B}}{\iff} & 2^{i-1} \text{ divides } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is even} \\ & \overset{\text{Fact 4}}{\iff} & 2^i \text{ divides } n \end{array}$$

For Point (2), we have the following equivalences:

$$\begin{array}{lcl} (\bar{t},n) \models \varphi_i & \iff & \exists j \in \{1,2,3\} : (q_j,n) \models \mu_i \\ & \overset{\text{Claim B}}{\iff} & \text{either } 2^{i-1} \text{ does not divide } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd (i.e. } j = 1), \\ & & \text{or } 2^{i-1} \text{ does not divide } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is even (i.e. } j = 3), \\ & & \text{or } 2^{i-1} \text{ divides } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd (i.e. } j = 2) \\ & \iff & 2^{i-1} \text{ does not divide } n \text{ or } (2^{i-1} \text{ divides } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd}) \\ & \overset{\text{Fact 4}}{\iff} & 2^i \text{ does not divide } n \end{array}$$

□

For checking if the i^{th} bit of a natural number is set to 1, we make use of the following fact.

Fact 7 *Let $n \geq 0$ and $i \geq 1$. Then $\mathsf{bit}_i(n) = 1$ if and only if $|\{n' \in [n] \mid 2^{i-1}$ divides $n'\}|$ is odd.*

Proof. We have

$$\begin{array}{lcl} \mathsf{bit}_i(n) = 1 & \iff & n \bmod 2^i \in [2^{i-1}, 2^i - 1] \\ & \iff & \exists r \in [0, 2^{i-1} - 1], k \geq 0 : n = r + (2k+1) \cdot 2^{i-1} \\ & \iff & |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd.} \end{array}$$

□

Let us now define a family of CTL formulas $(\psi_i)_{i \geq 1}$ such that for each $n \in \mathbb{N}$ we have $\mathsf{bit}_i(n) = 1$ if and only if $(\bar{t},n) \models \psi_i$. We set

$$\begin{array}{lcll} \psi_1 & = & \varphi_1 & \text{and} \\ \psi_i & = & \bar{t} \wedge \mathsf{EX}\left((q_1 \vee q_2) \wedge \mu_i\right) & \text{for each } i > 1. \end{array}$$

Fact 5 and the construction of ψ_i immediately yield the following fact.

Fact 8 $|\psi_i| \in O(i)$.

The following lemma shows the correctness of the construction.

Lemma 9. *Let $n \geq 0$ and let $i \geq 1$. Then $(\bar{t}, n) \models \psi_i$ if and only if $\mathsf{bit}_i(n) = 1$.*

Proof. The case $i = 1$ is covered by Lemma 6. For $i \geq 2$, the following equivalences hold:

$$\begin{aligned}
(\bar{t}, n) \models \psi_i \quad &\iff \quad (q_1, n) \models \mu_i \text{ or } (q_2, n) \models \mu_i \\
&\overset{\text{Claim B}}{\iff} \quad \text{either } 2^{i-1} \text{ does not divide } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd} \\
&\qquad\quad \text{or } 2^{i-1} \text{ divides } n \text{ and } |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd} \\
&\iff \quad |\{n' \in [n] \mid 2^{i-1} \text{ divides } n'\}| \text{ is odd} \\
&\overset{\text{Fact 7}}{\iff} \quad \mathsf{bit}_i(n) = 1
\end{aligned}$$

□

For our final step, let us give a reduction from QBF. Let α be the following quantified boolean formula

$$\alpha \quad = \quad Q_k x_k \; Q_{k-1} x_{k-1} \; \cdots \; Q_1 x_1 \; \beta(x_1, \ldots, x_k),$$

where β is a boolean formula over variables $\{x_1, \ldots, x_k\}$ and $Q_i \in \{\exists, \forall\}$ is a quantifier for each $i \in [k]$. Our overall goal is to give a CTL formula θ such that our QBF formula α is valid if and only if $(\bar{t}, 0) \models \theta$. A truth assignment $\vartheta : \{x_1, \ldots, x_k\} \to \{0, 1\}$ corresponds to the natural number $n(\vartheta) \in [0, 2^k - 1]$, where for each $i \in [k]$, $\mathsf{bit}_i(n(\vartheta)) = 1$ if and only if $\vartheta(x_i) = 1$. First, let $\widehat{\beta}$ be the CTL formula that is obtained from the boolean formula β by replacing every occurrence of every variable x_i by ψ_i. Hence we obtain that for each $\vartheta : \{x_1, \ldots, x_k\} \to \{0, 1\}$ we have $\vartheta \models \beta$ if and only if $(\bar{t}, n(\vartheta)) \models \widehat{\beta}$ by Lemma 9.

It remains to define θ. Recall that θ will be evaluated in $(\bar{t}, 0)$. Let us parse our quantified boolean formula α from left to right. Setting the variable x_k to 1 will correspond to adding 2^{k-1} to the counter and getting to state $(\bar{t}, 2^{k-1})$. Setting x_k to 0 on the other hand will correspond to adding 0 to the counter and hence remaining in state $(\bar{t}, 0)$. Next, setting x_{k-1} to 1 corresponds to adding to the current counter value 2^{k-2}, whereas setting x_{k-1} to 0 corresponds to adding 0, as expected. Adding zero to the counter will be realized by the finite path that jumps from control location $\bar{t}$ to p_0 and then back to $\bar{t}$. Adding 2^{i-1} to the counter, on the other hand, will be realized by a finite path that jumps from control location $\bar{t}$ to p_1 (and thereby adds 1 to the counter), then loops at p_1 as long as the counter value is not divisible by 2^{i-1} (which can be ensured by checking if $(p_1, n) \models \mathsf{EX}(\bar{t} \wedge \varphi_{i-1})$ by Lemma 6) and finally jumps back to $\bar{t}$ when the counter value is divisible by 2^{i-1} for the first time again. We repeat this process until we have to set x_1 either to 1 or to 0. Eventually setting x_1 to 1 will correspond to go from $\bar{t}$ to p_1 (hence adding 1 to the counter) and then getting back to $\bar{t}$, whereas setting x_1 to 0 will correspond to go from $\bar{t}$ to p_0 and then back to $\bar{t}$. After that, we finally check if $\widehat{\beta}$ holds. Recall that $Q_k, \ldots, Q_1$ are the quantifiers of our quantified boolean formula α. For each $i \in [2, k]$, let us define formula θ_i as

$$\begin{aligned}
\theta_i \quad &= \quad R_i \mathsf{X} \left((p_0 \vee p_1) \bigcirc_i \mathsf{E} \left((p_0 \vee \mathsf{EX}(\bar{t} \wedge \varphi_{i-1})) \; \mathsf{U} \; (\bar{t} \wedge \neg \varphi_{i-1} \wedge \theta_{i-1}) \right) \right) \text{ and} \\
\theta_1 \quad &= \quad R_1 \mathsf{X} \left((p_0 \vee p_1) \bigcirc_1 \mathsf{EX}(\bar{t} \wedge \widehat{\beta}) \right)
\end{aligned}$$

with $\bigcirc_i = \wedge$ and $R_i = \mathsf{E}$ in case $Q_i = \exists$ and $\bigcirc_i = \rightarrow$ and $R_i = \mathsf{A}$ in case $Q_i = \forall$ for each $i \in [k]$. As expected, we put $\theta = \theta_k$. Observe that the size of θ is polynomial in the size of α and that θ can be computed in logarithmic space from α. We finally obtain the following easy equivalence.

Lemma 10. *The formula α is valid if and only if $(\bar{t}, 0) \in [\![\theta]\!]_{\mathcal{T}(\mathcal{O})}$.*

This finishes our PSPACE lower bound proof for expression complexity of CTL over zero-test-free one-counter automata. We have the following theorem.

Theorem 11. CTL *model checking of the fixed zero-test-free one-counter automaton $\mathcal{O}$ from Figure 6 is* PSPACE*-hard.*

Note that the formula θ in our reduction necessarily has a leftward until depth that depends on the size of α. By Corollary 3 this cannot be avoided unless P = PSPACE. Observe that in order to express divisibility by powers of two, our CTL formulas $(\varphi_i)_{i \geq 0}$ have a linearly growing leftward until depth.

6 Tools from complexity theory

For Sections 7–9 we need some concepts from complexity theory. The *i^{th} level* BH_i *of the boolean hierarchy* is defined as follows: $\mathsf{BH}_1 = \mathsf{NP}$, $\mathsf{BH}_{2i} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{BH}_{2i-1}, L_2 \in \mathsf{coNP}\}$, and $\mathsf{BH}_{2i+1} = \{L_1 \cup L_2 \mid L_1 \in \mathsf{BH}_{2i}, L_2 \in \mathsf{NP}\}$. The *boolean hierarchy* BH is defined as $\cup_{i \geq 1}\mathsf{BH}_i$. The class P^{NP} is the class of all problems that can be solved on a polynomially time bounded deterministic Turing machine with access to an oracle from NP. By $\mathsf{P}^{\mathsf{NP}[\log]}$ we denote the class of all problems that can be solved on a polynomially time bounded deterministic Turing machines which can have access to an NP-oracle only logarithmically many times. It is known that $\mathsf{BH} \subseteq \mathsf{P}^{\mathsf{NP}[\log]}$.

For naturals $m \geq 1$ and $0 \leq M \leq 2^m - 1$ let $\mathsf{BIN}_m(M) = \mathsf{bit}_1(M) \cdots \mathsf{bit}_m(M) \in \{0,1\}^m$ denote the m-bit binary representation of M. For proving a P^{NP} lower bound model checking EF over zero-test-free one-counter automata, we will need the following theorem.

Theorem 12 ([35]). *The following problem is complete for* P^{NP}*:*
INPUT: A boolean formula $\psi(x_1, \ldots, x_m)$?
QUESTION: Is ψ satisfiable and is the maximal number $M \in [0, 2^m - 1]$ with $\psi(\mathsf{BIN}_m(M)) = 1$ even (i.e. is the lexicographically maximal satisfying assignment even)?

6.1 Circuit complexity

More details on circuit complexity can be found in [34]. A boolean circuit $C = C(x_1, \ldots, x_n)$ is a directed acyclic graph (DAG) with the following properties (in the following, nodes of C are called *gates*, the in-degree (resp. out-degree) of a gate is called its *fan-in* (resp. *fan-out*)):

- The gates with fan-in 0 (they are called *input gates* in the following) are labeled with one of the symbols $x_1, \neg x_1, \ldots, x_n, \neg x_n$.
- Every gate with fan-in at least one is labeled with either AND or with OR.
- The gates of fan-out 0 (they are called *output gates* in the following) are linearly ordered, we denote this order by $o_1, \ldots, o_m$ in the following.

Such a circuit computes a function $f_C : \{0,1\}^n \to \{0,1\}^m$ in the obvious way. *Threshold circuits* may in addition to boolean circuits contain *majority gates*. Such a gate outputs 1 if and only if at least half of its input gates evaluate to 1. The *fan-in of a circuit* is the maximal fan-in of a gate in the circuit. The *size of a circuit* is the number of gates in the circuit. The *depth of a circuit* is the number of gates along a longest path from an input gate to an output gate. An AC^0*-circuit family* (resp. TC^0*-circuit family*) is a sequence $(C_n)_{n \geq 1}$ of boolean circuits (resp. threshold circuits) such that for some polynomial $p(n)$ and constant c:

- the size of C_n is at most $p(n)$,
- the depth of C_n is at most c, and
- for each $k \geq 0$ there is at most one circuit in $(C_n)_{n \geq 1}$ with exactly k input gates.

An NC^1*-circuit family* is a sequence $(C_n)_{n \geq 1}$ of boolean circuits such that for some polynomial $p(n)$ and constant c:

- the size of C_n is at most $p(n)$,
- the depth of C_n is at most $c \cdot \log n$,
- the fan-in of C_n is at most 2, and
- for each $k \geq 0$ there is at most one circuit in $(C_n)_{n \geq 1}$ with exactly k input gates.

Circuit families of these types compute partial mappings on $\{0,1\}^*$ in the obvious way.[4] Finally, a circuit family $(C_n)_{n \geq 0}$ is called *logspace-uniform* if there exists a logspace transducer that computes on input 1^n a representation (e.g. as a node-labeled DAG) of the circuit C_n. In the literature on circuit complexity one can find more restrictive notions of uniformity, see e.g. [34], but logspace uniformity suffices for our

[4] Note that we do not require to have for every $n \geq 0$ a circuit with exactly n input gates in the family, therefore the computed mapping is in general only partially defined.

purposes. In fact, polynomial time uniformity suffices for proving our lower bounds w.r.t. polynomial time reductions. We recall that $\mathsf{AC}^0 \subseteq \mathsf{NC}^1$.

