Logical Methods in Computer Science
Volume 15, Issue 3, 2019, pp. 33:1–33:26
https://lmcs.episciences.org/

Submitted Jul. 03, 2018
Published Sep. 30, 2019

THE COMPLEXITY OF FLAT FREEZE LTL [*]

BENEDIKT BOLLIG [a], KARIN QUAAS [b], AND ARNAUD SANGNIER [c]

[a] CNRS, LSV, ENS Paris-Saclay, Université Paris-Saclay
e-mail address: bollig@lsv.fr

[b] Universität Leipzig
e-mail address: quaas@informatik.uni-leipzig.de

[c] IRIF, Université Paris Diderot
e-mail address: sangnier@irif.fr

Abstract. We consider the model-checking problem for freeze LTL on one-counter automata (OCA). Freeze LTL extends LTL with the freeze quantifier, which allows one to store different counter values of a run in registers so that they can be compared with one another. As the model-checking problem is undecidable in general, we focus on the flat fragment of freeze LTL, in which the usage of the freeze quantifier is restricted. In a previous work, Lechner et al. showed that model checking for flat freeze LTL on OCA with binary encoding of counter updates is decidable and in 2NEXPTIME. In this paper, we prove that the problem is, in fact, NEXPTIME-complete no matter whether counter updates are encoded in unary or binary. Like Lechner et al., we rely on a reduction to the reachability problem in OCA with parameterized tests (OCA(P)). The new aspect is that we simulate OCA(P) by alternating two-way automata over words. This implies an exponential upper bound on the parameter values that we exploit towards an NP algorithm for reachability in OCA(P) with unary updates. We obtain our main result as a corollary. As another application, relying on a reduction by Bundala and Ouaknine, one obtains an alternative proof of the known fact that reachability in closed parametric timed automata with one parametric clock is in NEXPTIME.

1. Introduction

One-counter automata (OCA) are a simple yet fundamental computational model operating on a single counter that ranges over the non-negative integers. Albeit being a classical model, OCA are in the focus of ongoing research in the verification community (*cf.*, for example, [HKOW09, GHOW12, LMO$^+$16, DG09, GHOW10, GL13, HLMT16]). A large body of work is devoted to model checking OCA, i.e., the question whether all runs of a given OCA satisfy a temporal-logic specification. A natural way to model runs of OCA is in terms of *data words*, i.e., words where every position carries two pieces of information: a set of propositions from a finite alphabet, and a datum from an infinite alphabet. In

Key words and phrases: one-counter automata, data words, freeze LTL, model checking.
[*] This work has been partly supported by the ANR research program PACS (ANR–14–CE28–0002) and by DFG, project QU 316/1–2. A preliminary version of this paper appeared as [BQS17].

our case, the datum represents the current counter value. Now, reasoning about the sequence of propositions that is produced by a run amounts to model checking OCA against classical temporal logics like linear-time temporal logic (LTL) and computation-tree logic (CTL) [GL13, GHOW12, GHOW10, Haa12]. But it is natural to also reason about the unboundedly many counter values that may occur. Several formalisms have been introduced that can handle infinite alphabets, including variants or extensions of monadic second-order logic, LTL, and CTL [DL09, DLS10, BDM$^+$11, FLQ15]. Freeze LTL is an extension of LTL that allows one to remember, in terms of the *freeze quantifier*, certain counter values for later comparison in a run of the OCA under consideration (*cf.* [Fre03, LP05, DLN07, DL09]). Unfortunately, satisfiability and model checking OCA against freeze LTL are undecidable [DL09, DLS10].

In this paper, we study model checking of OCA against formulas in *flat freeze LTL*, a fragment of freeze LTL that restricts the freeze quantifier, but allows for unlimited usage of comparisons. A typical property definable in flat freeze LTL is "there exists a counter value that occurs infinitely often". Moreover, the negation of many natural freeze LTL specifications can be expressed in flat freeze LTL . The approach of restricting the syntax of a temporal logic to its flat fragment has also been pursued in [CC00] for Constraint LTL, and in [BMOW08] for MTL.

Demri and Sangnier [DS10] reduced model checking OCA against flat freeze LTL to reachability in OCA *with parameterized tests* (OCA(P)). In an OCA(P), counter values may be compared with parameters whose values are arbitrary but fixed. Reachability asks whether a given state can be reached under *some* parameter instantiation. Essentially, the translation of a flat freeze LTL formula into an OCA(P) interprets every freeze quantifier as a parameter, whose value can be compared with counter values arbitrarily often. Decidability of the reachability problem for OCA(P), however, was left open. Recently, Lechner et al. proved decidability by a reduction to satisfiability in Presburger arithmetic [LMO$^+$16]. As a corollary, they obtain a 2NEXPTIME upper bound for model checking OCA against formulas in flat freeze LTL, assuming that counter updates in OCA are encoded in binary.

Our main technical contribution is an improvement of the result by Lechner et al. We proceed in two main steps. First, we show that reachability for OCA(P) (with unary counter updates) can be reduced to non-emptiness of alternating two-way (finite) automata. Interestingly, alternating two-way automata have already been used as an algorithmic framework for game-based versions of pushdown processes [KV00, Cac02], one-counter processes [Ser06], and timed systems [ABG$^+$14]. Our link already implies decidability of both reachability for OCA(P) (in PSPACE when counter updates are unary) and model checking OCA against flat freeze LTL (in EXPSPACE). However, we can go further. First, we deduce an exponential upper bound on the largest parameter value needed for an accepting run in the given OCA(P). Exploiting this bound and a technique by Galil [Gal76], we show that, actually, reachability for OCA(P) is in NP. As a corollary, we obtain a NEXPTIME upper bound for model checking OCA against flat freeze LTL . Using a result from [GHOW10], we can also show NEXPTIME-hardness, which establishes the precise complexity of the model-checking problem. Our result applies no matter whether counter updates in the given OCA are encoded in unary or binary.

Outline. In Section 2, we define OCA, (flat) freeze LTL, and the corresponding model-checking problems. Section 3 is devoted to reachability in OCA(P), which is at the heart of our model-checking procedures. The reduction of model checking to reachability in OCA(P) is given in Section 4, where we also present lower bounds. In Section 5, we consider the *universal* model-checking problem. Finally, in Section 6, we apply our results to show that reachability in parametric timed automata with one parametric clock (1PTA) and closed guards (which corresponds to the discrete-time case) is in $\mathsf{NEXPTIME}$, which matches the known lower bound. Note that this upper bound had already been shown in [BBLS15] for general 1PTA (over both discrete time and continuous time), using a different proof technique. We conclude in Section 7.

2. Preliminaries

2.1. One-Counter Automata with Parameterized Tests.

2.1. One-Counter Automata with Parameterized Tests. We start by defining one-counter automata *with all extras* such as succinct encodings of updates, parameters, and comparisons with constants. Well-known subclasses are then identified as special cases.

For the rest of this paper, we fix a countably infinite set $\mathbb{P}$ of *propositions*, which will label states of a one-counter automaton. Transitions of an automaton may perform tests to compare the current counter value with zero, with a parameter, or with a constant.

Definition 2.1. A *one-counter automaton with succinct updates, parameterized tests, and comparisons with constants*, OCA(S,P,C) for short, is a tuple $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta, \mu)$ where

- Q is a finite set of *states*,
- $\mathcal{X}$ is a finite set of *parameters* ranging over $\mathbb{N}$,
- $q_{\mathsf{in}} \in Q$ is the *initial state*,
- $\Delta = \Delta_{\mathsf{upd}} \uplus \Delta_{\mathsf{param}} \uplus \Delta_{\mathsf{const}}$ is the finite set of *transitions*, which is partitioned into
  - $\Delta_{\mathsf{upd}} \subseteq Q \times \mathbb{Z} \times Q$,
  - $\Delta_{\mathsf{param}} \subseteq Q \times \{<x\,, =x\,, >x \mid x \in \mathcal{X}\} \times Q$,
  - $\Delta_{\mathsf{const}} \subseteq Q \times \{<c\,, =c\,, >c \mid c \in \mathbb{N}\} \times Q$, and
- $\mu : Q \to 2^{\mathbb{P}}$ maps each state to a *finite* set of propositions.

We define the size of $\mathcal{A}$ as

$$|\mathcal{A}| = \left(\begin{array}{cc} & |Q| + |\mathcal{X}| + |\Delta| + \sum_{q \in Q} |\mu(q)| \\ + & \sum\{\log(|z|) \mid (q, z, q') \in \Delta_{\mathsf{upd}} \text{ with } z \neq 0\}\} \\ + & \sum\{\log(c) \mid (q, \bowtie c, q') \in \Delta_{\mathsf{const}} \text{ with } c > 0\} \end{array} \right).$$

Let $\mathcal{C}_{\mathcal{A}} := Q \times \mathbb{N}$ be the set of *configurations* of $\mathcal{A}$. In a configuration $(q, v) \in \mathcal{C}_{\mathcal{A}}$, the first component q is the current state and v is the current counter value, which is always non-negative. The semantics of $\mathcal{A}$ is given w.r.t. a *parameter instantiation* $\gamma : \mathcal{X} \to \mathbb{N}$ in terms of a global transition relation $\longrightarrow_\gamma \subseteq \mathcal{C}_{\mathcal{A}} \times \mathcal{C}_{\mathcal{A}}$. For two configurations (q, v) and (q', v'), we have $(q, v) \longrightarrow_\gamma (q', v')$ if there is $(q, \mathsf{op}, q') \in \Delta$ such that one of the following holds:

- $\mathsf{op} \in \mathbb{Z}$ and $v' = v + \mathsf{op}$,
- $\mathsf{op} = \bowtie x$ and $v = v'$ and $v \bowtie \gamma(x)$, for some $x \in \mathcal{X}$ and $\bowtie \in \{<, =, >\}$, or
- $\mathsf{op} = \bowtie c$ and $v = v'$ and $v \bowtie c$, for some $c \in \mathbb{N}$ and $\bowtie \in \{<, =, >\}$.

Thus, transitions of an OCA(S,P,C) either increment/decrement the counter by a value $z \in \mathbb{Z}$, or compare the counter value with a parameter value or a constant. Note that, in this model, the counter values are always positive. Moreover, we assume a binary encoding of constants and updates.

A *γ-run* of $\mathcal{A}$ is a finite or infinite sequence $\rho = (q_0, v_0) \longrightarrow_\gamma (q_1, v_1) \longrightarrow_\gamma \cdots$ of global transitions (a run may consist of one single configuration (q_0, v_0)). We say that ρ is *initialized* if $q_0 = q_{\mathsf{in}}$ and $v_0 = 0$.