For our lower bound on the data complexity of CTL, we use a deep result from [11, 22]. First, we need a few definitions. Let p_i denote the i^{th} prime number. It is well-known that p_i is polynomially bounded in i; hence the i^{th} prime requires $O(\log i)$ bits in its binary representation. Moreover, we need the following proposition, see e.g. [11]:

Proposition 13. *A list $(p_1, \ldots, p_m)$ of the first m prime numbers in unary notation can be computed in space $O(\log m)$.*

For a number $0 \leq M < \prod_{i=1}^{m} p_i$ we define the *Chinese remainder representation* $\mathsf{CRR}_m(M)$ as the boolean tuple

$$\mathsf{CRR}_m(M) = (x_{i,r})_{i\in[m],0\leq r<p_i} \quad \text{with } x_{i,r} = \begin{cases} 1 & \text{if } M \bmod p_i = r \\ 0 & \text{else} \end{cases}$$

By the following theorem, we can transform a CRR-representation very efficiently into binary representation.

Theorem 14 ([11, Thm. 3.3]). *There is a logspace-uniform NC^1-circuit family $(B_m((x_{i,r})_{i\in[m],0\leq r<p_i}))_{m\geq 1}$ such that for every $m \geq 1$, B_m has m output gates and*

$$\forall\, 0 \leq M < \prod_{i=1}^{m} p_i : B_m(\mathsf{CRR}_m(M)) = \mathsf{BIN}_m(M \bmod 2^m).$$

By [22], we could replace logspace-uniform NC^1-circuits in Theorem 14 even by DLOGTIME-uniform TC^0-circuits. The existence of a P-uniform NC^1-circuit family for converting from CRR-representation to binary representation was already shown in [5].

Usually the Chinese remainder representation of M is the tuple $(r_i)_{i\in[m]}$, where $r_i = M \bmod p_i$. Since the primes p_i will be always given in unary notation, there is no essential difference between this representation and our Chinese remainder representation. The latter is more suitable for our purpose.

6.2 Serializability

Intuitively, a complexity class $\mathcal{C}_1$ is called $\mathcal{C}_2$-serializable (where $\mathcal{C}_2$ is another complexity class) if every language $L \in \mathcal{C}_1$ can be accepted in the following way: There exists a polynomial $p(n)$ and a $\mathcal{C}_2$-machine (or $\mathcal{C}_2$-circuit family) A such that $x \in L$ is checked in $2^{p(|x|)}$ many stages, which are indexed by the strings from $\{0,1\}^{p(|x|)}$. In stage $y \in \{0,1\}^{p(|x|)}$, A gets from the stage indexed by the lexicographic predecessor of y a constant number of bits $b_1, \ldots, b_c$ and computes from these bits, the index y and the original input x new bits $b'_1, \ldots, b'_c$ which are delivered to the lexicographic next stage. In [10] it was shown that PSPACE is P-serializable; in [21] this result was sharpened to AC^0-serializability, see also [33]. It is not stated in [21, 33] but easy to see from the proofs that *logspace-uniform* AC^0 suffices for serializing PSPACE, see the appendix for more details.

For our purpose, a slightly different definition of AC^0-serializability is useful: A language L is AC^0-serializable if there exists an nondeterministic finite automaton A over the alphabet $\{0,1\}$, a polynomial $p(n)$, and a logspace-uniform AC^0-circuit family $(C_n)_{n\geq 0}$, where C_n has exactly $n + p(n)$ many inputs and one output, such that for every $x \in \{0,1\}^n$ we have:

$$x \in L \iff \prod_{y\in\{0,1\}^{p(n)}} C_n(x,y) \in L(A),$$

where "$\ldots$" refers to the lexicographic order on $\{0,1\}^{p(n)}$ and for every $y \in \{0,1\}^{p(n)}$, $C_n(x,y)$ is either 0 or 1 (hence, $\prod_{y\in\{0,1\}^{p(n)}} C_n(x,y)$ is a binary string of length $2^{p(n)}$). This definition of AC^0-serializability is equivalent to the one in [21]. A proof that every language in PSPACE is AC^0-serializable in the above sense can be found in the appendix.

6.3 Combining serializability and small-depth circuits

In this section, we will combine Theorem 14 with the AC^0-serializability of PSPACE to get a new characterization of PSPACE that will be suitable for our lower bound proofs in the rest of the paper. Again, let p_i denote the i^{th} prime number. Note that $\prod_{i=1}^{m} p_i \geq 2^m$ for all $m \geq 1$. Also note that a boolean formula can be seen as a boolean circuit, where every gate that is neither an input gate nor the output gate has fan-out 1 (i.e., is input gate for exactly one other gate). We only consider boolean formulas, where AND and OR are binary, i.e, every AND-expression and every OR-expression has exactly two arguments. In this case, we write $\wedge$ (resp., $\vee$) for AND (resp. OR).

Proposition 15. *For every language $L \subseteq \{0,1\}^*$ from* PSPACE *there exists a polynomial $p(n)$ and an NFA A over the alphabet $\{0,1\}$ such that the following holds: From a given input $x \in \{0,1\}^*$ with $|x| = n$ one can construct in logspace a boolean formula F with propositional variables $x_{i,r}$ ($i \in [p(n)]$ and $0 \leq r < p_i$) such that:*

$$x \in L \iff \prod_{M=0}^{2^m-1} F(\mathsf{CRR}_m(M)) \in L(A). \tag{2}$$

Proof. Let us fix a language $L \subseteq \{0,1\}^*$ from PSPACE. Recall from Section 6.2 that PSPACE is AC^0-serializable [21] and hence NC^1-serializable. Thus, there exists an NFA A over the alphabet $\{0,1\}$, a polynomial $p(n)$, and a logspace-uniform NC^1-circuit family $(C_n)_{n\geq 0}$, where C_n has $n + p(n)$ many inputs, such that for every $x \in \{0,1\}^n$ we have

$$x \in L \iff \prod_{y \in \{0,1\}^{p(n)}} C_n(x,y) \in L(A), \tag{3}$$

where the order in the concatenation $\prod$ is the lexicographic order on $\{0,1\}^{p(n)}$. Fix an input $x \in \{0,1\}^n$. Our construction of the boolean formula F can be split into three steps:

Step 1. Construct in space $O(\log n)$ the circuit C_n. Fix the first n inputs of C_n to the bits in x, and denote the resulting circuit by C; it has only $m := p(n)$ many inputs. Equivalence (3) can be written as

$$x \in L \iff \prod_{M=0}^{2^m-1} C(\mathsf{BIN}_m(M)) \in L(A). \tag{4}$$

Step 2. Compute in space $O(\log m) = O(\log n)$ the circuit $B = B_m((x_{i,r})_{i\in[m],0\leq r<p_i})$ from Theorem 14. Thus, B is a boolean circuit of fan-in 2 and depth $O(\log m) = O(\log n)$ with

$$B(\mathsf{CRR}_m(M)) = \mathsf{BIN}_m(M \bmod 2^m)$$

for every $0 \leq M < \prod_{i=1}^{m} p_i$.

Step 3. Now we compose the circuits B and C: For every $i \in [m]$, connect the i^{th} input of the circuit $C(x_1, \ldots, x_m)$ with the i^{th} output of the circuit B. The result is a circuit with fan-in 2 and depth $O(\log n)$. We can unfold this circuit into a boolean formula $F = F((x_{i,r})_{i\in[m],0\leq r<p_i})$. The resulting formula (or tree) has the same depth as the circuit, i.e., depth $O(\log n)$ and every tree node has at most 2 children. Hence, F has polynomial size. For every $0 \leq M < 2^m$ we have $F(\mathsf{CRR}_m(M)) = C(\mathsf{BIN}_m(M))$ and equivalence (4) can be written as

$$x \in L \iff \prod_{M=0}^{2^m-1} F(\mathsf{CRR}_m(M)) \in L(A).$$

The unfolding of the circuit can be done in space $O(\log n)$, since the circuit has depth $O(\log n)$ and fan-in 2 (this implies that a path in the circuit from the output gate down to a certain node can be stored in space $O(\log n)$). This proves the proposition. □

Proposition 15 can be used for PSPACE lower bound proofs for OCA. The idea is to store the number M in (2) on the counter. To check whether $F(\mathsf{CRR}_m(M))$ evaluates to true, the OCA traverses the boolean formula F. Each time, a variable $x_{i,r}$ is reached, it has to be checked whether the current counter value is congruent r modulo p_i. An OCA can do this. Recall that p_i can be constructed in space $O(\log i)$. Hence, we can also construct in space $O(\log i)$ an OCA that checks whether the current counter value is congruent r modulo p_i. When doing this modulo check, the original counter value is of course lost. In the context of CTL, which we discuss in the next section, this is not a problem, since a fixed CTL-formula can control a second computation path (on which the modulo test is done) that branches off the main computation path (which traverses the boolean formula).

For our PSPACE lower bound proof for timed automata in Section 10.1 we need the following variant of Proposition 15. The proof is the same as for Proposition 15, we just skip Step 2, i.e., the use of small depth circuits for transforming CRR-representations into binary representations.

Proposition 16. *For every language $L \subseteq \{0,1\}^*$ from PSPACE there exists a polynomial $p(n)$ and an NFA A over the alphabet $\{0,1\}$ such that the following holds: From a given input $x \in \{0,1\}^*$ with $|x| = n$ one can construct in logspace a boolean formula F with propositional variables x_i ($i \in [p(n)]$) such that:*

$$x \in L \iff \prod_{M=0}^{2^m-1} F(\mathsf{BIN}_m(M)) \in L(A).$$

7 Data complexity for CTL is hard for PSPACE

In this section, we prove that also the data complexity of CTL over zero-test-free one-counter automata is hard for PSPACE and therefore PSPACE-complete by the known upper bounds for the modal μ-calculus [30]. We will use Proposition 15 for this. Let us fix the set of propositions $\mathcal{P} = \{\alpha, \beta, \gamma\}$ for this section. In the following, we allow in the transition relation δ of a zero-test-free OCA transitions of the kind (q, k, q'), where $k \in \mathbb{Z}$ is given in unary representation with the expected intuitive meaning. Clearly, such transitions can be eliminated with a logspace transformation.

The following proposition formalizes the idea explained after the proof of Propositon 15.

Proposition 17. *For the fixed* EF *formula $\varphi = (\alpha \to \mathsf{EX}(\beta \wedge \mathsf{EF}(\neg\mathsf{EX}\gamma)))$ the following problem can be solved with a logspace transducer:*
INPUT: A list of the first m consecutive (unary encoded) prime numbers $p_1, \ldots, p_m$ and a boolean formula $F = F((x_{i,r})_{i\in[m],0\le r<p_i})$
OUTPUT: A zero-test-free OCA $\mathcal{O}_F$ with distinguished control locations in *and* out *such that for every number $0 \le M < \prod_{i=1}^{m} p_i$ the following are equivalent:*

- $F(\mathsf{CRR}_m(M)) = 1$
- *There exists a $[\![\varphi]\!]_{\mathcal{T}(\mathcal{O}_F)}$-path from (in, M) to (out, M) in $\mathcal{T}(\mathcal{O}_F)$.*

Proof. We first transform in logspace the input formula F into a *positive boolean formula* (i.e., a boolean formula that is built up from propositional variables and the binary operators $\wedge$ and $\vee$). For this we first push negations down so that they only occur in front of propositional variables $x_{i,r}$. Then, a negated variable $\neg x_{i,r}$ can be replaced by the disjunction $\bigvee\{x_{i,k} \mid 0 \le k < p_i, r \ne k\}$. Note that this can be done in logspace, since the primes p_i are given in unary representation. The $\bigvee$ can be replaced by binary $\vee$'s. We denote the resulting formula with F again.

Now, the idea is to traverse the positive boolean formula F with the zero-test-free OCA $\mathcal{O}_F$ in a depth first manner. Each time a variable $x_{i,r}$ is seen, the zero-test-free OCA may also enter another branch, where it is checked, whether the current counter value is congruent r modulo p_i. Let S_F be the syntax tree of the formula F, which is a binary, rooted, and node-labeled tree. Each node of S_F is labeled with one of the symbols $\wedge$, $\vee$, or $x_{i,r}$ ($1 \le i \le m$, $0 \le r < p_i$). Set V_F be the set of nodes of the tree S_F. Then $\mathcal{O}_F$ is

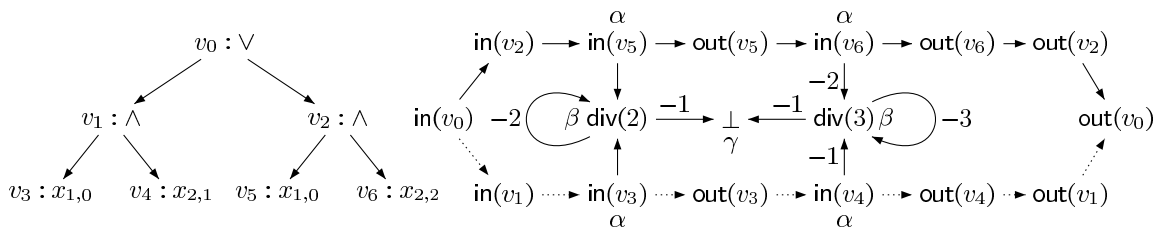


Fig. 7. The syntax tree S_F and the OCA $\mathcal{O}_F$ for the formula $F = (x_{1,0} \wedge x_{2,1}) \vee (x_{1,0} \wedge x_{2,0})$. Unlabeled transitions are implicitly labeled with 0.

defined as

$$
\begin{aligned}
\mathcal{O}_F &= (Q, \{Q_\alpha, Q_\beta, Q_\gamma\}, \delta), \text{ where} \\
Q &= \{\mathsf{in}(v), \mathsf{out}(v) \mid v \in V_F\} \cup \{\mathsf{div}(p_1), \ldots, \mathsf{div}(p_m), \perp\} \\
Q_\alpha &= \{\mathsf{in}(v) \mid v \in V_F \text{ is labeled with a variable } x_{i,r} (i \in [m], 0 \leq r < p_i)\} \\
Q_\beta &= \{\mathsf{div}(p_1), \ldots, \mathsf{div}(p_m)\} \\
Q_\gamma &= \{\perp\}.
\end{aligned}
$$

For $m = 2$ (and hence the primes $p_1 = 2$ and $p_2 = 3$) and the formula $F = (x_{1,0} \wedge x_{2,1}) \vee (x_{1,0} \wedge x_{2,2})$ the construction is shown in Figure 7. As an example $M = 4$ corresponds to the residue class 0 modulo $p_1 = 2$ and the residue class 1 modulo $p_2 = 3$, thus M assigns to the variables $x_{1,0}$ and $x_{2,1}$ the truth value 1 and to the other variables the truth value 0. A corresponding $[\![\varphi]\!]_{\mathcal{T}(\mathcal{O}_F)}$-path is displayed with dashed lines in Figure 7. We set $\mathsf{in} = \mathsf{in}(v_0)$ and $\mathsf{out} = \mathsf{out}(v_0)$, where v_0 is the root of S_F. Let us now define the transition set δ. Let v be a node of S_F, which is not a leaf, and let v_1 and v_2 be the children of v. If v is labeled with $\vee$, then we add the transitions

$$(\mathsf{in}(v), 0, \mathsf{in}(v_i)) \text{ and } (\mathsf{out}(v_i), 0, \mathsf{out}(v)) \text{ for } i \in \{1, 2\}$$

to δ. If v is labeled with $\wedge$, we add the transitions

$$(\mathsf{in}(v), 0, \mathsf{in}(v_1)), (\mathsf{out}(v_1), 0, \mathsf{in}(v_2)), \text{ and } (\mathsf{out}(v_2), 0, \mathsf{out}(v))$$

to δ. If v is a leaf of S_F that is labeled with the variable $x_{i,r}$, we add the transitions

$$(\mathsf{in}(v), 0, \mathsf{out}(v)) \text{ and } (\mathsf{in}(v), -r, \mathsf{div}(p_i))$$

to δ. For the control locations $\mathsf{div}(p_i)$ we add to δ the transitions

$$(\mathsf{div}(p_i), -p_i, \mathsf{div}(p_i)) \text{ and } (\mathsf{div}(p_i), -1, \perp).$$

This concludes the description of the zero-test-free OCA $\mathcal{O}_F$. Correctness of the construction can be easily checked by induction on the structure of the formula F. □

We are now ready to prove PSPACE-hardness of the data complexity.

Theorem 18. *There exists a fixed* CTL *formula of the form* $\mathsf{E}\varphi_1\mathsf{U}\varphi_2$*, where* φ_1 *and* φ_2 *are* EF *formulas, such that the following problem is* PSPACE*-complete:*
INPUT: A zero-test-free OCA $\mathcal{O}$ *and a control location* q *of* $\mathcal{O}$*.*
QUESTION: $(\mathcal{T}(\mathcal{O}), (q, 0)) \models \mathsf{E}\varphi_1\mathsf{U}\varphi_2$*?*

Proof. The PSPACE upper bound is already known and even holds for the combined complexity of model checking OCA against formulas of the modal μ-calculus as shown in [30]. It thus remains to prove PSPACE-hardness. For this, let us take an arbitrary PSPACE-complete language $L \subseteq \{0,1\}^*$. Fix the polynomial $p(n)$ and the NFA $A = (S, \{0,1\}, \delta, s_0, S_f)$ over the alphabet $\{0,1\}$ from Proposition 15. Let $x \in \{0,1\}^*$ be an input of length n, and let $m := p(n)$, where w.l.o.g. $m > 1$. By Proposition 15 one can

construct in space $\log n$ a boolean formula with propositional variables $x_{i,r}$ ($i \in [m]$ and $0 \leq r < p_i$) such that:

$$x \in L \iff \prod_{M=0}^{2^m-1} F(\mathsf{CRR}_m(M)) \in L(A). \tag{5}$$

By Proposition 13 we can compute in space $O(\log m) = O(\log n)$ a list $p_1, \ldots, p_m$ of the first m prime numbers in unary notation. Note that $\prod_{i=1}^{m} p_i > 2^m$ since $m > 1$.

We now apply our construction from Proposition 17 to the formula F. More precisely, let G be the boolean formula $\bigwedge_{i\in[m]} x_{i,r_i}$ were $r_i = 2^m \bmod p_i$ for $i \in [m]$ (these remainders can be computed in logarithmic space). For every 1-labeled transition $\tau \in \delta$ of the NFA A let $\mathcal{O}(\tau)$ be a copy of the zero-test-free OCA $\mathcal{O}(F \wedge \neg G)$. For every 0-labeled transition $\tau \in \delta$ let $\mathcal{O}_\tau$ be a copy of the zero-test-free OCA $\mathcal{O}_{\neg F \wedge \neg G}$. In both cases we write $\mathcal{O}_\tau$ as $(Q(\tau), \{Q_\alpha(\tau), Q_\beta(\tau), Q_\gamma(\tau)\}, \delta(\tau))$. Denote with $\mathsf{in}(\tau)$ (resp. $\mathsf{out}(\tau)$) the control location of this copy that corresponds to in (resp. out) in $\mathcal{O}_F$. Hence, for every b-labeled transition $\tau \in \delta$ ($b \in \{0,1\}$) and every $0 \leq M < \prod_{i=1}^{m} p_i$ there exists a $[\![\varphi]\!]_{\mathcal{T}(\mathcal{O}_\tau)}$-path ($\varphi$ is from Proposition 17) from $(\mathsf{in}(\tau), M)$ to $(\mathsf{out}(\tau), M)$ if and only if $F(\mathsf{CRR}_m(M)) = b$ and $M \neq 2^m$.

We now define a zero-test-free OCA $\mathcal{O} = (Q, \{Q_\alpha, Q_\beta, Q_\gamma\}, \delta')$ as follows: We take the disjoint union of all the OCA $\mathcal{O}_\tau$ for $\tau \in \delta$. Moreover, every state $s \in S$ of the automaton A becomes a control location of $\mathcal{O}$:

$$Q = S \cup \bigcup_{\tau\in\delta} Q(\tau)$$
$$Q_p = \bigcup_{\tau\in\delta} Q_p(\tau) \text{ for } p \in \{\alpha, \beta, \gamma\}$$

We add to δ' for every NFA-transition $\tau = (s, b, t) \in \delta$ the following transitions:

$$(s, 0, \mathsf{in}(\tau)),\ (\mathsf{out}(\tau), +1, t).$$

Then, by Proposition 17 and (5) we have $x \in L$ if and only if there exists a $[\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$-path in $\mathcal{T}(\mathcal{O})$ from $(s_0, 0)$ to $(s, 2^m)$ for some $s \in S_f$. Also note that there is no $[\![\varphi]\!]_{\mathcal{T}(\mathcal{O})}$-path in $\mathcal{T}(\mathcal{O})$ from $(s_0, 0)$ to some configuration (s, M) with $s \in S$ and $M > 2^m$. It remains to add to $\mathcal{O}$ some structure that enables $\mathcal{O}$ to check that the counter has reached the value 2^m.

For this, use Proposition 17 to construct the zero-test-free OCA $\mathcal{O}_G$ (where G is from above) and add it disjointly to $\mathcal{O}$. Moreover, add to δ' the transitions $(s, 0, \mathsf{in})$ for all $s \in S_f$, where in is the in control location of $\mathcal{O}_G$. Finally, introduce a new proposition ρ and set $Q_\rho = \{\mathsf{out}\}$, where out is the out control location of $\mathcal{O}_G$. By putting $q = s_0$ we obtain:

$$x \in L \iff (\mathcal{T}(\mathcal{O}), (q, 0)) \models \mathsf{E}\ \underbrace{(\alpha \to \mathsf{EX}(\beta \wedge \mathsf{EF}(\neg\mathsf{EX}\gamma)))}_{\varphi \text{ from Proposition 17}}\ \mathsf{U}\ \rho.$$

This concludes the proof of the theorem. □

By slightly modifying the proof of Theorem 18, the following corollary can be shown.

Corollary 19. *There exists a fixed* CTL *formula of the form* $\mathsf{EG}\psi$*, where* ψ *is an* EF *formula, such that the following problem is* PSPACE*-complete:*
INPUT: A zero-test-free OCA $\mathcal{O}$ *and a control location* q *of* $\mathcal{O}$*.*
QUESTION: $(\mathcal{T}(\mathcal{O}), (q, 0)) \models \mathsf{EG}\psi$*?*

Proof. The proof is almost identical to the proof of Theorem 18, except that we do not introduce the atomic proposition ρ. We rather add to δ' the transition $(\mathsf{out}, 0, \mathsf{in})$, where out is the out control location of $\mathcal{O}_G$ and in is the in control location of $\mathcal{O}_G$. We define $\psi = \varphi$, where again φ is the formula from Proposition 17. □

8 Combined complexity of EF is hard for P^{NP}

In this section, we will apply the efficient transformation from Chinese remainder representation to binary representation (Theorem 14) in order to prove that the combined complexity for EF over zero-test-free one-counter automata is hard for P^{NP}. For formulas represented succinctly by DAGs (directed acyclic graphs) this was already shown in [18]. The point here is that we use the standard tree representation for formulas. The following proposition states that evaluating a boolean formula whose variables are given in Chinese remainder representation can in fact be reduced to EF model checking an appropriate OCA whose counter value we assume to encode this Chinese remainder representation.

Proposition 20. *The following problem can be solved by a logspace transducer:*
INPUT: A list of the first m consecutive (unary encoded) prime numbers and a boolean circuit $C = C((x_{i,r})_{i\in[m],0\leq r<p_i})$ (with a single output gate)
OUTPUT: A zero-test-free OCA $\mathcal{O}_C$ with a distinguished state in *and an* EF *formula $\varphi(C)$ such that for every number $0 \leq M < \prod_{i=1}^{m} p_i$ we have:*

$$C(\mathsf{CRR}_m(M)) = 1 \quad \Longleftrightarrow \quad (\mathcal{T}(\mathcal{O}_C), (\mathsf{in}, M)) \models \varphi(C).$$

Proof. As in the proof of Proposition 17 we can eliminate in C negated input gates $\neg x_{i,r}$ by disjunctions of positive input gates. Moreover, we can w.l.o.g. assume that the circuit C is organized in $k+1$ layers, where each layer either contains only AND- or OR-gates. All children of a node in layer i belong to layer $i+1$. Layer 1 contains only the unique output gate of the circuit, whereas layer $k+1$ contains the input gates. For $i \in [k]$, let $\ell_i = \mathsf{AND}$ (resp. $\ell_i = \mathsf{OR}$) if layer i consists of AND-gates (resp. OR-gates).