We identify some well-known special cases of the general model. Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta, \mu)$ be an OCA(S,P,C). We say that

- $\mathcal{A}$ is an OCA(P,C) if $\Delta_{\mathsf{upd}} \subseteq Q \times \{+1, 0, -1\} \times Q$ (i.e., counter updates are *unary*);
- $\mathcal{A}$ is an OCA(S,P) if $\Delta_{\mathsf{const}} \subseteq Q \times \{=0\} \times Q$ (i.e., only comparisons with 0 are allowed);
- $\mathcal{A}$ is an OCA(P) if $\Delta_{\mathsf{upd}} \subseteq Q \times \{+1, 0, -1\} \times Q$ and $\Delta_{\mathsf{const}} \subseteq Q \times \{=0\} \times Q$ (i.e., counter updates are unary and only comparisons with 0 are allowed);
- $\mathcal{A}$ is an OCA(S) if $\mathcal{X} = \emptyset$ (i.e., $\Delta_{\mathsf{param}} = \emptyset$) and $\Delta_{\mathsf{const}} \subseteq Q \times \{=0\} \times Q$ (i.e., there is no parameter and only comparisons with 0 are allowed);
- $\mathcal{A}$ is an OCA if $\mathcal{X} = \emptyset$ and, moreover, all transition labels are among $\{+1, 0, -1\} \cup \{=0\}$ (i.e., there is no parameter, counter updates are *unary* and only comparisons with 0 are allowed).

We may omit $\mathcal{X}$ when it is equal to $\emptyset$ (i.e., in the case of an OCA(S) or OCA) and simply refer to $(Q, q_{\mathsf{in}}, \Delta, \mu)$. Since, in this case, the global transition relation does not depend on a parameter instantiation anymore, we may just write $\longrightarrow$ instead of $\longrightarrow_\gamma$. Similarly, for reachability problems (see below), μ is irrelevant so that it can be omitted, too.

One of the most fundamental decision problems we can consider is whether a given state $q_f \in Q$ is reachable. Given some parameter instantiation γ, we say that q_f is *γ-reachable* if there is an initialized run $(q_0, v_0) \longrightarrow_\gamma \cdots \longrightarrow_\gamma (q_n, v_n)$ such that $q_n = q_f$. Moreover, q_f is *reachable*, written $\mathcal{A} \models \mathit{Reach}(q_f)$, if it is γ-reachable for some γ. Now, for a class $\mathcal{C} \in \{\text{OCA}(\mathsf{S},\mathsf{P},\mathsf{C}), \text{OCA}(\mathsf{P},\mathsf{C}), \text{OCA}(\mathsf{S},\mathsf{P}), \text{OCA}(\mathsf{P}), \text{OCA}(\mathsf{S}), \text{OCA}\}$, the *reachability problem* is defined as follows:

$\mathcal{C}$-Reachability	
Input:	$\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta) \in \mathcal{C}$ and $q_f \in Q$
Question:	Do we have $\mathcal{A} \models \mathit{Reach}(q_f)$?

Towards the *Büchi problem* (or *repeated reachability*), we write $\mathcal{A} \models \mathit{Reach}^\omega(q_f)$ if there are a parameter instantiation γ and an infinite initialized γ-run $(q_0, v_0) \longrightarrow_\gamma (q_1, v_1) \longrightarrow_\gamma \cdots$ such that $q_i = q_f$ for infinitely many $i \in \mathbb{N}$. The Büchi problem asks whether some state from a given set $F \subseteq Q$ can be visited infinitely often:

$\mathcal{C}$-Büchi	
Input:	$\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta) \in \mathcal{C}$ and $F \subseteq Q$
Question:	Do we have $\mathcal{A} \models \mathit{Reach}^\omega(q_f)$ for some $q_f \in F$?

It is known that OCA-Reachability and OCA-Büchi are NLOGSPACE-complete [VP75, DG09], and that OCA(S)-Reachability and OCA(S)-Büchi are NP-complete [HKOW09, Haa12] (cf. also Table 1).

2.2. **Freeze LTL and its Flat Fragment.** We now define freeze LTL . To do so, we fix a countably infinite supply of registers $\mathcal{R}$.

Definition 2.2 (Freeze LTL)**.** The logic *freeze LTL*, denoted by $\mathrm{LTL}^{\downarrow}$, is given by the grammar

$$\varphi \ ::= \ p \ \mid \ \bowtie r \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \mathsf{X}\varphi \ \mid \ \varphi \,\mathsf{U}\, \varphi \ \mid \ \varphi \,\mathsf{R}\, \varphi \ \mid \ \downarrow_r \varphi$$

where $p \in \mathbb{P}$, $r \in \mathcal{R}$, and $\bowtie \in \{<, =, >\}$.

Note that, apart from the until operator U, we also include the dual release operator R. This is convenient in proofs, as it allows us to assume formulas to be in negation normal form. We also use common abbreviations such as $\varphi \to \psi$ for $\neg\varphi \vee \psi$, $\mathsf{F}\varphi$ for $\mathsf{true}\,\mathsf{U}\,\varphi$ (with $\mathsf{true} = p \vee \neg p$ for some proposition p), and $\mathsf{G}\varphi$ for $\neg\mathsf{F}\neg\varphi$.

We call $\downarrow_r$ the *freeze quantifier*. It stores the current counter value in register r. Atomic formulas of the form $\bowtie r$, called *register tests*, perform a comparison of the current counter value with the contents of r, provided that they are in the scope of a freeze quantifier. Actually, inequalities of the form $< r$ and $> r$ were not present in the original definition of $\mathrm{LTL}^{\downarrow}$, but our techniques will take care of them without extra cost or additional technical complication. Classical LTL is obtained as the fragment of $\mathrm{LTL}^{\downarrow}$ that uses neither the freeze quantifier nor register tests.

A formula $\varphi \in \mathrm{LTL}^{\downarrow}$ is interpreted over an infinite sequence $w = (P_0, v_0)(P_1, v_1) \ldots \in (2^{\mathbb{P}} \times \mathbb{N})^{\omega}$ with respect to a position $i \in \mathbb{N}$ and a register assignment $\nu : \mathcal{R} \to \mathbb{N}$. The satisfaction relation $w, i \models_\nu \varphi$ is inductively defined as follows:

- $w, i \models_\nu p$ if $p \in P_i$,
- $w, i \models_\nu \bowtie r$ if $v_i \bowtie \nu(r)$,
- $w, i \models_\nu \mathsf{X}\varphi$ if $w, i+1 \models_\nu \varphi$,
- $w, i \models_\nu \varphi_1 \,\mathsf{U}\, \varphi_2$ if there is $j \geq i$ such that $w, j \models_\nu \varphi_2$ and $w, k \models_\nu \varphi_1$ for all $k \in \{i, \ldots, j-1\}$,
- $w, i \models_\nu \varphi_1 \,\mathsf{R}\, \varphi_2$ if one of the following holds: (i) $w, k \models_\nu \varphi_2$ for all $k \geq i$, or (ii) there is $j \geq i$ such that $w, j \models_\nu \varphi_1$ and $w, k \models_\nu \varphi_2$ for all $k \in \{i, \ldots, j\}$,
- $w, i \models_\nu \downarrow_r \varphi$ if $w, i \models_{\nu[r \mapsto v_i]} \varphi$, where the register assignment $\nu[r \mapsto v_i]$ maps register r to v_i and coincides with ν on all other registers.

Negation, disjunction, and conjunction are interpreted as usual.

A formula φ is a *sentence* when every subformula of φ of the form $\bowtie r$ is in the scope of a freeze quantifier $\downarrow_r$. In that case, the initial register assignment is irrelevant, and we simply write $w \models \varphi$ if $w, 0 \models_\nu \varphi$ (with ν arbitrary). Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta, \mu)$ be an OCA(S,P,C) and let $\rho = (q_0, v_0) \longrightarrow_\gamma (q_1, v_1) \longrightarrow_\gamma \cdots$ be an infinite run of $\mathcal{A}$. We say that ρ satisfies φ, written $\rho \models \varphi$, if $(\mu(q_0), v_0)(\mu(q_1), v_1) \ldots \models \varphi$. Moreover, we write $\mathcal{A} \models_\exists \varphi$ if there *exist* γ and an infinite initialized γ-run ρ of $\mathcal{A}$ such that $\rho \models \varphi$.

Model checking for a class $\mathcal{C}$ of OCA(S,P,C) and a logic $\mathcal{L} \subseteq \mathrm{LTL}^{\downarrow}$ is defined as follows:

$\mathcal{C}$-MC($\mathcal{L}$)

Input:	$\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta, \mu) \in \mathcal{C}$ and a sentence $\varphi \in \mathcal{L}$
Question:	Do we have $\mathcal{A} \models_\exists \varphi$?

	Reachability / Büchi	MC(LTL)	MC($\flat$LTL$^{\downarrow}$)
OCA	NLOGSPACE-complete [VP75] / [DG09]	PSPACE-complete (e.g., [Haa12])	NEXPTIME-complete Theorem 4.9
OCA(S)	NP-complete [HKOW09] / [Haa12]	PSPACE-complete [GHOW10]	NEXPTIME-complete Theorem 4.9

Table 1: Old and new results

Note that, following [DLS10, LMO^{+}16], we study the existential version of the model-checking problem ("Is there some run satisfying the formula?"). The reason is that the flat fragment that we consider next is not closed under negation, and for many useful freeze LTL formulas the negation is actually *flat* (cf. Example 2.4 below). In Section 5, we will study the *universal* version of the model-checking problem ("Do all runs satisfy the formula?").

Unfortunately, OCA-MC(LTL$^{\downarrow}$) is undecidable [DLS10], even if formulas use only one register. This motivates the study of the flat fragment of LTL$^{\downarrow}$, restricting the usage of the freeze quantifier [DLS10]. Essentially, it is no longer possible to overwrite a register unboundedly often.

Definition 2.3 (Flat Freeze LTL)**.** The *flat* fragment of LTL$^{\downarrow}$, denoted by $\flat$LTL$^{\downarrow}$, contains a formula φ if, for every occurrence of a subformula $\psi = \varphi_1 \, \mathsf{U} \, \varphi_2$ (respectively, $\psi = \varphi_2 \, \mathsf{R} \, \varphi_1$) in φ, the following hold: (i) If the occurrence of ψ is in the scope of an even number of negations, then the freeze quantifier does not occur in φ_1, and (ii) if it is in the scope of an odd number of negations, then the freeze quantifier does not occur in φ_2.

Example 2.4. An example $\flat$LTL$^{\downarrow}$ formula is $\mathsf{F}\downarrow_r \mathsf{G}\big((<r) \vee (=r)\big)$, saying that a run of an OCA takes only finitely many different counter values. On the other hand, the formula $\varphi = \mathsf{G}\downarrow_r\big(\mathit{req} \rightarrow \mathsf{F}(\mathit{serve} \wedge (=r))\big)$ (from [LMO^{+}16]), which says that every request is eventually served with the same ticket, is *not* in $\flat$LTL$^{\downarrow}$, but its negation $\neg\neg\big(\mathsf{true}\,\mathsf{U}\,\neg\downarrow_r\big(\mathit{req} \rightarrow \mathsf{F}(\mathit{serve} \wedge (=r))\big)\big)$ is. Indeed, the until modality lies in the scope of an even number of negations, and the freeze quantifier is in the second argument of the until.