The set of control locations of the zero-test-free OCA $\mathcal{O}_C$ contains all gates of the circuit C; the unique output gate becomes the distinguished state in. We add the transition $(g_1, 0, g_2)$ to $\mathcal{O}_C$ if gate g_2 is a child of gate g_1. If gate g is an input gate labeled with $x_{i,r}$ then we add the transition $(g, -r, \mathsf{div}(p_i))$ to $\mathcal{O}_C$. Finally, for the states $\mathsf{div}(p_i)$ we have the same transitions as in the proof of Proposition 17. This concludes the description of the zero-test-free OCA $\mathcal{O}_C$.

In order to describe the EF formula $\varphi(C)$ let $\mathsf{M}_i = \mathsf{EX}$ (resp. $\mathsf{M}_i = \mathsf{AX}$) if $\ell_i = \mathsf{OR}$ (resp. $\ell_i = \mathsf{AND}$) for $i \in [k]$. Then let

$$\varphi(C) = \mathsf{M}_1\mathsf{M}_2 \cdots \mathsf{M}_k\mathsf{EX}\,\mathsf{EF}(\neg\mathsf{EX}\gamma), \tag{6}$$

where the proposition γ is used in the same way as in the proof of Proposition 17 to allow to test if the counter value is zero. It is clear that this formula fulfills the requirements of the theorem. □

We now prove that model checking EF on zero-test-free OCA is hard for P^{NP}.

Theorem 21. *The following problem is P^{NP}-hard:*
INPUT: A zero-test-free OCA $\mathcal{O}$, a state q_0 of $\mathcal{O}$, and an EF *formula φ.*
QUESTION: $(\mathcal{T}(\mathcal{O}), (q_0, 0)) \models \varphi$?

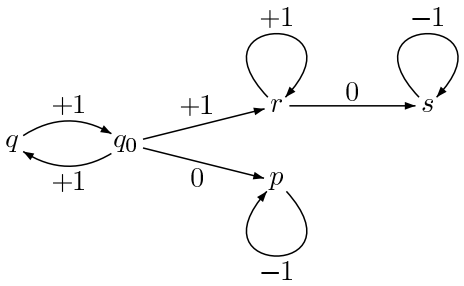
Proof. Let us take a boolean formula $\psi(x_1, \ldots, x_m)$. By Theorem 12 it suffices to construct (in logspace) a zero-test-free OCA $\mathcal{O}_\psi$ with a distinguished state q_0 and an EF formula φ_ψ such that $(\mathcal{T}(\mathcal{O}_\psi), (q_0, 0)) \models \varphi_\psi$ if and only if ψ is satisfiable and the maximal number $M \in [0, 2^m - 1]$ with $\psi(\mathsf{BIN}_m(M)) = 1$ is even.

By Proposition 13 we can compute in space $\log m$ a list $p_1, \ldots, p_m$ of the first m consecutive primes. Moreover, let us compute in space $\log m$ the circuit $B = B_m((x_{i,r})_{i\in[m],0\leq r<p_i})$ of logarithmic depth and fan-in at most two from Theorem 14. We combine B with the boolean formula $\psi(x_1, \ldots, x_m)$ and obtain a boolean circuit $C = C((x_{i,r})_{i\in[m],0\leq r<p_i})$ such that for every number $0 \leq M \leq 2^m - 1$:

$$\psi(\mathsf{BIN}_m(M)) = 1 \iff C(\mathsf{CRR}_m(M)) = 1. \tag{7}$$

As in the proof of Theorem 18 let G be the boolean formula $\bigwedge_{i\in[m]} x_{i,r_i}$ were $r_i = 2^m \bmod p_i$ for $i \in [m]$.

The main structure of the zero-test-free OCA $\mathcal{O}_\psi$ is described by the following diagram:



From the states q_0, p, r, and s some further 0-labeled transitions emanate to zero-test-free OCA of the form constructed in Proposition 20:

- From q_0 a transition into the initial state in of a copy of $\mathcal{O}_C$.
- From p and s a transition into the initial state in of a copy of $\mathcal{O}_G$.
- From r a transition into the initial state in of a copy of $\mathcal{O}_{\neg C}$.

Now our EF formula φ_ψ expresses the following: We can reach a configuration (q_0, M_1) from $(q_0, 0)$ in the zero-test-free OCA $\mathcal{O}_\psi$ such that the following holds:

- $C(\mathsf{CRR}_m(M_1)) = 1$,
- from (q_0, M_1) we cannot reach a configuration (p, M_0) with $0 \leq M_0 \leq M_1$ and $G(\mathsf{CRR}_m(M_0)) = 1$ (i.e., $M_0 = 2^m \bmod \prod_{i=1}^{m} p_i$), and
- for all configurations (r, M_2) that are reachable from (q_0, M_1) (hence $M_2 > M_1$) the following holds: If we cannot reach a configuration (s, M_3) from (r, M_2) with $G(\mathsf{CRR}_m(M_3)) = 1$ then $C(\mathsf{CRR}_m(M_2)) = 0$.

Using the formulas constructed in Proposition 20, it is straightforward to transform this description into a real EF formula. This concludes the proof. □

At the moment we cannot prove P^{NP}-hardness for the data complexity of EF over OCA. For this, it would be sufficient to have a fixed EF formula $\varphi(C)$ in (6). Note that this formula only depends on the number of layers k of the circuit C. Hence, if C is from an AC^0-circuit family, then $\varphi(C)$ is in fact a fixed formula. In our case, the circuit is the composition of two circuits, one from an NC^1-circuit family (coming from Theorem 14, where we could even assume a TC^0-circuit family) and a boolean formula, which can be assumed to be in conjunctive normal form. Hence, the main obstacle for getting a fixed formula is the fact that converting from Chinese remainder representation to binary representation is not possible in AC^0 (this is provably the case).

9 Reachability objectives on one-counter Markov decision processes

In this section we show that the techniques developed in the previous sections can be used to improve a lower bound on verifying reachability objectives on one-counter Markov decision processes from [8].

A *probability distribution* on a non-empty finite set S is a function $f : S \to \{x \in \mathbb{Q} \mid 0 \leq x \leq 1\}$ such that $\sum_{s \in S} f(s) = 1$. We restrict here to rational probabilities, in order to get finite representations for probability distributions. A (image-finite) *Markov chain* is a triple $\mathcal{C} = (S, \to, f)$, where $(S, \to)$ is an image-finite and deadlock-free directed graph (S is also called the set of states of $\mathcal{C}$) and f assigns to each $s \in S$ a probability distribution $f(s)$ over all (the finitely many) successors of s w.r.t. $\to$. If $s \to t$, then we also use the notations $f(s,t) = x$ or $s \xrightarrow{x} t$ for $(f(s))(t) = x \in \mathbb{Q}$. A (image-finite) *Markov decision process* (MDP) is a triple $\mathcal{D} = (V, \hookrightarrow, f)$, where $(V, \hookrightarrow)$ is again an image-finite and deadlock-free directed graph, the set V of vertices is partitioned as $V = V_N \uplus V_P$ (V_N is the set of *nondeterministic* vertices, V_P is the set of *probabilistic* vertices), and f assigns to each probabilistic vertex $v \in V_P$ a probability distribution on v's successors. A *strategy* σ is a function that assigns to each wv with $w \in V^*$ and $v \in V_N$ a probability distribution on v's successors. If σ assigns to wv and v' (where $v \hookrightarrow v'$) the probability x, then we write $\sigma(wv, v') = x$. Every strategy σ determines a Markov chain $\mathcal{D}(\sigma) = (V^+, \to, f)$, where $wv \xrightarrow{x} wvv'$ if

and only if $v \hookrightarrow v'$ and moreover either $v \in V_P$ and $f(v,v') = x$, or $v \in V_N$ and $\sigma(wv,v') = x$. Let $\mathsf{path}_\omega(\mathcal{D}) = \mathsf{path}_\omega(V, \hookrightarrow)$ and $\mathsf{path}_\omega(\mathcal{D}(\sigma)) = \mathsf{path}_\omega(V^+, \to)$; paths in these sets will be called *runs* in $\mathcal{D}$ or $\mathcal{D}(\sigma)$, respectively. Note that every run in $\mathcal{D}$ corresponds to a unique run in $\mathcal{D}(\sigma)$ and vice versa in a natural way. In order to simplify notation, we will quite often identify these corresponding runs. Let us fix a set of *target vertices* $T \subseteq V$ of the MDP $\mathcal{D}$. For each strategy σ and each vertex $v \in V$ of $\mathcal{D}$, let

$$\text{Reach}_T^\sigma(v) = \{w_1 w_2 \cdots \in \mathsf{path}_\omega(\mathcal{D}(\sigma)) \mid w_1 = v \text{ and } \exists i \geq 1 : w_i \in V^*T\}$$

denote all runs in $\mathcal{D}(\sigma)$ that start in v and that satisfy the reachability objective T in $\mathcal{D}$. For each T, each σ and each v, the set $\text{Reach}_T^\sigma(v)$ is measurable [4]. The probability $\mathcal{P}(\text{Reach}_T^\sigma(v))$ for the set $\text{Reach}_T^\sigma(v)$ can be obtained as follows: Take all finite paths $w \in \mathsf{path}_+(\mathcal{D}(\sigma))$ that start in v and such that the last state of w is from V^*T but no previous state in w is from V^*T (this set is prefix free). For each such finite path $w = w_1 \cdots w_n$ such that $w_i \xrightarrow{x_i} w_{i+1}$ in $\mathcal{D}(\sigma)$ the probability is $x_1 \cdot x_2 \cdots x_{n-1}$. Finally, the probability for $\text{Reach}_T^\sigma(v)$ is the (possibly infinite) sum of all these probabilities. Now, let us define the *T-reachability value in v* by

$$\text{ValReach}_T(v) \;=\; \sup\{\mathcal{P}(\text{Reach}_T^\sigma(v)) \mid \sigma \text{ is a strategy in } \mathcal{D}\}.$$

Observe that it is not required that this supremum is actually reached by a certain strategy σ. If however a strategy σ does reach the T-reachability value, i.e., $\mathcal{P}(\text{Reach}_T^\sigma(v)) = \text{ValReach}_T(v)$, then σ is called *optimal*.

A one-counter Markov decision process (OC-MDP) is a tuple $\mathcal{M} = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$, where $Q = Q_N \uplus Q_P$ is a finite set of *control locations* which is partitioned into *nondeterministic control locations* Q_N and *probabilistic control locations* Q_P, $\delta_0 \subseteq Q \times \{0,1\} \times Q$ is a set of *zero transitions* and $\delta_{>0} \subseteq Q \times \{-1,0,1\} \times Q$ is a set of *positive transitions* such that each $q \in Q$ has at least one outgoing zero transition and at least one outgoing positive transition, and finally f_0 (resp. $f_{>0}$) assigns to each $q \in Q_P$ a probability distribution over all outgoing zero (resp. positive) transitions of q. The MDP that $\mathcal{M}$ describes is $\mathcal{D}(\mathcal{M}) = (V, \hookrightarrow, f)$, where

- $V_N = Q_N \times \mathbb{N}$ and $V_P = Q_P \times \mathbb{N}$, and
- $(q,n) \hookrightarrow (q', n+i)$ if and only if one of the following two conditions holds:
  - $n = 0$ and $(q,i,q') \in \delta_0$. If furthermore $q \in Q_P$, then f assigns to $(q,n) \hookrightarrow (q', n+i)$ the probability $f_0(q,i,q')$.
  - $n > 0$ and $(q,i,q') \in \delta_{>0}$. If furthermore $q \in Q_P$, then f assigns to $(q,n) \hookrightarrow (q', n+i)$ the probability $f_{>0}(q,i,q')$.

Given an OC-MDP $\mathcal{M} = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$ and a set of control locations $R \subseteq Q$, define

$$\mathsf{ValOne}(R) \;=\; \{(q,n) \in Q \times \mathbb{N} \mid \text{ValReach}_{R \times \{0\}}(q,n) = 1\}$$

and

$$\mathsf{OptValOne}(R) \;=\; \{(q,n) \in Q \times \mathbb{N} \mid \exists \text{ strategy } \sigma : \mathcal{P}(\text{Reach}_{R\times\{0\}}^\sigma(q,n)) = 1\}$$

(both sets are defined w.r.t $\mathcal{D}(\mathcal{M})$). In other words: $\mathsf{ValOne}(R)$ is the set of all states (q,n) of the MDP $\mathcal{D}(\mathcal{M})$ such that for every $\varepsilon > 0$ there exists a strategy σ_ε under which the probability of reaching from (q,n) a control location in R and at the same time having counter value 0 is at least $1 - \varepsilon$. $\mathsf{OptValOne}(R)$ is the set of all states (q,n) of the MDP $\mathcal{D}(\mathcal{M})$ for which there exists a specific strategy under which this probability becomes 1. The following theorem recalls the complexity bounds that are known for the latter variant.