In [DS10], it has been shown that OCA-MC($\flat$LTL$^{\downarrow}$) can be reduced to OCA(P)-Büchi (without comparisons with constants), but decidability of the latter was left open (positive results were only obtained for a restricted fragment of $\flat$LTL$^{\downarrow}$). In [LMO^{+}16], Lechner et al. showed that OCA(P)-Büchi is decidable. In fact, they establish that OCA(S)-MC($\flat$LTL$^{\downarrow}$) is in 2NEXPTIME.

Our main result states that the problems OCA-MC($\flat$LTL$^{\downarrow}$) and OCA(S)-MC($\flat$LTL$^{\downarrow}$) are both NEXPTIME-complete. A comparison with known results can be found in Table 1. The proof outline is depicted in Figure 1. Essentially, we also rely on a reduction of OCA(S)-MC($\flat$LTL$^{\downarrow}$) to OCA(P)-Reachability, and the main challenge is to establish the precise complexity of the latter. In Section 3, we reduce OCA(P)-Reachability to non-emptiness of alternating two-way automata over infinite words, which is then exploited to prove an NP upper bound. The reductions from OCA(S)-MC($\flat$LTL$^{\downarrow}$) to OCA(P)-Reachability themselves are given in Section 4, which also contains our main results.

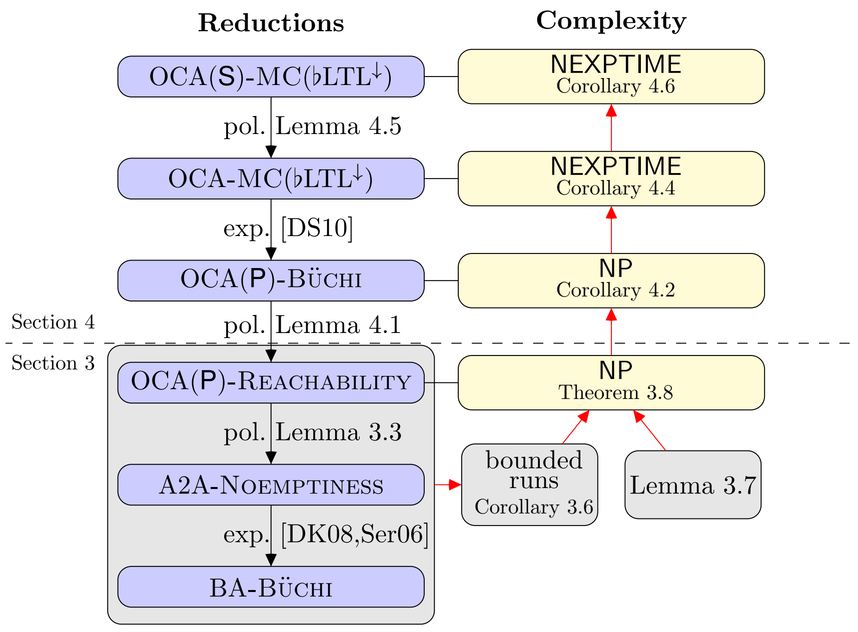

Figure 1: Proof structure for upper bounds

3. OCA(P)-Reachability is NP-complete

In this section, we reduce OCA(P)-Reachability to the non-emptiness problem of alternating two-way automata (A2As) over infinite words. While this already implies that OCA(P)-Reachability is in PSPACE, we then go a step further and show that the problem is in NP.

3.1. From OCA(P) to Alternating Two-Way Automata.

Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ be an OCA(P). The main idea is to encode a parameter instantiation $\gamma : \mathcal{X} \to \mathbb{N}$ as a word over the alphabet $\Sigma = \mathcal{X} \cup \{\Box\}$, where $\Box$ is a fresh symbol. A word $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$, with $a_i \in \Sigma$, is called a *parameter word* (over $\mathcal{X}$) if $a_0 = \Box$ and, for all $x \in \mathcal{X}$, there is exactly one position $i \in \mathbb{N}$ such that $a_i = x$. In other words, w starts with $\Box$ and every parameter occurs exactly once. Then, w determines a parameter instantiation $\gamma_w : \mathcal{X} \to \mathbb{N}$ as follows: if $x = a_i$, then $\gamma_w(x) = |a_1 \cdots a_{i-1}|_\Box$ where $|a_1 \cdots a_{i-1}|_\Box$ denotes the number of occurrences of $\Box$ in $a_1 \cdots a_{i-1}$ (note that we start at the second position of w). For example, given $\mathcal{X} = \{x_1, x_2, x_3\}$, both $w = \Box x_2 \Box\Box x_1 x_3 \Box^\omega$ and $w' = \Box x_2 \Box\Box x_3 x_1 \Box^\omega$ are parameter words with $\gamma_w = \gamma_{w'} = \{x_1 \mapsto 2, x_2 \mapsto 0, x_3 \mapsto 2\}$. Note that, for every parameter instantiation γ, there is at least one parameter word w such that $\gamma_w = \gamma$. Let $\mathcal{W}_\mathcal{X}$ denote the set of all parameter words over $\mathcal{X}$.

From $\mathcal{A}$ and a state $q_f \in Q$, we will build an A2A that accepts the set of parameter words w such that q_f is γ_w-reachable. Like a Turing machine, an A2A can read a letter, change its state, and move its head to the left or to the right (or stay at the current position). In addition, it can spawn several independent copies, for example one that goes to the left and one that goes to the right. However, unlike a Turing machine, an A2A is not allowed to modify the input word so that its expressive power does not go beyond finite (Büchi) automata. The simulation proceeds as follows. When the OCA(P) increments its counter, the A2A moves to the right to the next occurrence of $\Box$. To simulate a decrement, it moves to the left until it encounters the previous $\Box$. To mimic the zero test, it verifies that it is currently on the first position of the word. Moreover, it will make use of the letters in $\mathcal{X}$ to simulate parameter tests. At the beginning of an execution, the A2A spawns independent copies that check whether the input word is a valid parameter word.

Let us define A2As and formalize the simulation of an OCA(P). Given a finite set Y, we denote by $\mathbb{B}^+(Y)$ the set of positive Boolean formulas over Y, including `true` and `false`. A subset $Y' \subseteq Y$ *satisfies* $\beta \in \mathbb{B}^+(Y)$, written $Y' \models \beta$, if β is evaluated to true when assigning `true` to every variable in Y', and `false` to every variable in $Y \setminus Y'$. In particular, we have $\emptyset \models \texttt{true}$. For $\beta \in \mathbb{B}^+(Y)$, let $|\beta|$ denote the size of β, defined inductively by $|\beta_1 \vee \beta_2| = |\beta_1 \wedge \beta_2| = |\beta_1| + |\beta_2| + 1$ and $|\beta| = 1$ for atomic formulas β.

Definition 3.1. An *alternating two-way automaton* (A2A) is a tuple $\mathcal{T} = (S, \Sigma, s_{\mathsf{in}}, \delta, S_f)$, where
- S is a finite set of states,
- Σ is a finite alphabet,
- $s_{\mathsf{in}} \in S$ is the initial state,
- $S_f \subseteq S$ is the set of accepting states, and
- $\delta \subseteq S \times (\Sigma \cup \{\mathsf{first?}\}) \times \mathbb{B}^+(S \times \{+1, 0, -1\})$ is the finite transition relation[1]. A transition $(s, \mathit{test}, \beta) \in \delta$ will also be written $s \xrightarrow{\mathit{test}} \beta$.

The size of $\mathcal{T}$ is defined as $|\mathcal{T}| = |S| + |\Sigma| + \sum_{(s,\mathit{test},\beta)\in\delta} |\beta|$.

While, in an OCA(P), $+1$ and -1 are interpreted as *increment* and *decrement* the counter, respectively, their interpretation in an A2A is *go to the right* and *go to the left* in the input word. Moreover, 0 means *stay*. Actually, when $\delta \subseteq S \times \Sigma \times (S \times \{+1\})$, then we deal with a classical *one-way finite automaton*.

A *run* of $\mathcal{T}$ on an infinite word $w = a_0a_1a_2 \cdots \in \Sigma^\omega$ is a rooted tree (possibly infinite, but finitely branching) whose vertices are labeled with elements in $S \times \mathbb{N}$. A node with label (s, n) represents a proof obligation that has to be fulfilled starting from state s and position n in the input word. The root of a run is labeled by $(s_{\mathsf{in}}, 0)$. Moreover, we require that, for every vertex labeled by (s, n) with $k \in \mathbb{N}$ children labeled by $(s_1, n_1), \ldots, (s_k, n_k)$, there is a transition $(s, \mathit{test}, \beta) \in \delta$ such that (i) the set $\{(s_1, n_1 - n), \ldots, (s_k, n_k - n)\} \subseteq S \times \{+1, 0, -1\}$ satisfies β, (ii) $\mathit{test} = \mathsf{first?}$ implies $n = 0$, and (iii) $\mathit{test} \in \Sigma$ implies $a_n = \mathit{test}$. Note that, similarly to an OCA(P), a transition with move -1 is blocked if $n = 0$, i.e., if $\mathcal{T}$ is at the first position of the input word. A run is *accepting* if every infinite branch visits some accepting state from S_f infinitely often. The language of $\mathcal{T}$ is defined as $L(\mathcal{T}) = \{w \in \Sigma^\omega \mid \text{there exists an accepting run of } \mathcal{T} \text{ on } w\}$. The *non-emptiness problem* for A2As is to decide, given an A2A $\mathcal{T}$, whether $L(\mathcal{T}) \neq \emptyset$.

Theorem 3.2 [Ser06]**.** *The non-emptiness problem for A2As is in* PSPACE.

It is worth noting that [Ser06] also uses two-wayness to simulate one-counter automata, but in a game-based setting (the latter is reflected by alternation).

We will now show how to build an A2A from the OCA(P) $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ and a target state $q_f \in Q$.

[1]One often considers a transition *function* $\delta : S \times \Sigma \times \{\mathsf{first?}, \neg\mathsf{first?}\} \to \mathbb{B}^+(S \times \{+1, 0, -1\})$, but a relation is more convenient for us.

Lemma 3.3. *There is an A2A $\mathcal{T} = (S, \Sigma, s_{\mathsf{in}}, \delta, S_f)$, with $\Sigma = \mathcal{X} \cup \{\Box\}$, such that $L(\mathcal{T}) = \{w \in \mathcal{W}_{\mathcal{X}} \mid q_f$ is γ_w-reachable in $\mathcal{A}\}$. Moreover, $|\mathcal{T}| = \mathcal{O}(|\mathcal{A}|^2)$.*

Proof. The states of $\mathcal{T}$ include Q (to simulate $\mathcal{A}$), a new initial state s_{in}, and some extra states, which will be introduced below.

Starting in s_{in} and at the first letter of the input word, the A2A $\mathcal{T}$ spawns several copies: $s_{\mathsf{in}} \xrightarrow{\Box} (q_{\mathsf{in}}, 0) \wedge \bigwedge_{x \in \mathcal{X}} (\mathsf{search}(x), +1)$. The copy starting in q_{in} will henceforth simulate $\mathcal{A}$. Moreover, from the new state $\mathsf{search}(x)$, we will check that x occurs exactly once in the input word. This is accomplished using the transitions $\mathsf{search}(x) \xrightarrow{x} (\checkmark_x, +1)$, as well as $\mathsf{search}(x) \xrightarrow{y} (\mathsf{search}(x), +1)$ and $\checkmark_x \xrightarrow{y} (\checkmark_x, +1)$ for all $y \in \Sigma \setminus \{x\}$. Thus, $\checkmark_x$ signifies that x has been seen and must not be encountered again. Since the whole word has to be scanned, $\checkmark_x$ is visited infinitely often so that we set $\checkmark_x \in S_f$.

It remains to specify how $\mathcal{T}$ simulates $\mathcal{A}$. The underlying idea is very simple. A configuration $(q, v) \in \mathcal{C}_{\mathcal{A}}$ of $\mathcal{A}$ corresponds to the configuration/proof obligation (q, i) of $\mathcal{T}$ where position i is the $(v+1)$-th occurrence of $\Box$ in the input parameter word. The simulation then proceeds as follows.

Increment/decrement: To mimic a transition $(q, +1, q') \in \Delta$, the A2A $\mathcal{T}$ simply goes to the next occurrence of $\Box$ on the right hand side and enters q'. This is accomplished by several A2A-transitions: $q \xrightarrow{\Box} (\mathsf{right}(q'), +1)$, $\mathsf{right}(q') \xrightarrow{x} (\mathsf{right}(q'), +1)$ for every $x \in \mathcal{X}$, and $\mathsf{right}(q') \xrightarrow{\Box} (q', 0)$. Decrements are handled similarly, using states of the form $\mathsf{left}(q')$. Finally, $(q, 0, q') \in \Delta$ is translated to $q \xrightarrow{\Box} (q', 0)$.

Equality test: A zero test $(q, {=}0, q') \in \Delta$ corresponds to $q \xrightarrow{\mathsf{first?}} (q', 0)$. To simulate a transition $(q, {=}x, q') \in \Delta$, the A2A spawns two copies. One goes from q to q' and stays at the current position. The other goes to the right and accepts if it sees x before hitting another $\Box$-symbol. Thus, we introduce a new state $\mathsf{present}(x)$ and transitions $q \xrightarrow{\Box} (q', 0) \wedge (\mathsf{present}(x), +1)$, $\mathsf{present}(x) \xrightarrow{x} \mathtt{true}$, and $\mathsf{present}(x) \xrightarrow{y} (\mathsf{present}(x), +1)$ for all $y \in \mathcal{X} \setminus \{x\}$. Note that there is *no* transition that allows $\mathcal{T}$ to read $\Box$ in state $\mathsf{present}(x)$.

Inequality tests: To simulate a test of the form $> x$, we proceed similarly to the previous case. The A2A generates a branch in charge of verifying that the parameter x lies strictly to the left of the current position. Note that transitions with label $< x$ are slightly more subtle, as x has to be retrieved strictly *beyond* the next delimiter $\Box$ to the right of the current position. More precisely, $(q, < x, q') \in \Delta$ translates to $q \xrightarrow{\Box} (q', 0) \wedge (\mathsf{search}^+(x), +1)$. We stay in $\mathsf{search}^+(x)$ until we encounter the next occurrence of $\Box$. We then go into $\mathsf{search}(x)$ (introduced at the beginning of the proof), which will be looking for an occurrence of x. Formally, we introduce $\mathsf{search}^+(x) \xrightarrow{y} (\mathsf{search}^+(x), +1)$ for all $y \in \mathcal{X} \setminus \{x\}$, and $\mathsf{search}^+(x) \xrightarrow{\Box} (\mathsf{search}(x), +1)$.

In state q_f, we accept: $q_f \xrightarrow{\Box} \mathtt{true}$. The only infinite branches in an accepting run are those that eventually stay in a state of the form $\checkmark_x$. Thus, we set $S_f = \{\checkmark_x \mid x \in \mathcal{X}\}$. It is not hard to see that $L(\mathcal{T}) = \{w \in \mathcal{W}_{\mathcal{X}} \mid q_f$ is γ_w-reachable$\}$. Note that $\mathcal{T}$ has a linear number of states and a quadratic number of transitions (some transitions of $\mathcal{A}$ are simulated by $\mathcal{O}(|\mathcal{X}|)$-many transitions of $\mathcal{T}$). $\Box$

A similar reduction takes care of Büchi reachability. Together with Theorem 3.2, we thus obtain the following:

Corollary 3.4. OCA(P)-Reachability *and* OCA(P)-Büchi *are both in* PSPACE.

Note that we are using alternation only to a limited extent. We could also reduce reachability in OCA(P) to non-emptiness of the intersection of several two-way automata. However, this would require a more complicated word encoding of parameter instantiations, which then have to include letters of the form $< x$ and $> x$.

3.2. **OCA(P)-Reachability is in NP.**

We will now show that we can improve the upper bounds given in Corollary 3.4 to NP. This upper bound is then optimal: NP-hardness, even in the presence of a single parameter, can be proved using a straightforward reduction from the non-emptiness problem for nondeterministic two-way automata over finite words over a *unary* alphabet and two end-markers, which is NP-complete [Gal76].

In a first step, we exploit our reduction from OCA(P) to A2As to establish a bound on the parameter values. To solve the reachability problems, it will then be sufficient to consider parameter instantiations up to that bound. For Lemma 3.5 and Corollary 3.6 below, we fix an OCA(P) $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ and a state $q_f \in Q$.

Lemma 3.5. *There is $d \in 2^{\mathcal{O}(|\mathcal{A}|^4)}$ such that the following holds: If $\mathcal{A} \models \mathit{Reach}(q_f)$, then there is a parameter instantiation $\gamma : \mathcal{X} \to \mathbb{N}$ such that $\gamma(x) \leq d$ for all $x \in \mathcal{X}$ and q_f is γ-reachable in $\mathcal{A}$.*

Proof. According to Lemma 3.3, there is an A2A $\mathcal{T} = (S, \mathcal{X} \cup \{\Box\}, s_{\mathsf{in}}, \delta, S_f)$ such that $L(\mathcal{T}) = \{w \in \mathcal{W}_{\mathcal{X}} \mid q_f \text{ is } \gamma_w\text{-reachable in } \mathcal{A}\}$ and $|\mathcal{T}| = \mathcal{O}(|\mathcal{A}|^2)$. By [Var98], there is a (nondeterministic) Büchi automaton $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{T})$ and $|\mathcal{B}| \in 2^{\mathcal{O}(|\mathcal{T}|^2)}$ (cf. also [DK08]).

Let $d = |\mathcal{B}|$. Suppose $\mathcal{A} \models \mathit{Reach}(q_f)$. This implies that $L(\mathcal{B}) \neq \emptyset$. But then, there must be a word $u \in \Sigma^*$ such that $|u| \leq |\mathcal{B}|$ ($|u|$ denoting the length of u) and $w = u\Box^\omega \in L(\mathcal{B})$. We have that q_f is γ_w-reachable and $\gamma_w(x) \leq |\mathcal{B}| = d$ for all $x \in \mathcal{X}$. □

As a corollary, we obtain that it is sufficient to consider bounded runs only. For a parameter instantiation $\gamma : \mathcal{X} \to \mathbb{N}$ and a bound $d \in \mathbb{N}$, we say that a *γ-run* $\rho = (q_0, v_0) \longrightarrow_\gamma (q_1, v_1) \longrightarrow_\gamma (q_2, v_2) \longrightarrow_\gamma \cdots$ is *d-bounded* if all its counter values $v_0, v_1, \ldots$ are at most d.

Corollary 3.6. *There is $d \in 2^{\mathcal{O}(|\mathcal{A}|^4)}$ such that the following holds: If $\mathcal{A} \models \mathit{Reach}(q_f)$, then there is a parameter instantiation $\gamma : \mathcal{X} \to \mathbb{N}$ such that $\gamma(x) \leq d$ for all $x \in \mathcal{X}$ and q_f is reachable within a d-bounded γ-run.*

Proof. Consider $d \in 2^{\mathcal{O}(|\mathcal{A}|^4)}$ due to Lemma 3.5. A standard argument in OCA with n states is that, for reachability, it is actually sufficient to consider runs up to some counter value in $\mathcal{O}(n^3)$ [LLT05, CCH+16]. We can apply the same argument here to deduce, together with Lemma 3.5, that parameter and counter values can be bounded by $d + \mathcal{O}(|Q|^3)$. □

The algorithm that solves OCA(P)-Reachability in NP will guess a parameter instantiation γ and a maximal counter value in binary representation (thus, of polynomial size). It then remains to determine, in polynomial time, whether the target state is reachable under this guess. To this end, we will exploit the lemma below due to Galil [Gal76].

Let $\mathcal{A} = (Q, q_{\mathsf{in}}, \Delta)$ be an OCA, $q, q' \in Q$, and $v, v' \in \mathbb{N}$. A run $(q_0, v_0) \longrightarrow (q_1, v_1) \longrightarrow \cdots \longrightarrow (q_{n-1}, v_{n-1}) \longrightarrow (q_n, v_n)$ of $\mathcal{A}$, $n \in \mathbb{N}$, is called a *(v, v')-run* if $\min\{v, v'\} < v_i < \max\{v, v'\}$ for all $i \in \{1, \ldots, n-1\}$. In other words, all intermediate configurations have counter values strictly between v and v'.

Lemma 3.7 [Gal76]. *The following problems can be solved in polynomial time:*

Input:	An OCA $\mathcal{A} = (Q, q_{\mathsf{in}}, \Delta)$, $q, q' \in Q$, and $v, v' \in \mathbb{N}$ given in binary.
Question 1:	Is there a (v, v')-run from (q, v) to (q', v') ?
Question 2:	Is there a (v, v')-run from (q, v) to (q', v) ?

Actually, [Gal76] uses the polynomial-time algorithms stated in Lemma 3.7 to show that non-emptiness of two-way automata on finite words over a unary alphabet is in NP (cf. the proof of the corresponding lemma on page 84 of [Gal76], where the end-markers of a given word correspond to v and v').

We can now deduce the following result, which is an important step towards model checking, but also of independent interest.

Theorem 3.8. OCA(P)-Reachability *is* NP*-complete.*

Sketch of proof. Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ be an OCA(P) with $\mathcal{X} = \{x_1, \ldots, x_n\}$, and let $q_f \in Q$ be a state of $\mathcal{A}$. Without loss of generality, we consider reachability of the *configuration* $(q_f, 0)$ (since one can add a new target state and a looping decrement transition).