Theorem 22 ([8]). *The following problem is* PSPACE*-hard and in* EXPTIME*:*
INPUT: An OCA-MDP $\mathcal{M} = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$, $R \subseteq Q$, and $q \in Q$.
QUESTION: $(q,0) \in \mathsf{OptValOne}(R)$*?*

The lower bound in Theorem 22 was proven by a reduction from the PSPACE-complete emptiness problem for alternating finite word automata over a singleton alphabet ([23], see also [24] for a simplified presentation).

Theorem 23 ([8]). *The following problem is hard for every level of* BH*:*
INPUT: An OC-MDP $\mathcal{M} = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$, $R \subseteq Q$, and $q \in Q$.
QUESTION: $(q, 0) \in \mathsf{ValOne}(R)$*?*

Currently, it is open whether the problem stated in Theorem 23 is decidable; the corresponding problem for MDPs defined by pushdown automata is undecidable [15].

From the proof of Theorem 23 it can be seen that the authors prove actually hardness for $\mathsf{P}^{\mathsf{NP}[\log]}$. Moreover, it is pointed out in [8] that various difficulties arise when trying to improve the latter lower bound. In this section, we will improve the lower bound for membership in $\mathsf{ValOne}(R)$ to PSPACE. We aim at demonstrating that the tools from computational complexity that we have used so far, can again be applied for the $\mathsf{ValOne}(R)$ problem, rather than emphasizing that one can prove that $\mathsf{ValOne}(R)$ is indeed PSPACE-hard: as mentioned above, $\mathsf{ValOne}(R)$ is not known to be decidable. From our proof one can easily see that we reprove PSPACE-hardness of OptValOne as a byproduct. But first, we need the following proposition.

Proposition 24. *The following problem can be solved by a logspace transducer:*
INPUT: A list of the first m consecutive (unary encoded) prime numbers and a boolean formula $F = F((x_{i,r})_{i\in[m],0\leq r<p_i})$.
OUTPUT: An OC-MDP $\mathcal{M} = \mathcal{M}(F)$ with control locations Q, a set $R = R(F) \subseteq Q$, and some control location $q_F \in Q$ such that for every number $0 \leq M < \prod_{i=1}^{m} p_i$ the following holds:

- *If $F(\mathsf{CRR}_m(M)) = 1$, then there exists a strategy σ such that $\mathcal{P}(Reach^{\sigma}_{R\times\{0\}}(q_F, M)) = 1$.*
- *If $F(\mathsf{CRR}_m(M)) = 0$, then for every strategy σ we have $\mathcal{P}(Reach^{\sigma}_{R\times\{0\}}(q_F, M)) \leq 1 - 2^{-|F|}$.*

Proof. As in the proof of Proposition 17, we can assume that F is a positive boolean formula. The OC-MDP $\mathcal{M} = \mathcal{M}(F) = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$ will have for each subformula G of F a control location q_G. If G is of the form $G = G_1 \vee G_2$, then q_G will be nondeterministic and both in δ_0 and in $\delta_{>0}$ there is a transition from q_G to both q_{G_1} and q_{G_2} that does not change the counter value. If G is of the form $G = G_1 \wedge G_2$, then q_G will be probabilistic and both in δ_0 and in $\delta_{>0}$ there will be a transition to both q_{G_1} and q_{G_2} that does not change the counter value and which will be chosen with probability $\frac{1}{2}$ each. Now assume that G is a variable $x_{i,r}$. Recall that $x_{i,r}$ is set to 1 if and only if $M \bmod p_i = r$. We introduce in $\mathcal{M}$ further (deterministically behaving) control locations $q(j, p_i)$ for $0 \leq j < p_i$ that allow to test if M is congruent r modulo p_i by allowing the following transitions in $\delta_{>0}$ for each $0 \leq j < p_i$:

$$(q(j, p_i), -1, q(j - 1 \bmod p_i, p_i))$$

Since each $q(j, p_i)$ must have an outgoing transition both in δ_0 and $\delta_{>0}$, we add the transition

$$(q(j, p_i), 0, q(j, p_i))$$

to δ_0 for each $0 \leq j < p_i$. We put $q_{x_{i,r}}$ to be nondeterministic with a transition both in δ_0 and in $\delta_{>0}$ from $q_{x_{i,r}}$ to $q(r, p_i)$ that does not change the counter value. Finally we put $R = \{q(0, p_i) \mid i \in [m]\}$.

Assume first that $F(\mathsf{CRR}_m(M)) = 1$. We prove that there exists a strategy σ such that

$$\mathcal{P}(\mathrm{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) = 1$$

in $\mathcal{D}(\mathcal{M})$. Note that the only nondeterministic states in $\mathcal{D}(\mathcal{M})$ that have more than one successor are states which correspond to a disjunctive subformula $G = G_1 \vee G_2$ of F. If $G(\mathsf{CRR}_m(M)) = 1$, then there exists some $i \in \{1, 2\}$ such that $G_i(\mathsf{CRR}_m(M)) = 1$. Our strategy σ will choose (q_G, M)'s successor (q_{G_i}, M) with probability 1. If $G(\mathsf{CRR}_m(M)) = 0$, then the choice of σ is irrelevant and we let σ choose (q_G, M)'s successor uniformly distributed, say. It is now easy to verify that $\mathcal{P}(\mathrm{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) = 1$.

On the other hand, assume that $F(\mathsf{CRR}_m(M)) = 0$ and consider an arbitrary strategy σ. The question is how close can $\mathcal{P}(\mathrm{Reach}^{\sigma}_{R\times\{0\}}(q_F, M))$ reach 1. We prove by induction on the structure of the formula F that

$$\mathcal{P}(\mathrm{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) \leq 1 - 2^{-k}, \tag{8}$$

where k is the number of conjunctions that appear in F. If F is a variable $x_{i,r}$, then

$$\mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) = 0 = 1 - 2^0.$$

If $F = F_1 \vee F_2$ then $F_1(\mathsf{CRR}_m(M)) = F_2(\mathsf{CRR}_m(M)) = 0$. Assume that σ assigns to the transition from (q_F, M) to (q_{F_i}, M) the probability x_i, where $x_1 + x_2 = 1$. With the induction hypothesis, we get

$$\begin{aligned}\mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) &= x_1 \cdot \mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_{F_1}, M)) + x_2 \cdot \mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_{F_2}, M)) \\ &\leq x_1(1 - 2^{-k_1}) + x_2(1 - 2^{-k_2}),\end{aligned}$$

where k_i the number of conjunctions that appear in F_i. Since $k_i \leq k$, we get (8). Finally, assume that $F = F_1 \wedge F_2$ and let k_i be the number of conjunctions that appear in F_i. Hence, $k_i \leq k-1$. If $F_1(\mathsf{CRR}_m(M)) = F_2(\mathsf{CRR}_m(M)) = 0$ then we get $\mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) \leq 1 - 2^{-k+1} \leq 1 - 2^{-k}$. On the other hand, if e.g. $F_1(\mathsf{CRR}_m(M)) = 0$ but $F_2(\mathsf{CRR}_m(M)) = 1$ (the other case is symmetric), then we get

$$\begin{aligned}\mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) &= \frac{1}{2} \cdot \mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_{F_1}, M)) + \frac{1}{2} \cdot \mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_{F_2}, M)) \\ &\leq \frac{1}{2} \cdot (1 - 2^{-k+1}) + \frac{1}{2} = 1 - 2^{-k}.\end{aligned}$$

This concludes the proof of (8). Since $k \leq |F|$ we obtain $\mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(q_F, M)) \leq 1 - 2^{-|F|}$. This concludes the proof of Proposition 24. □

Theorem 25. *The following problem is* PSPACE*-hard:*
INPUT: An OC-MDP $\mathcal{M} = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$, $R \subseteq Q$, and $q \in Q$.
QUESTION: $(q, 0) \in \mathsf{ValOne}(R)$*?*

Proof. Take an arbitrary PSPACE-complete language $L \subseteq \{0,1\}^*$. Fix the polynomial $p(n)$ and the NFA $A = (S, \{0,1\}, \delta, s_0, S_f)$ over the alphabet $\{0,1\}$ from Proposition 15. Let $x \in \{0,1\}^*$ be an input of length n, and let $m := p(n)$, where w.l.o.g. $m > 1$. By Proposition 15 one can construct in space $O(\log n)$ a boolean formula F with propositional variables $x_{i,r}$ ($i \in [m]$ and $0 \leq r < p_i$) such that

$$x \in L \iff \prod_{M=0}^{2^m-1} F(\mathsf{CRR}_m(M)) \in L(A). \tag{9}$$

By Proposition 13 we can compute in space $O(\log m) = O(\log n)$ a list $p_1, \ldots, p_m$ of the first m prime numbers in unary notation. Note that $\prod_{i=1}^{m} p_i > 2^m$ since $m > 1$.

By doubling, if necessary, the set of final states of A we can assume that states from S_f do not have outgoing transitions but every state from $S \setminus S_f$ has at least one outgoing transition. This assumption will slightly simplify our construction.

Let $G = \bigwedge_{i\in[m]} x_{i,r_i}$ with $r_i = 2^m \bmod p_i$ for each $i \in [m]$ be the boolean formula that tests if M equals 2^m. We will build an OC-MDP $\mathcal{M} = (Q, \delta_0, \delta_{>0}, f_0, f_{>0})$ with $S \subseteq Q$ and a target set of control locations $R \subseteq Q$ such that

$$\prod_{M=0}^{2^m-1} F(\mathsf{CRR}_m(M)) \in L(A) \iff \text{ValReach}_{R\times\{0\}}(s_0, 0) = 1.$$

Moreover, our reduction will have the additional property that

$$\text{ValReach}_{R\times\{0\}}(s_0, 0) = 1 \iff \exists \text{ strategy } \sigma : \mathcal{P}(\text{Reach}^{\sigma}_{R\times\{0\}}(s_0, 0)) = 1.$$

Hence, we prove PSPACE-hardness of OptValOne as a byproduct. The control locations in $S \setminus S_f$ are nondeterministic in $\mathcal{M}$ ($\mathcal{M}$ will hence behave nondeterministically in control locations from $S \setminus S_f$). The NFA A on input $F(\mathsf{CRR}_m(0)) \cdots F(\mathsf{CRR}_m(2^m - 1))$ will be simulated by $\mathcal{M}$ from state $(s_0, 0)$ by consecutively incrementing the counter and checking if for the current counter value M and for the current

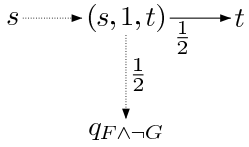
(to be simulated) b-labeled transition of A we have $F(\mathsf{CRR}_m(M)) = b$. This simulation will be done until a state $(s, 2^m)$ with $s \in S_f$ is reached. Recall that by Proposition 24 we can compute OC-MDPs $\mathcal{M}(F \wedge \neg G)$, $\mathcal{M}(\neg F \wedge \neg G)$, and $\mathcal{M}(G)$ together with sets of control locations $R(F \wedge \neg G)$, $R(\neg F \wedge \neg G)$, and $R(G)$, and control locations $q_{F \wedge \neg G}$, $q_{\neg F \wedge \neg G}$, and q_G such that, e.g., $\mathcal{M}(F \wedge \neg G)$ satisfies for each $0 \leq M < \prod_{i=1}^{m} p_i$:

$$F(\mathsf{CRR}_m(M)) = 1 \wedge M \neq 2^m \Rightarrow \exists \text{ strategy } \sigma : \mathcal{P}(\mathrm{Reach}^{\sigma}_{R(F \wedge \neg G) \times \{0\}}(q_{F \wedge \neg G}, M)) = 1$$

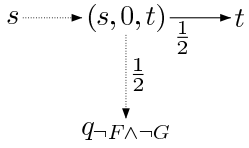
$$F(\mathsf{CRR}_m(M)) = 0 \vee M = 2^m \Rightarrow \forall \text{ strategies } \sigma : \mathcal{P}(\mathrm{Reach}^{\sigma}_{R(F \wedge \neg G) \times \{0\}}(q_{F \wedge \neg G}, M)) \leq 1 - 2^{-|F \wedge \neg G|}$$

The OC-MDPs $\mathcal{M}(\neg F \wedge \neg G)$ and $\mathcal{M}(G)$ have analogous properties.

In the following diagrams we draw transitions that do *not* modify the counter value by dashed lines and we draw transitions that increase the counter value by one in normal width. We realize each NFA-transition $(s, 1, t) \in \delta$ (where necessarily $s \in S \setminus S_f$) both in δ_0 and in $\delta_{>0}$ by



whereas each transition $(s, 0, t) \in \delta$ (where necessarily $s \in S \setminus S_f$) is realized in $\mathcal{M}$ by



i.e. we connect the intermediate control location $(s, b, t) \in \delta$ to $\mathcal{M}(F \wedge \neg G)$ (if $b = 1$) or $\mathcal{M}(\neg F \wedge \neg G)$ (if $b = 0$) for checking if $F(\mathsf{CRR}_m(M)) = b$ and $M < 2^m$ for the current counter value M. Moreover, for all final states $s \in S_f$ we add a transition $s \xrightarrow{1} q_G$ to both δ_0 and $\delta_{>0}$ that does not change the counter value. As expected, we put $R = R(F \wedge \neg G) \cup R(\neg F \wedge \neg G) \cup R(G)$. Let $\mathcal{D} = \mathcal{D}(\mathcal{M})$ in the following. Note that since every non-final state has at least one outgoing transition in A, $\mathcal{D}$ is indeed an MDP, i.e., the underlying graph is deadlock-free.

Now assume that $x \in L$. We show that there exists a strategy σ such that $\mathcal{P}(\mathrm{Reach}^{\sigma}_{R \times \{0\}}(s_0, 0)) = 1$. Since $x \in L$, we have $\prod_{M=0}^{2^m-1} F(\mathsf{CRR}_m(M)) \in L(A)$ along with some accepting run

$$s_0 \xrightarrow{b_0} s_1 \xrightarrow{b_1} \quad \cdots \quad s_{2^m-1} \xrightarrow{b_{2^m-1}} s_{2^m} \in S_f,$$

where $s_M \notin S_f$ and $b_M = F(\mathsf{CRR}_m(M))$ for all $M \in [0, 2^m - 1]$. For each $M \in [0, 2^m - 1]$ our strategy σ will assign to (s_M, M)'s successor $((s_M, b_M, s_{M+1}), M)$ probability 1. Moreover, by Proposition 24 we can choose the strategy σ such that:

$$\begin{aligned} b_M = 1 \quad &\Longrightarrow \quad \mathcal{P}(\mathrm{Reach}^{\sigma}_{R \times \{0\}}(q_{F \wedge \neg G}, M)) = 1 \\ b_M = 0 \quad &\Longrightarrow \quad \mathcal{P}(\mathrm{Reach}^{\sigma}_{R \times \{0\}}(q_{\neg F \wedge \neg G}, M)) = 1 \end{aligned}$$

for each $0 \leq M < 2^m$ and $\mathcal{P}(\mathrm{Reach}^{\sigma}_{R \times \{0\}}(q_G, 2^m)) = 1$. It follows

$$\mathcal{P}(\mathrm{Reach}^{\sigma}_{R \times \{0\}}(s_0, 0)) = 1.$$

Conversely, assume now that $x \notin L$. Recall that all states from S_f have no outgoing transitions in A and all states from $S \setminus S_f$ have at least one outgoing transition. Hence, every maximal path in A (a path is maximal if it cannot be extended) is either infinite or terminates in a final state. With $x \notin L$ and (9) we obtain the following claim:

Claim 1. Every maximal path in the automaton A that starts in the initial state s_0 has exactly one prefix

$$s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \quad \cdots \quad s_M \xrightarrow{c_M} s_{M+1}$$

that satisfies one of the following three mutually exclusive cases (recall that every state in S_f has no outgoing transitions):

(a) $M \leq 2^m - 1$, $c_M \neq F(\mathsf{CRR}_m(M))$, and $c_N = F(\mathsf{CRR}_m(N))$ for all $N \in [0, M-1]$
(b) $M < 2^m - 1$, $c_N = F(\mathsf{CRR}_m(N))$ for all $N \in [0, M]$, and $s_{M+1} \in S_f$
(c) $M = 2^m - 1$, $c_N = F(\mathsf{CRR}_m(N))$ for all $N \in [0, M]$, and $s_{M+1} \notin S_f$

Our goal is to prove a global non-zero lower bound on the probability of runs in $\mathcal{D}(\sigma)$ that begin in $(s_0, 0)$ and that do *not* reach $R \times \{0\}$, where σ is an arbitrary strategy. For this, let us first fix an arbitrary strategy σ in $\mathcal{D}$. We distinguish the following three types (A), (B) and (C) of finite paths π in the Markov chain $\mathcal{D}(\sigma)$ (type (X) corresponds to case (x) from Claim 1):

Case (A): π is of the form

$$(s_0,0) \xrightarrow{\alpha_0} ((s_0,c_0,s_1),0) \xrightarrow{\frac{1}{2}} (s_1,1) \xrightarrow{\alpha_1} ((s_1,c_1,s_2),1) \cdots$$
$$((s_{M-1},c_{M-1},s_M),M-1) \xrightarrow{\frac{1}{2}} (s_M,M) \xrightarrow{\alpha_M} ((s_M,c_M,s_{M+1}),M),$$

where $M \leq 2^m - 1$, $c_M \neq F(\mathsf{CRR}_m(M))$, and $c_N = F(\mathsf{CRR}_m(N))$ for all $N \in [0, M-1]$. The α_N are probabilities that result from the strategy σ. Let $\alpha = \prod_{N \in [0,M]} \alpha_N$. The probability for the set of all runs from $(s_0, 0)$ that (i) start with π, then (ii) proceed to $(q_{F \wedge \neg G}, M)$ (if $c_M = 1$) or to $(q_{\neg F \wedge \neg G}, M)$ (if $c_M = 0$), and (iii) do *not* visit $R \times \{0\}$ is at least

$$\alpha \cdot 2^{-(M+1)} \cdot 2^{-|\neg F \wedge \neg G|} \geq \alpha \cdot 2^{-(2^m + |\neg F \wedge \neg G|)}.$$

Case (B): π is of the form

$$(s_0,0) \xrightarrow{\beta_0} ((s_0,c_0,s_1),0) \xrightarrow{\frac{1}{2}} (s_1,1) \xrightarrow{\beta_1} ((s_1,c_1,s_2),1) \xrightarrow{\frac{1}{2}} (s_2,2) \cdots$$
$$(s_M,M) \xrightarrow{\beta_M} ((s_M,c_M,s_{M+1}),M) \xrightarrow{\frac{1}{2}} (s_{M+1},M+1) \xrightarrow{1} (q_G,1),$$

where $M < 2^m - 1$, $c_N = F(\mathsf{CRR}_m(N))$ for all $N \in [0, M]$, and $s_{M+1} \in S_f$. Let $\beta = \prod_{N \in [0,M]} \beta_N$. The probability for the set of all runs from $(s_0, 0)$ that (i) start with π and (ii) do *not* visit $R \times \{0\}$ is at least

$$\beta \cdot 2^{-(M+1)} \cdot 2^{-|G|} \geq \beta \cdot 2^{-(2^m + |\neg F \wedge \neg G|)}.$$

Case (C): π is of the form

$$(s_0,0) \xrightarrow{\gamma_0} ((s_0,c_0,s_1),0) \xrightarrow{\frac{1}{2}} (s_1,1) \xrightarrow{\gamma_1} ((s_1,c_1,s_2),1) \cdots$$
$$(s_{2^m-1}, 2^m - 1) \xrightarrow{\gamma_{2^m-1}} ((s_{2^m-1}, c_{2^m-1}, s_{2^m}), 2^m - 1) \xrightarrow{\frac{1}{2}} (s_{2^m}, 2^m),$$

where $c_N = F(\mathsf{CRR}_m(N))$ for all $N \in [0, 2^m - 1]$ and $s_{2^m} \notin S_f$. Let $\gamma = \prod_{N \in [0, 2^m - 1]} \gamma_N$. The probability of the set of runs in $\mathcal{D}(\sigma)$ that (i) begin with π, then (ii) proceed (via an intermediate control location of the form (s_{2^m}, b, t)) to either $(q_{F \wedge \neg G}, 2^m)$ or $(q_{\neg F \wedge \neg G}, 2^m)$ and (iii) that do *not* reach $R \times \{0\}$ is at least

$$\gamma \cdot 2^{-(2^m+1)} \cdot 2^{-|\neg F \wedge \neg G|} = \gamma \cdot 2^{-(2^m + 1 + |\neg F \wedge \neg G|)}.$$

Now, the crucial point is that the sum of all values α from (A), all values β from (B), and all values γ from (C) is 1. This is a consequence of Claim 1 and the fact that the nondeterministic choices in $\mathcal{D}$ correspond exactly to the selection of transitions of the NFA A. Since moreover the set of paths in (A), (B), and (C) are pairwise disjoint, it follows that the probability of the set of runs that do *not* reach $R \times \{0\}$ is at least $2^{-(2^m + 1 + |\neg F \wedge \neg G|)}$. This concludes the proof of the theorem. □

10 Timed automata

In this section, we present an application of the serializability technique to timed automata. Let us start with the definition of timed automata, see e.g. [7] for more details.

As usual, we fix a countable set $\mathcal{P}$ of atomic propositions. Let C be a finite set, whose elements are called *clocks*. A mapping $t \in \mathbb{R}_+^C$ from C to the set $\mathbb{R}_+$ of positive real numbers is also called a *clock valuation*. The set $B(C)$ of *clock constraints* over C is the set of all boolean formulas with atomic formulas of the form $c \sim k$, where $c \in C$, $k \in \mathbb{N}$ and $\sim \in \{\leq, \geq\}$. We use the usual abbreviations, e.g., we write $c = k$ for $c \leq k \wedge c \geq k$. Let us define the size of the clock constraint $c \sim k$ as $|c \sim k| = \lceil \log k \rceil$; it is the length of the binary encoding of the number k. A clock valuation $t \in \mathbb{R}_+^C$ satisfies a clock constraint $\gamma \in B(C)$, if the formula γ becomes true, when each clock $c \in C$ is replaced by the value $t(c)$.

A *timed automaton* (TA) is a tuple $\mathcal{A} = (Q, \{Q_p \mid p \in \mathcal{P}\}, C, \delta)$, where

- Q is a finite set of *control locations*,
- $Q_p \subseteq Q$ for each $p \in \mathcal{P}$ but $Q_p = \emptyset$ for all but finitely many $p \in \mathcal{P}$,
- C is a finite set of *clocks*, and
- $\delta \subseteq Q \times B(C) \times 2^C \times Q$ is a finite set of *transitions*.

The *size* of the TA $\mathcal{A}$ is defined as $|\mathcal{A}| = |Q| + |C| + \sum_{p \in \mathcal{P}} |Q_p| + \sum_{(p,\gamma,R,q) \in \delta} |\gamma|$. A timed automaton $\mathcal{A} = (Q, \{Q_p \mid p \in \mathcal{P}\}, C, \delta)$ defines a transition system

$$\mathcal{T}(\mathcal{A}) = (Q \times \mathbb{R}_+^C, \{Q_p \times \mathbb{R}_+^C \mid p \in \mathcal{P}\}, \rightarrow),$$

where $(q, t) \rightarrow (q', t')$ if and only if one of the following two cases holds:

- $q = q'$ and there exists $d \in \mathbb{R}_+$ such that $t'(c) = t(c) + d$ for all $c \in C$ (time d elapses).
- There exists a transition $(q, \gamma, R, q') \in \delta$ such that (i) the mapping $t : C \rightarrow \mathbb{R}_+$ satisfies the clock constraint γ, (ii) $t'(c) = t(c)$ for all $c \in C \setminus R$, and (iii) $t'(c) = 0$ for all $c \in R$ (i.e., all clocks from the set R are reset).

10.1 CTL model checking on timed automata

In this section, we will only consider timed automata with only two clocks x and y. For a natural number m let $t_m : \{x, y\} \rightarrow \mathbb{R}_+$ be the clock valuation with $t_m(x) = m$ and $t_m(y) = 0$.

In [25], it was shown that model checking CTL over 2-clock timed automata is PSPACE-complete. The proof in [25] for PSPACE-hardness only works if the timed automaton and the CTL formula are part of the input. Here we sharpen this result by showing that model checking CTL over 2-clock timed automata is PSPACE-hard already for a fixed CTL formula. Let us fix the set of propositions $\mathcal{P} = \{\alpha, \beta, \gamma\}$ for this section. The following lemma is implicitly shown in [25], see Figure 4 in that paper.