Our nondeterministic polynomial-time algorithm proceeds as follows. First, it guesses $d \in 2^{\mathcal{O}(|\mathcal{A}|^4)}$ due to Corollary 3.6 (note that the binary representation of d is of polynomial size with respect to $|\mathcal{A}|$), as well as some parameter instantiation γ satisfying $\gamma(x_i) \leq d$ for all $1 \leq i \leq n$ where each $d_i := \gamma(x_i)$ is represented in binary, too. We may assume $n \geq 1$ and that $d_0 < d_1 < d_2 < \cdots < d_n < d_{n+1}$, where $d_0 = 0$ and $d_{n+1} = d$ (otherwise, we can rename the parameters accordingly). Second, the algorithm checks that there exists a d-bounded γ-run of the form

$$(q_0, v_0) \longrightarrow_\gamma^* (q_0', v_0) \longrightarrow_\gamma^* (q_1, v_1) \longrightarrow_\gamma^* (q_1', v_1) \longrightarrow_\gamma^* \cdots \longrightarrow_\gamma^* (q_k, v_k) \longrightarrow_\gamma^* (q_k', v_k)$$

such that (letting $D = \{d_0, d_1, \ldots, d_n, d_{n+1}\}$)

(1) $(q_0, v_0) = (q_{\mathsf{in}}, 0)$ and $(q_k', v_k) = (q_f, 0)$,
(2) $v_j \in D$ for all $j \in \{0, \ldots, k\}$,
(3) between (q_j, v_j) and (q_j', v_j), the counter values always equal v_j, for all $j \in \{0, \ldots, k\}$,
(4) (strictly) between (q_j', v_j) and (q_{j+1}, v_{j+1}), the counter values are always different from the values in D, for all $j \in \{0, \ldots, k-1\}$.

Note that we can assume $k \leq |Q| \cdot (|\mathcal{X}| + 2)$, since in every longer run, the exact same configuration is encountered at least twice.

Hence, to check whether there exists a finite initialized d-bounded γ-run $(q_{\mathsf{in}}, 0) \longrightarrow_\gamma^* (q_f, 0)$, it is sufficient to guess $2k$ configurations, where $k \leq |Q| \cdot (|\mathcal{X}| + 2)$, and to verify that these configurations contribute to constructing a run of the form described above. This can be checked in polynomial time: The case (3) is obvious, and (4) is due to Lemma 3.7.

The lower bound can be easily obtained from [Gal76] stating that the non-emptiness problem for nondeterministic two-way automata over finite words over a *unary* alphabet and two end-markers is NP-complete.[2] In fact, we can easily build an OCA(P) with a single parameter that simulates the behavior of such an automaton and where the parameter is used to guess the length of the read word (and, consequently, to detect the end-marker). □

[2]On the other hand, the non-emptiness problem for nondeterministic two-way automata over a unary alphabet and with a single end-marker (which can be simulated by an OCA) is $\mathsf{NLOGSPACE}$-complete.

3.3. **Dealing with binary constants.** The previous NP algorithm can be adapted to the case of OCA(P,C) where, in addition to parameters, the counter values can be compared with constants $c \in \mathbb{N}$ encoded in binary. To this end, we transform a given OCA(P,C) $\mathcal{A}$ into an OCA(P) $\mathcal{A}'$ by interpreting every constant c from $\mathcal{A}$ as a parameter x_c and build the corresponding A2A as stated by Lemma 3.3. Then, we can adapt the proof of Lemma 3.5 to show that, if $\mathcal{A} \models \mathit{Reach}(q_f)$ for a given q_f, then there is a parameter instantiation where all parameter values can be bounded by an exponential bound. This is enough to obtain the desired upper-bound following the same reasoning as in the proof of Theorem 3.8.

Theorem 3.9. OCA(P,C)-Reachability *is* NP*-complete.*

Proof. Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ be an OCA(P,C) and let $q_f \in Q$. Moreover, let $C = \{c \in \mathbb{N} \mid c \text{ appears in } \Delta_{\mathsf{const}}\}$. From $\mathcal{A}$, we build an OCA(P) $\mathcal{A}' = (Q, \mathcal{X}', q_{\mathsf{in}}, \Delta')$ where $\mathcal{X}' = \mathcal{X} \cup \{x_c \mid c \in C\}$ and Δ' is obtained from Δ by replacing each transition $(q, \bowtie c, q') \in \Delta_{\mathsf{const}}$ by $(q, \bowtie x_c, q')$ and keeping the other transitions.

Now, using Lemma 3.3, we know there is an A2A $\mathcal{T}'$ such that $L(\mathcal{T}') = \{w \in \mathcal{W}_{\mathcal{X}'} \mid q_f \text{ is } \gamma_w\text{-reachable in } \mathcal{A}'\}$ and $|\mathcal{T}'| = \mathcal{O}(|\mathcal{A}'|^2)$. As in the proof of Lemma 3.5, we know then from [Var98] that there is a (nondeterministic) Büchi automaton $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{T}')$ and $|\mathcal{B}| \in 2^{\mathcal{O}(|\mathcal{T}'|^2)}$. Suppose $\mathcal{A} \models \mathit{Reach}(q_f)$. This implies that there is a parameter word w over $\mathcal{X}'$ that belongs to $L(\mathcal{B})$ and such that $\gamma_w(x_c) = c$ for all $c \in C$. Let $c_{\max}$ be maximal in C. Note that, since the constants are encoded in binary in $\mathcal{A}$, we have $c_{\max} \in 2^{\mathcal{O}(|\mathcal{A}|)}$. Then, there must also be a word $u \in (\mathcal{X}' \cup \{\Box\})^*$ such that $|u| \leq |\mathcal{B}| + c_{\max}$ ($|u|$ denoting the length of u), $w' := u\Box^\omega \in L(\mathcal{B})$, and $\gamma_{w'}(x_c) = c$ for all $c \in C$. Hence, q_f is $\gamma_{w'}$-reachable and $\gamma_{w'}(x) \leq c_{\max} + |\mathcal{B}| =: d$ for all $x \in \mathcal{X}'$. Note that $d \in 2^{\mathcal{O}(|\mathcal{A}|^4)}$.

Then, we follow the same reasoning as in the proof of Theorem 3.8: We guess a parameter instantiation γ satisfying $\gamma(x) \leq d$ for all $x \in \mathcal{X}'$. In fact, the only difference is that, for $c \in C$, we do not *guess* $\gamma(x_c)$ but *determine* $\gamma(x_c)$ to be c. This ensures that, if we find a γ-run in $\mathcal{A}'$ reaching q_f, then $\mathcal{A} \models \mathit{Reach}(q_f)$. Furthermore, the previous reasoning ensures that the bound d is enough to witness a run in $\mathcal{A}$ reaching q_f. This allows us to deduce that OCA(P,C)-Reachability is in NP. The hardness part comes from the fact that the problem is already NP-hard when considering OCA(P). ☐

Remark 3.10. Using a reduction from [BO17], Theorem 3.9 will allow us to establish an optimal $\mathsf{NEXPTIME}$ algorithm for deciding non-emptiness of closed parametric timed automata with a single parametric clock [AHV93, BO17] (cf. Section 6).

4. From OCA(P,C)-Reachability to OCA(S)-Model-Checking

This section is dedicated to model checking OCA against $\flat$LTL$^{\downarrow}$. In Section 4.1, we establish a couple of reductions from OCA(S)-MC($\flat$LTL$^{\downarrow}$) down to OCA(P)-Reachability, which allow us to conclude that the former problem is in $\mathsf{NEXPTIME}$. Then, in Section 4.2, we provide a matching lower bound.

4.1. **Upper Bounds.** First, we use the fact that OCA(P)-Reachability is in NP (Theorem 3.8) to show that OCA(P)-Büchi is in NP, too:

Lemma 4.1. OCA(P)-Büchi *is polynomial-time reducible to* OCA(P)-Reachability.

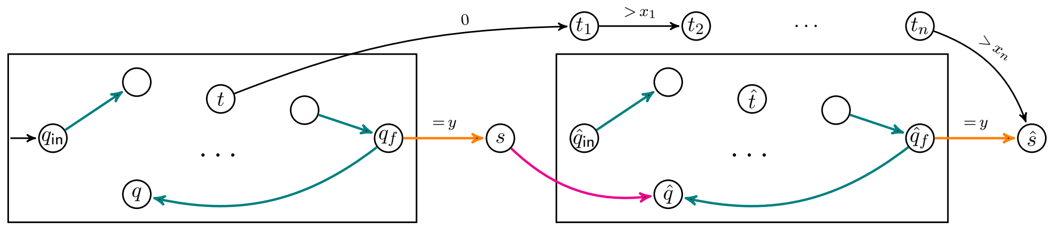

Figure 2: The OCA(P) $\mathcal{A}'$ constructed from $\mathcal{A}$ in the proof of Lemma 4.1.

Proof. Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ be an OCA(P) with $\mathcal{X} = \{x_1, \ldots, x_n\}$ (without loss of generality, we suppose $n \geq 1$), and let $F \subseteq Q$. It is sufficient to give an algorithm that determines whether a particular state $q_f \in F$ can be repeated infinitely often. Thus, we construct an OCA(P) $\mathcal{A}'$, with a distinguished state $\hat{s}$, such that $\mathcal{A} \models \mathit{Reach}^{\omega}(q_f)$ iff $\mathcal{A}' \models \mathit{Reach}(\hat{s})$.

To this end, we first define an OCA $\mathcal{M}$, which is obtained from $\mathcal{A}$ by (i) removing all transitions whose labels are of the form $=0$, $=x$, or $<x$, for some $x \in \mathcal{X}$, and (ii) replacing all transition labels $>x$ (for some $x \in \mathcal{X}$) by 0. Then, we compute the set $T \subseteq Q$ of states t of $\mathcal{M}$ from which there is an infinite run starting in $(t, 0)$ and visiting q_f infinitely often. This can be done in $\mathsf{NLOGSPACE}$ [DG09].

Now, we obtain the desired OCA(P) $\mathcal{A}' = (Q', \mathcal{X}', q_{\mathsf{in}}, \Delta')$ as follows. The set of states is $Q' = Q \uplus \{\hat{q} \mid q \in Q\} \uplus \{s, \hat{s}, t_1, \ldots, t_n\}$. That is, we introduce a copy $\hat{q}$ for every $q \in Q$ as well as fresh states $s, \hat{s}, t_1, \ldots t_n$. Recall that $\hat{s}$ will be the target state of the reachability problem we reduce to. Moreover, $\mathcal{X}' = \mathcal{X} \uplus \{y\}$ where y is a fresh parameter. It remains to define the transition relation Δ', which includes Δ as well as some new transitions (*cf.* Figure 2). Essentially, there are two cases to consider:

(1) First, q_f may be visited infinitely often in $\mathcal{A}$, and infinitely often along with the same counter value. To take this case into account, we use the new parameter y and a new transition $(q_f, =y, s)$, which allows $\mathcal{A}'$ to "store" the current counter value in y. From s, we then enter and simulate the copy of $\mathcal{A}$, i.e., we introduce transitions $(s, \mathsf{op}, \hat{q})$ for all $(q_f, \mathsf{op}, q) \in \Delta$, as well as $(\hat{q}_1, \mathsf{op}, \hat{q}_2)$ for all $(q_1, \mathsf{op}, q_2) \in \Delta$. If, in the copy, $\hat{q}_f$ is visited along with the value stored in y, we may thus enter $\hat{s}$.

(2) Now, suppose that a γ-run ρ of $\mathcal{A}$ visits q_f infinitely often, but not infinitely often with the same counter value. If ρ visits some counter value $v \leq \max\{\gamma(x) \mid x \in \mathcal{X}\}$ infinitely often, then it necessarily contains a subrun of the form $(q, v) \longrightarrow_{\gamma}^{*} (q_f, v') \longrightarrow_{\gamma}^{*} (q, v)$ for some $q \in Q$ and $v' \in \mathbb{N}$. But then, there is an infinite γ-run that visits (q_f, v') infinitely often so case (1) above applies. Otherwise, ρ will eventually stay strictly above $\max\{\gamma(x) \mid x \in \mathcal{X}\}$. Thus, we add the following transitions to Δ', which will allow $\mathcal{A}'$ to move from $t \in T$ to $\hat{s}$ provided the counter value is greater than the maximal parameter value: $(t, 0, t_1)$ for all $t \in T$, as well as $(t_1, >x_1, t_2), (t_2, >x_2, t_3), \ldots, (t_n, >x_n, \hat{s})$. □

From Theorem 3.8 and Lemma 4.1, we obtain the exact complexity of OCA(P)-Büchi (the lower bound is by a straightforward reduction from OCA(P)-Reachability).

Corollary 4.2. OCA(P)-Büchi *is* NP*-complete.*

The following link between OCA-MC($\flat$LTL$^{\downarrow}$) and OCA(P)-Büchi is due to [DS10]:

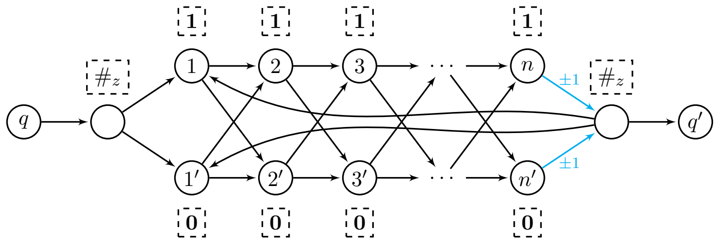

Figure 3: The OCA-gadget for simulating an OCA(S)-transition (q, z, q') with binary update z. The new propositions are depicted in dashed boxes. States $1, \ldots, n, 1', \ldots, n'$, where $n = \mathit{bits}(z)$, represent the bits needed to encode z in binary. At the transitions originating from n and n', the counter is updated by $+1$ or -1, depending on whether z is positive or negative, respectively.

Lemma 4.3 [DS10]**.** *Let $\mathcal{A}$ be an* OCA *and $\varphi \in \flat\mathrm{LTL}^{\downarrow}$ be a sentence. One can compute, in exponential time, an* OCA(P) $\mathcal{A}' = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta)$ *(of exponential size) and a set $F \subseteq Q$ such that $\mathcal{A} \models_{\exists} \varphi$ iff $\mathcal{A}' \models \mathit{Reach}^{\omega}(q_f)$ for some $q_f \in F$.*

Actually, the construction from [DS10] only considers the fragment of $\flat\mathrm{LTL}^{\downarrow}$ without any register tests of the form $<r$ and $>r$, but it can easily be extended to handle them. On the other hand, the restriction to $\flat\mathrm{LTL}^{\downarrow}$ is crucial. It allows one to assume that a register is written at most once so that, in the OCA(P), it can be represented as a parameter. In particular, $\mathcal{X}$ is the set of registers that occur in φ. From Corollary 4.2 and Lemma 4.3, we now deduce our first model-checking result:

Corollary 4.4. OCA-MC($\flat\mathrm{LTL}^{\downarrow}$) *is in* NEXPTIME*.*

However, it turns out that OCA(S) model checking is no harder than OCA model checking. The reason is that succinct updates can be encoded in small LTL formulas.

Lemma 4.5. OCA(S)-MC($\flat\mathrm{LTL}^{\downarrow}$) *is polynomial-time reducible to* OCA-MC($\flat\mathrm{LTL}^{\downarrow}$)*.*

Proof. Let $\mathcal{A} = (Q, q_{\mathsf{in}}, \Delta, \mu)$ be an OCA(S) and $\varphi \in \flat\mathrm{LTL}^{\downarrow}$ be a sentence. Without loss of generality, we assume that φ is in negation normal form, i.e., negation is applied only to atomic propositions. This is thanks to the logical equivalences $\neg(\varphi_1 \mathsf{U} \varphi_2) \equiv \neg\varphi_1 \mathsf{R} \neg\varphi_2$ and $\neg(\varphi_1 \mathsf{R} \varphi_2) \equiv \neg\varphi_1 \mathsf{U} \neg\varphi_2$. We construct an OCA $\mathcal{A}' = (Q', q'_{\mathsf{in}}, \Delta', \mu')$ and a sentence $\varphi' \in \flat\mathrm{LTL}^{\downarrow}$ of polynomial size such that $\mathcal{A} \models_{\exists} \varphi$ iff $\mathcal{A}' \models_{\exists} \varphi'$.

Let $Z = \{z \mid (q, z, q') \in \Delta \cap (Q \times \mathbb{Z} \times Q) \text{ with } |z| \geq 2\}$ and let $\Lambda = \{\#_z \mid z \in Z\} \cup \{\mathbf{0}, \mathbf{1}\}$ be a set of fresh propositions. To obtain $\mathcal{A}'$ from $\mathcal{A}$, we replace every transition $(q, z, q') \in \Delta \cap (Q \times Z \times Q)$ by the gadget depicted in Figure 3. The gadget implements a binary counter generating sequences (of sets of propositions) of the form depicted in Figure 4, for $z = 6$. We assume that the least significant bit is on the left. To make sure that the binary counter works as requested, we define an LTL formula *Counter*.

Towards its definition, let us first introduce some notation and abbreviations. For $z \in Z$, let $\mathit{bits}(z)$ denote the number of bits needed to represent the binary encoding of $|z|$. For example, $\mathit{bits}(6) = 3$. For $i \in \{1, \ldots, \mathit{bits}(z)\}$, we let $\mathit{bit}_i(z)$ denote the i-th bit in that encoding. For example, $(\mathit{bit}_1(6), \mathit{bit}_2(6), \mathit{bit}_3(6)) = (\mathbf{0}, \mathbf{1}, \mathbf{1})$. In the following, we write Λ for $\bigvee_{\gamma \in \Lambda} \gamma$ and $\neg\Lambda$ for $\bigwedge_{\gamma \in \Lambda} \neg\gamma$. For an illustration of the following LTL formulas, we refer to Figure 4. Formula $\mathit{first}(\#_z) = \neg\Lambda \wedge \mathsf{X}\#_z$ holds right before a first delimiter symbol. Similarly,

$$\begin{array}{ccccccccccccccc}
\mu(q) & \#_6 & \mathbf{100} & \#_6 & \mathbf{010} & \#_6 & \mathbf{110} & \#_6 & \mathbf{001} & \#_6 & \mathbf{101} & \#_6 & \mathbf{011} & \#_6 & \mu(q') \\
 & \uparrow & & & \uparrow & & & & & & & \uparrow & & \uparrow & \\
 & \mathit{first}(\#_6) & & & \mathit{suff}^{=}_{\#_6} & & & & & & & \mathit{last}^{-}(\#_6) & & \mathit{last}(\#_6) &
\end{array}$$

Figure 4: A counting sequence

$\mathit{last}(\#_z) = \#_z \wedge \mathsf{X}\neg\Lambda$ identifies all last delimiters and $\mathit{last}^{-}(\#_z) = \#_z \wedge \mathsf{X}\big((\mathbf{0} \vee \mathbf{1})\,\mathsf{U}\,\mathit{last}(\#_z)\big)$ all second-last delimiters. Finally, we write $\curvearrowright_z\psi$ for $\mathsf{X}^{\mathit{bits}(z)+1}\psi$.

Now, $\mathit{Counter} = \mathit{Init} \wedge \mathit{Fin} \wedge \mathit{Inc} \wedge \mathit{Exit}$ is the conjunction of the following formulas:

- *Init* says that the first binary number is always one:
$$\mathit{Init} = \textstyle\bigwedge_{z\in Z} \mathsf{G}\big(\mathit{first}(\#_z) \ \rightarrow\ \mathsf{X}^2(\mathbf{1} \ \wedge\ \mathsf{X}(\mathbf{0}\,\mathsf{U}\,\#_z))\big)$$
- *Fin* says that the last value represents $|z|$:
$$\mathit{Fin} = \textstyle\bigwedge_{z\in Z} \mathsf{G}\big(\mathit{last}^{-}(\#_z) \ \rightarrow\ \bigwedge_{i=1}^{\mathit{bits}(z)} \mathsf{X}^i\, \mathit{bit}_i(z)\big)$$
- *Inc* implements the increments:
$$\mathit{Inc} = \bigwedge_{z\in Z} \mathsf{G}\left(\begin{array}{l} \qquad\big(\#_z \wedge \neg\mathit{last}^{-}(\#_z) \wedge \neg\mathit{last}(\#_z)\big) \\ \rightarrow\ \ \mathsf{X}\big((\mathbf{1} \wedge \curvearrowright_z\mathbf{0})\,\mathsf{U}\,(\mathbf{0} \wedge \curvearrowright_z\mathbf{1} \wedge \mathsf{X}\,\mathit{suff}^{=}_z)\big)\end{array}\right)$$
  where $\mathit{suff}^{=}_z = \big((\mathbf{0} \wedge \curvearrowright_z\mathbf{0}) \vee (\mathbf{1} \wedge \curvearrowright_z\mathbf{1})\big)\,\mathsf{U}\,\#_z$ verifies that the suffixes (w.r.t. the current position) of the current and the following binary number coincide.
- *Exit* states that we do not loop forever in the gadget from Figure 3:
$$\mathit{Exit} = \textstyle\bigwedge_{z\in Z} \mathsf{G}\big(\mathit{first}(\#_z) \ \rightarrow\ \mathsf{F}\ \mathit{last}(\#_z)\big)$$

Note that the gadget makes sure that, in between two delimiters $\#_z$, there are exactly $\mathit{bits}(z)$-many bits.