Lemma 26 ([25]). *There is a logspace transducer that computes from two unary encoded numbers $1 \leq i \leq n$ and a bit $b \in \{0, 1\}$ a 2-clock TA $\mathcal{A}_{n,i,b}$ with a distinguished control location $\mathsf{in}_{n,i,b}$ such that for every number $0 \leq M \leq 2^n - 1$ the following are equivalent:*

- $\mathsf{bit}_i(M) = b$
- *In the transition system $\mathcal{T}(\mathcal{A}_{n,i,b})$ there is a path from $(\mathsf{in}_{n,i,b}, t_M)$ to a γ-labeled state.*

Proof. Let us first construct the TA $\mathcal{A}_{n,i,1}$. The set of control locations is $\{q_0, q_1, \ldots, q_n, p\}$, where $\mathsf{in}_{n,i,b} = q_0$ and p is the only control location labeled with proposition γ. We add the following transitions:

- $(q_{j-1}, (y = 0), \emptyset, q_j)$ for all $1 \leq j \leq n$
- $(q_{j-1}, (y = 2^{j-1}), \{y\}, q_j)$ for all $1 \leq j \leq n$ with $j \neq i$
- $(q_n, (y = 0 \wedge x = 2^n - 1), \emptyset, p)$

For the TA $\mathcal{A}_{n,i,0}$ we replace the transition $(q_{i-1}, (y = 0), \emptyset, q_i)$ by the transition $(q_{i-1}, (y = 2^{i-1}), \{y\}, q_i)$.

The intuition for the TA $\mathcal{A}_{n,i,1}$ is the following: Clearly, in the transition system $\mathcal{T}(\mathcal{A}_{n,i,1})$ there is a path from $(\mathsf{in}_{n,i,b}, t_M)$ to a γ-labeled state if and only if there is a path from (q_0, t_M) to (q_n, t_{2^n-1}). For each $1 \leq j \leq n$ with $i \neq j$ we can either decide to add 2^{j-1} to the current value of clock x (using the transition $(q_{j-1}, (y = 2^{j-1}), \{y\}, q_j)$) or to add 0 to the current value of clock x (using the transition $(q_{j-1}, (y = 0), \emptyset, q_j)$). Moreover, for $j = i$, we are forced to add 0. Using these transitions, we can move from state (q_0, t_M) to (q_n, t_{2^n-1}) if and only if the i^{th} bit of M is 1. □

The proof of Proposition 27 below is very similar to the proof of Proposition 17.

Proposition 27. *For the fixed* EF *formula* $\varphi = (\alpha \to \mathsf{EX}(\beta \wedge \mathsf{EF}(\gamma)))$ *the following problem can be solved with a logspace transducer:*
INPUT: A boolean formula $F = F(x_1, \ldots, x_n)$
OUTPUT: A 2-clock TA $\mathcal{A}(F)$ *with distinguished control locations* in *and* out *such that for every number* $0 \leq M \leq 2^n - 1$ *the following are equivalent:*

- $F(\mathsf{BIN}_n(M)) = 1$
- *There exists a* $[\![\varphi]\!]_{\mathcal{T}(\mathcal{A}(F))}$*-path from* (in, t_M) *to* (out, t_M) *in* $\mathcal{T}(\mathcal{A}(F))$.

Proof. W.l.o.g. we may assume that negations occur in F only in front of variables.[5] Let S_F be the syntax tree of F and let V_F be the set of nodes of S_F. Analogously to the proof of Proposition 17, the tree S_F is traversed with the TA $\mathcal{A}(F)$ in a depth first manner. Each time a node that is labeled with a variable x_i (resp., a negated variable $\neg x_i$) is seen, the TA may also enter the TA $\mathcal{A}_{n,i,1}$ (resp. $\mathcal{A}_{n,i,0}$) in order to check whether the i^{th} bit of the current value of clock x (which is M) is 1 (resp., 0). We first construct all TAs $\mathcal{A}_{n,i,b}$ from Lemma 26 for $1 \leq i \leq n$ and $b \in \{0,1\}$. Let $\mathcal{A}_n$ be the disjoint union of all these TAs, where in addition every control location $\mathsf{in}_{n,i,b}$ is labeled with the proposition β. To construct $\mathcal{A}(F)$, we add to $\mathcal{A}_n$ all control locations $\mathsf{in}(v)$ and $\mathsf{out}(v)$ for $v \in V_F$. We label every control location $\mathsf{in}(v)$, where $v \in V_F$ is labeled with a variable x_i or a negated variable $\neg x_i$ ($1 \leq i \leq n$), with the proposition α. Moreover, we set $\mathsf{in} = \mathsf{in}(v_0)$ and $\mathsf{out} = \mathsf{out}(v_0)$, where v_0 is the root of S_F. Let us finally add transitions as follows. Let $v \in S_F$ with children v_1 and v_2. If v is labeled with $\vee$, then we add the transitions

$$(\mathsf{in}(v), (y = 0), \emptyset, \mathsf{in}(v_i)) \text{ and } (\mathsf{out}(v_i), (y = 0), \emptyset, \mathsf{out}(v)) \text{ for } i \in \{1, 2\}.$$

If v is labeled with $\wedge$, then we add the transitions

$$(\mathsf{in}(v), (y = 0), \emptyset, \mathsf{in}(v_1)), (\mathsf{out}(v_1), (y = 0), \emptyset, \mathsf{in}(v_2)), \text{ and } (\mathsf{out}(v_2), (y = 0), \emptyset, \mathsf{out}(v)).$$

For every leaf $v \in S_F$ that is labeled with x_i ($1 \leq i \leq n$) we add the transitions

$$(\mathsf{in}(v), (y = 0), \emptyset, \mathsf{out}(v)) \text{ and } (\mathsf{in}(v), (y = 0), \emptyset, \mathsf{in}_{n,i,1}).$$

For every leaf $v \in S_F$ that is labeled with $\neg x_i$ ($1 \leq i \leq n$) we add the transitions

$$(\mathsf{in}(v), (y = 0), \emptyset, \mathsf{out}(v)), \text{ and } (\mathsf{in}(v), (y = 0), \emptyset, \mathsf{in}_{n,0,1}).$$

This concludes the description of the TA $\mathcal{A}(F)$. Correctness of the construction can be easily checked by induction on the structure of the formula F. □

Theorem 28. *There exists a fixed* CTL *formula of the form* $\mathsf{E}\varphi_1\mathsf{U}\varphi_2$*, where* φ_1 *and* φ_2 *are* EF *formulas, such that the following problem is* PSPACE*-complete:*
INPUT: A 2-clock TA $\mathcal{A}$ *and a control location* q *of* $\mathcal{A}$.
QUESTION: $(\mathcal{T}(\mathcal{A}), (q, t_0)) \models \mathsf{E}\varphi_1\mathsf{U}\varphi_2$*?*

Proof. The proof is analogous to the proof of Theorem 18 but simpler, since we do not need to work with CRR-representations of natural numbers. Let us take an arbitrary PSPACE-complete language L. Fix the polynomial $p(n)$ and the NFA $A = (S, \{0,1\}, \delta, s_0, S_f)$ over the alphabet $\{0,1\}$ from Proposition 16. Let $x \in \{0,1\}^*$ be an input of length n, and let $m := p(n)$, where w.l.o.g. $m > 1$. By Proposition 16 one can construct in space $\log n$ a boolean formula with propositional variables x_i ($i \in [m]$) such that:

$$x \in L \iff \prod_{M=0}^{2^m-1} F(\mathsf{BIN}_m(M)) \in L(A). \tag{10}$$

[5] In contrast to the proof of Proposition 17 (where we work with the CRR-representation) we cannot eliminate negations in front of propositional variables x_i here.

We now apply our construction from Proposition 27 to the formula F. For every 1-labeled transition $\tau \in \delta$ of the NFA A let $\mathcal{A}(\tau)$ be a copy of the TA $\mathcal{A}(F)$. For every 0-labeled transition $\tau \in \delta$ let $\mathcal{A}(\tau)$ be a copy of the TA $\mathcal{A}(\neg F)$. In both cases we write $\mathcal{A}(\tau)$ as $(Q(\tau), \{Q_\alpha(\tau), Q_\beta(\tau), Q_\gamma(\tau)\}, \{x, y\}, \delta(\tau))$. Denote with $\mathsf{in}(\tau)$ (resp. $\mathsf{out}(\tau)$) the control location of this copy that corresponds to in (resp. out) in $\mathcal{A}(F)$. Thus, for a b-labeled transition $\tau \in \delta$ ($b \in \{0, 1\}$) and $0 \leq M \leq 2^m - 1$ there exists a $[\![\varphi]\!]_{\mathcal{T}(\mathcal{A}(\tau))}$-path ($\varphi$ is from Proposition 27) from $(\mathsf{in}(\tau), t_M)$ to $(\mathsf{out}(\tau), t_M)$ if and only if $F(\mathsf{BIN}_m(M)) = b$.

We define a 2-clock TA $\mathcal{A}$ as follows: Take the disjoint union of all TAs $\mathcal{A}(\tau)$ for $\tau \in \delta$ and add all states $s \in S$ of the NFA A as well as the special control location final to the set of control locations. Moreover, we add to the set of transitions of $\mathcal{A}$ for every NFA-transition $\tau = (s, b, s') \in \delta$ the transitions

$$(s, (x \leq 2^m - 1 \wedge y = 1), \{y\}, \mathsf{in}(\tau)),\ (\mathsf{out}(\tau), (y = 0), \emptyset, s').$$

Then, by Proposition 27 and (10) we have $x \in L$ if and only if there exists a $[\![\varphi]\!]_{\mathcal{T}(\mathcal{A})}$-path in $\mathcal{T}(\mathcal{A})$ from (s_0, t_0) to (s, t_{2^m-1}) for some $s \in S_f$. Finally, introduce a new proposition ρ, let final be the only ρ-labeled control location, and add for every $q \in S_f$ the transition

$$(q, (x = 2^m - 1 \wedge y = 0), \emptyset, \mathsf{final}).$$

By putting $q = s_0$ we obtain:

$$x \in L \iff (\mathcal{T}(\mathcal{A}), (q, t_0)) \models \mathsf{E}\ \underbrace{(\alpha \rightarrow \mathsf{EX}(\beta \wedge \mathsf{EF}(\gamma)))}_{\varphi \text{ from Proposition 27}}\ \mathsf{U}\ \rho.$$

This concludes the proof of the theorem. □

10.2 Reachability of timed automata with modulo tests

The final application of our lower bound technique concerns the control location reachability problem of timed automata with two clocks but very simple modulo tests. The expressiveness of timed automata with periodic clock constraints has already been studied in [12]. We refer to [27], where it has been shown that the control location reachability problem (or equivalently the emptiness problem) for 2-clock timed automata with modulo tests is PSPACE-hard (and in fact PSPACE-complete). However, the lower bound construction in [27] heavily requires the constants appearing in the clock constraints to be presented in *binary*.

The set $\mathrm{Mod}(C)$ of *modulo clock constraints* over a set of clocks C is the set of boolean formulas with atomic formulas of the form $c \equiv k \bmod \ell$ and $c \sim k$, where $c \in C$, $k, \ell \in \mathbb{N}$ and $\sim \in \{\leq, \geq\}$. A *modulo timed automaton (MTA)* is a tuple $\mathcal{A} = (Q, \{Q_p \mid p \in \mathcal{P}\}, C, \delta)$, where everything is the same as for timed automata, but where $\delta \subseteq Q \times \mathrm{Mod}(C) \times 2^C \times Q$. The *size* $|\mathcal{A}|$ of an MTA $\mathcal{A}$ is defined in analogy to TA. A clock valuation $t : C \to \mathbb{R}_+$ *satisfies* a modulo constraint of the form $c \equiv k \bmod \ell$ whenever $\lfloor t(c) \rfloor \equiv k \bmod \ell$, where for each $r \in \mathbb{R}_+$ we define $\lfloor r \rfloor$ to be the largest non-negative integer n such that $n \leq r$. The transition system $\mathcal{T}(\mathcal{A})$ of an MTA $\mathcal{A}$ is defined analogously as for timed automata (by taking into account the above definition when a clock valuation satisfies a modulo constraint).

We only sketch the proof of the following theorem.

Theorem 29. *The following problem is* PSPACE*-hard, even if all constants that occur in the input are given in unary:*
INPUT: An MTA $\mathcal{A} = (Q, \{Q_p \mid p \in \mathcal{P}\}, C, \delta)$ with only two clocks x and y and two distinguished control locations $q_0, q_1 \in Q$ such that every transition $(q, \gamma, R, q') \in \delta$ satisfies

- *γ does not contain any atomic formulas of the form $x \sim k$,*
- *$x \notin R$ (i.e. x is never reset),*
- *γ does not contain any atomic formulas of the form $y \equiv k$ mod ℓ, and*
- *if $y \sim k$ is an atomic formula in γ, then $k = 1$.*

QUESTION: Does $(q_0, t_0) \to^* (q_1, t)$ hold for some clock valuation $t \in \mathbb{R}_+^{\{x,y\}}$ in $\mathcal{T}(\mathcal{A})$?

Proof sketch. The theorem is proven very similar as Theorem 18. Since we aim at presenting all constants of the MTA in unary, we use the Chinese remainder representation of the current number $M \in [0, 2^m - 1]$. We represent the current value of M by the clock x. The clock y is only used to increment x by one. Instead of using the formula φ from Proposition 17 to control the Chinese remainder representation of M, we directly test x by using modulo tests in the transitions of the MTA. □

References

1. R. Alur and D.L. Dill. A theory of timed automata *Theoretical Computer Science*, 126(2):183–235, 1994.
2. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambrdige University Press, 2009.
3. C. Baier and J. P. Katoen. *Principles of Model Checking*. MIT Press, 2009.
4. V. I. Bogachev. *Measure theory*. Springer, 2006.
5. P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.
6. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR'97)*, number 1243 in Lecture Notes in Computer Science, pages 135–150. Springer, 1997.
7. P. Bouyer and F. Laroussinie. Model checking timed automata. In S. Merz and N. Navet, editors, *Modeling and Verification of Real-Time Systems*, pages 111–140. ISTE Ltd. – John Wiley & Sons, Ltd., 2008.
8. T. Brázdil, V. Brozek, K. Etessami, A. Kucera, and D. Wojtczak. One-counter markov decision processes. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 863–874. SIAM, 2010.
9. T. Cachat. Uniform solution of parity games on prefix-recognizable graphs. *Electronic Notes in Theoretical Computer Science*, 68(6), 2002.
10. J.-Y. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Science*, 2(1):67–76, 1991.
11. A. Chiu, G. Davida, and B. Litow. Division in logspace-uniform NC^1. *RAIRO - Theoretical Informatics and Applications. Informatique Théorique et Applications*, 35(3):259–275, 2001.
12. C. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5(4):371–404, 2000.
13. C. Courcoubetis and M. Yannakakis. Minimum and Maximum Delay Problems in Real-Time Systems. *Formal Methods in System Design*, 1(4):385–415, 1992.
14. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*, number 1855 in Lecture Notes in Computer Science, pages 232–247. Springer, 2000.
15. K. Etessami and M. Yannakakis. Recursive markov decision processes and recursive stochastic games. In *Proceesing of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*, number 3580 in Lecture Notes in Computer Science, pages 891–903, 2005.
16. S. Göller and M. Lohrey. Infinite state model-checking of propositional dynamic logics. In *Proceedings of the 20th International Conference on Computer Science Logic (CSL 2006)*, number 4207 in Lecture Notes in Computer Science, pages 349–364. Springer, 2006.
17. S. Göller and M. Lohrey. Branching-time model checking of one-counter processes. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, volume 5 of *LIPIcs*, pages 405–416. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
18. S. Göller, R. Mayr, and A. W. To. On the computational complexity of verifying one-counter processes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, pages 235–244. IEEE Computer Society Press, 2008.
19. C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *Proceedings of the 20th International Conference on Concurrency Theory (CONCUR09)*, number 5710 in Lecture Notes in Computer Science, pages 369–383. Springer, 2009.
20. C. Haase, J. Ouaknine, and J. Worrell. On the Relationship between Reachability Problems in Timed and Counter Automata. In *Proceedings of the 6th International Workshop on Reachability Problems (RP 2012)*, number 7550 in Lecture Notes in Computer Science, pages 54–65. Springer, 2012.
21. U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 200–207. IEEE Computer Society Press, 1993.
22. W. Hesse, E. Allender, and D. A. M. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65:695–716, 2002.

23. M. Holzer. On emptiness and counting for alternating finite automata. In *Proceedings of the 2nd International Conference on Developments in Language Theory (DLT 1995)*, pages 88–97. World Scientific, 1996.
24. P. Jančar and Z. Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Information Processing Letters*, 104(5):164–167, 2007.
25. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking timed automata with one or two clocks. *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR'04)*, number 3170 in Lecture Notes in Computer Science, pages 387–401. Springer, 2004.
26. M. Nair. On Chebyshev-type inequalities for primes. *The American Mathematical Monthly*, 89(2):126–129, 1982.
27. G. Naves. Accessibilité dans les automates temporisé à deux horloges. *Memoire de Master 2*, ENS Cachan (France), 2006.
28. N. Piterman and M. Y. Vardi. Global model-checking of infinite-state systems. In *Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004)*, number 3114 in Lecture Notes in Computer Science, pages 387–400. Springer, 2004.
29. O. Serre. Note on winning positions on pushdown games with ω-regular conditions. *Inf. Process. Lett.*, 85(6):285–291, 2003.
30. O. Serre. Parity games played on transition graphs of one-counter processes. In *Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2006)*, number 3921 in Lecture Notes in Computer Science. Springer, 2006.
31. D. Thérien and T. Wilke. Temporal logic and semidirect products: An effective characterization of the until hierarchy. In *In Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, pages 256–263. IEEE Computer Society Press, 1996.
32. A. W. To. Model checking FO(R) over one-counter processes and beyond. In *Proceedings of the 23rd International Conference on Computer Science Logic (CSL 2009)*, number 5771 in Lecture Notes in Computer Science, pages 485–499. Springer Verlag, 2009.
33. H. Vollmer. A generalized quantifier concept in computational complexity theory. Technical report, arXiv.org, 1998. http://arxiv.org/abs/cs.CC/9809115.
34. H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.
35. K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51:53–80, 1987.
36. I. Walukiewicz. Model checking CTL properties of pushdown systems. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS 2000)*, number 1974 in Lecture Notes in Computer Science, pages 127–138. Springer, 2000.
37. I. Walukiewicz. Pushdown processes: games and model-checking. *Information and Computation*, 164(2):234–263, 2001.

Appendix

Let M be a nondeterministic Turing machine with a linear ordering on the set of all transition tuples. Assume furthermore that M does not contain infinite computation paths. Then, for every input x, the computation tree $T(x)$ of the machine M on input x is a finite ordered tree. Let $v_1, v_2, \ldots, v_n$ be a list of all leafs of $T(x)$ in left-to-right enumeration. Then the *leaf string* $\mathsf{leaf}(M, x)$ is the string $a_1 a_2 \cdots a_n$, where $a_i = 1$ (resp. $a_i = 0$) if v_i is an accepting (resp. rejecting) configuration.

Theorem 30. *Let A be a language in* PSPACE. *Then A is* AC^0*-serializable, i.e., there exists a regular language $L \subseteq \{0,1\}^*$, a polynomial $p(n)$, and a logspace-uniform* AC^0*-circuit family $(B_n)_{n \geq 0}$, where B_n has exactly $n + p(n)$ many inputs and one output, such that for every $x \in \{0,1\}^n$ we have:*

$$x \in A \iff B_n(x, 0^{p(n)}) \cdots B_n(x, 1^{p(n)}) \in L,$$

where "$\cdots$" refers to the lexicographic order on $\{0,1\}^{p(n)}$.

Proof. Let $A \subseteq \{0,1\}^*$ be a language in PSPACE. By the work of [21] there exists a nondeterministic polynomial time Turing machine

$$M = (Q, \Gamma, \Delta, q_0, q_f, \Box)$$

and a regular language $K \subseteq \{0,1\}^*$ such that

$$x \in A \iff \mathsf{leaf}(M, x) \in K. \tag{11}$$

Here, Q is the set of states, Γ is the tape alphabet, $\Delta \subseteq Q \times \Gamma \times Q \times \Gamma \cup \{\rightarrow, \leftarrow\}$ is the set of transition tuples, q_0 is the initial state, q_f is the final (accepting) state, and $\Box$ is the blank symbol. W.l.o.g. we can assume that every computation path of M on an input of length n has length $q(n)$ for a polynomial q. This can be enforced by introducing a counter. Note that the counter can be incremented deterministically, hence the produced leaf string does not change. Assume that $\Delta = \{\delta_1, \ldots, \delta_m\}$, where $\delta_1 < \delta_1 < \cdots < \delta_m$ is the fixed order on the transition tuples of M.

Let $\Omega = Q \cup \Gamma \cup \Delta$, where all three sets are assumed to be pairwise disjoint. We will encode a computation of M of length $q(n)$, starting on input $x \in \Sigma^n$, by a word from the language

$$\begin{aligned} C(x) = \{ & c_0 t_1 c_1 t_2 \cdots c_{q(n)-1} t_{q(n)} c_{q(n)} \mid t_1, \ldots, t_{q(n)} \in \Delta \\ & c_0 = q_0 x \Box^{q(n)-n},\ c_1, \ldots, c_{q(n)} \in \Gamma^* Q \Gamma^+, \\ & |c_1| = \cdots = |c_{q(n)}| = q(n)+1,\ \forall 0 \leq i < q(n) : c_i \vdash_{t_{i+1}} c_{i+1} \}. \end{aligned}$$

Here, $c_i \vdash_{t_{i+1}} c_{i+1}$ means that configuration c_{i+1} results from configuration c_i by applying transition t_{i+1}. Let $D(x)$ be the subset of $C(x)$ consisting of all successful computations $c_0 t_1 c_1 t_2 \cdots c_{q(n)-1} t_{q(n)} c_{q(n)} \in C(x)$, where in addition $c_{q(n)} \in \Gamma^* q_f \Gamma^+$.

Note that every word in $C(x)$ has length $(q(n)+1)^2 + q(n)$. We use some block encoding $\gamma : \Omega \rightarrow \{0,1\}^k$ such that $\gamma(\delta_{i+1})$ is lexicographically larger than $\gamma(\delta_i)$ for $i \in [m-1]$. This ensures that if we list all bit strings of length $k \cdot ((q(n)+1)^2 + q(n))$ in lexicographic order than the subset $C(x)$ of all (encodings of) valid computations appears as a subsequence in the same order as in the computation tree $T(x)$.

Let us next describe a logspace-uniform AC^0-circuit family $(C_n)_{n \geq 0}$, where the n^{th} circuit C_n has $n + k \cdot ((q(n)+1)^2 + q(n))$ many inputs and accepts exactly all strings of the form xw, where $x \in \{0,1\}^n$ and $w \in C(x)$. Constructing C_n is tedious but straightforward. The most difficult part is to check $c_i \vdash_{t_{i+1}} c_{i+1}$ for all $0 \leq i < q(n)$. For this, we use an AND-gate g with $q(n)$ many children $g_0, \ldots, g_{q(n)-1}$. Gate g_i is an OR-gate with $q(n)$ many children $g_{i,1}, \ldots, g_{i,q(n)}$. Gate $g_{i,j}$ evaluates to 1 if and only if c_{i+1} results from c_i by applying the transition t_{i+1} at position j. To achieve this, $g_{i,j}$ becomes an AND-gate with $k(q(n)+1)$ many input gates. Each of these gates compares two corresponding bits in the γ-encodings of c_i and c_{i+1}. It should be clear that such a circuit C_n can be built in logarithmic space. Analogously we can construct a logspace-uniform AC^0-circuit family $(D_n)_{n \geq 0}$ which accepts all strings of the form xw, where $x \in \{0,1\}^n$ and $w \in D(x)$.

Finally, we construct from the two families $(C_n)_{n \geq 0}$ and $(D_n)_{n \geq 0}$ a new logspace-uniform AC^0-circuit family $(B_n)_{n \geq 0}$, where B_n has $n + k \cdot ((q(n)+1)^2 + q(n)) + 1$ many inputs. On input $xw0$ (with $x \in \Sigma^n$) it outputs $C_n(xw)$. On input $xw1$, B_n outputs $D_n(xw)$. Now, let us construct from the regular language $K \subseteq \{0,1\}^*$ the new regular language $L = \varphi(K \sqcup\!\sqcup \{a\}^*)$, where $\sqcup\!\sqcup$ is the shuffle operator, $a \notin \{0,1\}$ is a new symbol, and φ is the homomorphism with $\varphi(a) = 00$, $\varphi(0) = 10$, $\varphi(1) = 11$.

The regular language L, the polynomial $p(n) = k \cdot ((q(n)+1)^2 + q(n)) + 1$, and the circuit family $(B_n)_{n \geq 0}$ fulfill the requirements from the theorem. $\square$