Now, we set $\varphi' = [\varphi] \wedge \mathit{Counter}$. Here, $[\varphi]$ simulates φ on all positions that do not contain propositions from Λ. It is inductively defined as follows:

$$[p] = p \qquad [\neg p] = \neg p \qquad [\bowtie r] = \bowtie r \qquad [\downarrow_r\psi] = \downarrow_r[\psi]$$

$$[\psi_1 \vee \psi_2] = [\psi_1] \vee [\psi_2] \qquad [\psi_1 \wedge \psi_2] = [\psi_1] \wedge [\psi_2] \qquad [\mathsf{X}\varphi] = \mathsf{X}\big(\Lambda\,\mathsf{U}\,(\neg\Lambda \wedge [\varphi])\big)$$

$$[\psi_1\,\mathsf{U}\,\psi_2] = (\neg\Lambda \rightarrow [\psi_1])\,\mathsf{U}\,(\neg\Lambda \wedge [\psi_2]) \qquad [\psi_1\ \mathsf{R}\ \psi_2] = (\neg\Lambda \wedge [\psi_1])\ \mathsf{R}\ (\neg\Lambda \rightarrow [\psi_2])$$

Note that, since φ was required to be in negation normal form, φ' is indeed in $\flat\mathrm{LTL}^{\downarrow}$. $\square$

Corollary 4.4 and Lemma 4.5 imply the NEXPTIME upper bound for OCA(S) model checking:

Corollary 4.6. OCA(S)-MC($\flat$LTL$^{\downarrow}$) *is in* NEXPTIME*.*

4.2. Lower Bounds.

This subsection presents a matching lower bound for model checking $\flat$LTL$^{\downarrow}$. We first state NEXPTIME-hardness of OCA(P)-MC(LTL), which we reduce, in a second step, to OCA-MC($\flat$LTL$^{\downarrow}$).

Theorem 4.7. OCA(P)-MC(LTL) *and* OCA(S,P)-MC(LTL) *are* NEXPTIME*-complete.*

Proof. For $\mathsf{NEXPTIME}$-hardness of OCA(S,P)-MC(LTL), we adapt the proof of [GHOW10, Theorem 8], where Göller et al. show $\mathsf{NEXPTIME}$-hardness of LTL model checking OCA with parametric and succinct *updates.* In an OCA with parametric and succinct updates, the counter can be updated by either an integer (encoded in binary) or by some parameter, which, similarly to parameterized *tests* in OCA(P) and OCA(S,P), must be assigned some concrete integer value by some parameter instantiation; note that the model in [GHOW10] does not allow for parameterized tests, but only for zero tests. Since the value of the counter before an update is a priori not known, it is not possible to simulate parametric updates with parameterized tests. However, in the reduction used in the proof of [GHOW10], parametric updates are preceded directly by a zero test. Hence, in this specific case, one can substitute the parametric update with a self loop increasing the counter and a parameterized test. This allows us to reuse the technique from [GHOW10].

Let us give a few more details. The proof of Theorem 8 in [GHOW10] is a reduction from Succinct 3-SAT: given a Boolean circuit $\mathbb{C}$ that encodes a Boolean formula $\psi_{\mathbb{C}}$ in 3-CNF, Göller et al. construct an OCA $\mathcal{A}$ with parametric and succinct updates as well as an LTL formula φ such that there exist some parameter valuation γ and some γ-run of $\mathcal{A}$ satisfying φ if, and only if, $\psi_{\mathbb{C}}$ is satisfiable. The OCA $\mathcal{A}$ uses a single parameter x. One can think of the value of x as the encoding of a guessed variable assignment for the variables occurring in $\psi_{\mathbb{C}}$. Göller et al. use $\mathcal{A}$ and φ to verify whether this guessed assignment satisfies the Boolean formula $\psi_{\mathbb{C}}$. For proving Theorem 4.7, we can almost completely rely on the proof of Theorem 8 in [GHOW10]. As mentioned above, all we need to change is that, in Figure 3 of [GHOW10], we replace the parametric update $+x$ by an equality test $=x$ and add a self-loop to the start state with label $+1$. Like in [GHOW10], the value of the counter, after satisfying the parameterized test $=x$, encodes the variable assignment of the variables occurring in $\psi_{\mathbb{C}}$. Satisfaction can then be verified using the same OCA gadgets and the same LTL formula as in [GHOW10]. We remark that the lower bound already holds for OCA(S,P) that use only one parameter x and where all parameterized tests are of the form $=x$.

Membership of OCA(P)-MC(LTL) in $\mathsf{NEXPTIME}$ is by a standard argument. The given LTL formula can be translated into a Büchi automaton of exponential size. Then, we check non-emptiness of its product with the given OCA(P) using Corollary 4.2.

Moreover, OCA(S,P)-MC(LTL) can be reduced to OCA(P)-MC(LTL) in polynomial time, using the construction from Lemma 4.5. This yields $\mathsf{NEXPTIME}$-hardness of OCA(P)-MC(LTL) and $\mathsf{NEXPTIME}$-membership of OCA(S,P)-MC(LTL). □

Lemma 4.8. OCA-MC($\flat$LTL$^{\downarrow}$) *and* OCA(S)-MC($\flat$LTL$^{\downarrow}$) *are* $\mathsf{NEXPTIME}$*-hard.*

Proof. We present a reduction from OCA(P)-MC(LTL), which is $\mathsf{NEXPTIME}$-complete by Theorem 4.7. Let $\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta, \mu)$ be an OCA(P) and let φ be an LTL formula. We define an OCA $\mathcal{A}' = (Q', q'_{\mathsf{in}}, \Delta', \mu')$ and a sentence $\varphi' \in \flat\text{LTL}^{\downarrow}$ such that $\mathcal{A} \models_{\exists} \varphi$ iff $\mathcal{A}' \models_{\exists} \varphi'$.

Without loss of generality, by [GHOW10] (cf. proof of Theorem 4.7), we may assume that $\mathcal{A}$ uses only one parameter x and that all parameterized tests are of the form $=x$.

The idea is as follows. A new initial state q'_{in}, in which only a fresh proposition *freeze* holds, will allow $\mathcal{A}'$ to go to an arbitrary counter value, say, v. The $\flat$LTL$^{\downarrow}$ formula of the form $\mathsf{F}(\mathit{freeze} \wedge \downarrow_r \psi)$ stores v, which will henceforth be interpreted as parameter x. Hereby, formula ψ makes sure that, whenever $\mathcal{A}$ performs an equality check $=x$, the current counter value coincides with v. To do so, we create two copies of the original state space: $Q \times \{0, 1\}$. A transition $(q, =x, q')$ of $\mathcal{A}$ is then simulated, in $\mathcal{A}'$, by a 0-labeled transition to $(q', 1)$.

States of the latter form are equipped with a fresh proposition $p_{=x}$ indicating that the current counter value has to coincide with the contents of r. Thus, we can set $\psi = \mathsf{G}(p_{=x} \to {=r})$.

Formally, $\mathcal{A}'$ is given as follows. We let $Q' = (Q \times \{0,1\}) \uplus \{q'_{\mathsf{in}}\}$ and

$$\begin{aligned}\Delta' = \quad & \{(q'_{\mathsf{in}}, \mathsf{op}, q'_{\mathsf{in}}) \mid \mathsf{op} \in \{+1,-1\}\} \ \cup\ \{(q'_{\mathsf{in}}, {=0}, (q_{\mathsf{in}}, 0))\}\\ \cup \quad & \{((q,b), 0, (q,1)) \mid (q, {=x}, q') \in \Delta \text{ and } b \in \{0,1\}\}\\ \cup \quad & \{((q,b), \mathsf{op}, (q,0)) \mid (q, \mathsf{op}, q') \in \Delta \setminus (Q \times \{=x\} \times Q) \text{ and } b \in \{0,1\}\}\,.\end{aligned}$$

Moreover, we set $\mu'(q'_{\mathsf{in}}) = \{\mathit{freeze}\}$ as well as $\mu'((q,0)) = \mu(q)$ and $\mu'((q,1)) = \mu(q) \cup \{p_{=x}\}$ for all $q \in Q$. Finally, $\varphi' = \mathsf{F}\big(\mathit{freeze} \wedge {\downarrow_r}\mathsf{G}(p_{=x} \to {=r})\big) \wedge \big(\mathit{freeze}\, \mathsf{U}\, (\neg\mathit{freeze} \wedge \varphi)\big)$. Note that the subformula $\mathit{freeze}\,\mathsf{U}\,(\neg\mathit{freeze} \wedge \varphi)$ makes sure that φ is satisfied as soon as we enter the original initial state q_{in} (actually, $(q_{\mathsf{in}}, 0)$).

We now prove the correctness of the above construction.

Completeness: Assume $\rho = (q_0, v_0) \longrightarrow_\gamma (q_1, v_1) \longrightarrow_\gamma \ldots$ is a γ-run of $\mathcal{A}$ for some parameter instantiation γ such that $\rho \models \varphi$. Using the definition of Δ', it is easy to construct from ρ a run ρ' of $\mathcal{A}'$: replace every transition $(q_i, v_i) \longrightarrow_\gamma (q_{i+1}, v_{i+1})$ resulting from some transition $(q_i, {=x}, q_{i+1}) \in \Delta$ by $((q_i, b), v_i) \longrightarrow ((q_{i+1}, 1), v_{i+1})$, and replace every transition $(q_i, v_i) \longrightarrow_\gamma (q_{i+1}, v_{i+1})$ resulting from some transition $(q_i, \mathsf{op}_i, q_{i+1}) \in \Delta \setminus (Q \times \{=x\} \times Q)$ by $((q_i, b), v_i) \longrightarrow ((q_{i+1}, 0), v_{i+1})$, where, in both cases, $b \in \{0,1\}$ is such that the result ρ' is a run of $\mathcal{A}'$. We obtain an initialized run ρ'' of $\mathcal{M}$ by putting in front of ρ' a prefix run ρ_x of the form

$$(q'_{\mathsf{in}}, 0) \longrightarrow \underbrace{\ldots \longrightarrow (q'_{\mathsf{in}}}_{\gamma(x) \text{ times “}+1\text{”}}, \gamma(x)) \longrightarrow \underbrace{(q'_{\mathsf{in}}, \gamma(x)-1) \longrightarrow \ldots \longrightarrow (q'_{\mathsf{in}}}_{\gamma(x) \text{ times “}-1\text{”}}, 0) \longrightarrow ((q_{\mathsf{in}}, 0), 0).$$

That $\rho'' \models \varphi'$ can be easily seen. First of all, the subformula ${\downarrow_r}\mathsf{G}(p_{=x} \to {=r})$ is satisfied at the configuration $(q'_{\mathsf{in}}, \gamma(x))$: the formula stores the value $\gamma(x)$ into the register. From that position on, every time a state $(q,1)$ (with proposition $p_{=x}$) is seen along the run ρ'', we know that $\mathcal{A}'$ has just simulated a parameterized test $=x$ of $\mathcal{A}$, hence the value of the counter is indeed equal to $\gamma(x)$. Second, the subformula $(\neg\mathit{freeze} \wedge \varphi)$ is satisfied at $((q_{\mathsf{in}}, 0), 0)$, by assumption and by the definition of μ'.

Soundness: Assume $\rho' = (q'_0, v'_0) \longrightarrow (q'_1, v'_1) \longrightarrow \ldots$ is a run of $\mathcal{A}'$ such that $\rho' \models \varphi'$. Recall that $\mu'(q) = \mathit{freeze}$ implies $q = q'_{\mathsf{in}}$. By $\rho' \models \mathit{freeze}\,\mathsf{U}\,(\neg\mathit{freeze} \wedge \varphi)$ we can conclude that there exists some position j in ρ' such that $q'_i = q'_{\mathsf{in}}$ for all $0 \le i < j$, and $q'_j = (q_{\mathsf{in}}, 0)$, and

$$\rho', j \models \varphi. \tag{4.1}$$

By $\rho' \models \mathsf{F}\big(\mathit{freeze} \wedge {\downarrow_r}\mathsf{G}(p_{=x} \to {=r})\big)$ we further conclude that there exists some position $k < j$ such that

$$\rho', k \models \mathit{freeze} \wedge {\downarrow_r}\mathsf{G}(p_{=x} \to {=r}). \tag{4.2}$$

Note that this stores the counter value v'_k into the register r. Intuitively, v'_k will correspond to the value of the parameter x. We thus define $\gamma(x) = v'_k$. By construction, the subrun ρ'_{suffix} of ρ' starting in position j is of the form

$$((q_0, b_0), v_0) \longrightarrow ((q_1, b_1), v_1) \longrightarrow ((q_2, b_2), v_2) \longrightarrow \ldots$$

Recall that whenever $p_{=x} \in \mu'(s)$, then $s = (q, 1)$ for some $q \in Q$. By (4.2), $v_i = v'_k$ for all i with $b_i = 1$. Using this, it is easy to construct from ρ'_{suffix} a γ-run ρ of $\mathcal{A}$. By (4.1), we obtain $\rho \models \varphi$. □

We conclude stating the exact complexity of model checking OCA and OCA(S) against $\flat$LTL$^{\downarrow}$:

Theorem 4.9. OCA-MC($\flat$LTL$^{\downarrow}$) *and* OCA(S)-MC($\flat$LTL$^{\downarrow}$) *are* NEXPTIME*-complete.*

5. Universal Model Checking

In this section, we consider the *universal* version of the model-checking problem ("Do all runs satisfy the formula?"). We write $\mathcal{A} \models_\forall \varphi$ if, for *all* parameter instantiations γ and infinite initialized γ-runs ρ of $\mathcal{A}$, we have $\rho \models \varphi$. With this, universal model checking for a class $\mathcal{C}$ of OCA(S,P,C) and a logic $\mathcal{L} \subseteq$ LTL$^{\downarrow}$ is defined as follows:

$\mathcal{C}$-MC$^{\forall}$($\mathcal{L}$)	
Input:	$\mathcal{A} = (Q, \mathcal{X}, q_{\mathsf{in}}, \Delta, \mu) \in \mathcal{C}$ and a sentence $\varphi \in \mathcal{L}$
Question:	Do we have $\mathcal{A} \models_\forall \varphi$?

We prove in the following that, contrasting the NEXPTIME-completeness of the problem OCA(S)-MC($\flat$LTL$^{\downarrow}$), we have undecidability for OCA-MC$^{\forall}$($\flat$LTL$^{\downarrow}$).

The proof goes a little detour over the co-flat fragment of LTL$^{\downarrow}$. An LTL$^{\downarrow}$ formula φ is *co-flat* if its negation $\neg\varphi$ is flat. We use co-$\flat$LTL$^{\downarrow}$ to denote the set of all co-flat LTL$^{\downarrow}$ formulas.

Theorem 5.1. OCA-MC(co-$\flat$LTL$^{\downarrow}$) *is undecidable.*

Proof. A close inspection of the proof of [DLS10, Theorem 17] (proving that OCA-MC(LTL$^{\downarrow}$) is undecidable) reveals that all formulas defined are in co-$\flat$LTL$^{\downarrow}$ so that the same proof yields undecidability of OCA-MC(co-$\flat$LTL$^{\downarrow}$). □

A trivial reduction (simply negate the input formula) from OCA-MC(co-$\flat$LTL$^{\downarrow}$) proves:

Theorem 5.2. OCA-MC$^{\forall}$($\flat$LTL$^{\downarrow}$) *is undecidable.*

6. Parametric Timed Automata

Although OCA(P) have mainly been used for deciding a model-checking problem for OCA, they constitute a versatile tool as well as a natural stand-alone model. In fact, parameterized extensions of a variety of system models have been explored recently, among them OCA with parameterized *updates* [HKOW09, GHOW12], and parametric timed automata [AHV93, BO17], where clocks can be compared with parameters (similarly to parameterized tests in OCA(P)).

In this section, we present an application of our results to the reachability problem for parametric timed automata [AHV93]. We refer to [And19] for an overview of recent advances in this field. We obtain that reachability in parametric timed automata with one parametric clock (1PTA) and closed guards (which corresponds to the discrete-time case) is in NEXPTIME, matching the known lower bound. Note that this result is subsumed by [BBLS15] where a NEXPTIME upper bound is shown over both discrete time and continuous time, using a different proof technique.

Let $\mathcal{C}$ be a finite set of *clock variables* (clocks, for short) ranging over the non-negative real numbers, and let $\mathcal{X}$ be a finite set of parameters ranging over the non-negative integers. We define *tests* π over $\mathcal{C}$ and $\mathcal{X}$ by the following grammar

$$\pi \ ::= \ \mathfrak{c} \bowtie x \ \mid \ \mathfrak{c} \bowtie d \ \mid \ \pi \wedge \pi$$

where $\mathfrak{c} \in \mathcal{C}$, $x \in \mathcal{X}$, $d \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. We use $\Pi(\mathcal{C}, \mathcal{X})$ to denote the set of all tests over $\mathcal{C}$ and $\mathcal{X}$. We say that $\pi \in \Pi(\mathcal{C}, \mathcal{X})$ is *closed* if $\bowtie \in \{\leq, =, \geq\}$ for all atomic tests $\mathfrak{c} \bowtie x$ and $\mathfrak{c} \bowtie d$ in π.

Definition 6.1. A *parametric timed automaton* (PTA) is a tuple $\mathcal{T} = (S, \mathcal{C}, \mathcal{X}, s_{\mathsf{in}}, \Delta)$, where

- S is a finite set of *states*,
- $\mathcal{C}$ is a finite set of *clock variables*,
- $\mathcal{X}$ is a finite set of *parameters*,
- $s_{\mathsf{in}} \in S$ is the *initial state*, and
- $\Delta \subseteq S \times \Pi(\mathcal{C}, \mathcal{X}) \times 2^{\mathcal{C}} \times S$ is the finite set of *transitions*.

We say that a clock $\mathfrak{c} \in \mathcal{C}$ is *parametric* if there exists some transition in Δ labeled with a test $\mathfrak{c} \bowtie x$ that compares $\mathfrak{c}$ with a parameter $x \in \mathcal{X}$; otherwise we say that $\mathfrak{c}$ is *non-parametric*. We say that $\mathcal{T}$ is *closed* if all tests occurring in transitions of $\mathcal{T}$ are closed.

Let $\Gamma_{\mathcal{T}} = (S \times (\mathbb{R}_{\geq 0})^{\mathcal{C}})$ be the set of configurations of $\mathcal{T}$, where $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers. In a configuration (s, ν), the first component s is the current state, and ν is the current clock valuation, i.e., a mapping $\mathcal{C} \to \mathbb{R}_{\geq 0}$ assigning some non-negative real number to each clock. Like for OCA, the semantics of a PTA $\mathcal{T}$ is given with respect to a parameter instantation $\gamma : \mathcal{X} \to \mathbb{N}$ in terms of a global transition relation $\longrightarrow_{\gamma} \subseteq \Gamma_{\mathcal{T}} \times \Gamma_{\mathcal{T}}$.

For two configurations (s, ν) and (s', ν'), we have $(s, \nu) \longrightarrow_{\gamma} (s', \nu')$ if there exist a time delay $\delta \in \mathbb{R}_{\geq 0}$ and some transition $(s, \pi, \lambda, s') \in \Delta$ such that $\nu + \delta \models_{\gamma} \pi$ and $\nu' = (\nu + \delta)[\lambda := 0]$. Here,

- $\nu + \delta$ is the clock valuation ν_{δ} defined by $\nu_{\delta}(\mathfrak{c}) := \nu(\mathfrak{c}) + \delta$ for every $\mathfrak{c} \in \mathcal{C}$,
- $(\nu+\delta)[\lambda := 0]$ is the clock valuation ν_{λ} defined by $\nu_{\lambda}(\mathfrak{c}) := 0$ if $\mathfrak{c} \in \lambda$, and $\nu_{\lambda}(\mathfrak{c}) := (\nu+\delta)(\mathfrak{c})$, otherwise, and
- $(\nu + \delta) \models_{\gamma} \pi$ if $(\nu + \delta)(\mathfrak{c}) \bowtie \gamma(x)$ for every atomic test $\mathfrak{c} \bowtie x$ in π, and $(\nu + \pi)(\mathfrak{c}) \bowtie d$ (for $d \in \mathbb{N}$) for every atomic test $\mathfrak{c} \bowtie d$ in π.

A γ-run ρ of $\mathcal{T}$ is a finite sequence of the form $(s_0, \nu_0) \longrightarrow_{\gamma} (s_1, \nu_1) \longrightarrow_{\gamma} \ldots \longrightarrow_{\gamma} (s_k, \nu_k)$, with $k \geq 0$. We say that ρ is initialized if $s_0 = s_{\mathsf{in}}$ and $\nu_0 \in \{0\}^{\mathcal{C}}$.

The reachability problem for PTA is defined analogously to that for OCA: given some parameter instantiation γ, we say that a state $s_f \in S$ is *γ-reachable* if there exists an initialized γ-run $(s_0, \nu_0) \longrightarrow_{\gamma} (s_1, \nu_1) \longrightarrow_{\gamma} \ldots \longrightarrow_{\gamma} (s_k, \nu_k)$ such that $s_k = s_f$. We say that s_f is *reachable*, written $\mathcal{T} \models \mathit{Reach}(s_f)$, if it is γ-reachable for some γ.

PTA-Reachability

Input:	A PTA $\mathcal{T} = (S, \mathcal{C}, \mathcal{X}, s_{\mathsf{in}}, \Delta)$ and $s_f \in S$
Question:	Do we have $\mathcal{T} \models \mathit{Reach}(s_f)$?

When introducing PTA in 1993 [AHV93], Alur, Henzinger, and Vardi proved that the problem PTA-Reachability is undecidable in general; however, they gave a decision procedure for PTA that use at most one parametric clock (and unboundedly many non-parametric clocks). The computational complexity of this decision procedure is non-elementary [BO17]. When the PTA uses only a single parametric clock and no non-parametric clock, it is known that the problem is NP-complete [Mil00]. Moreover, Bundala and Ouaknine proved that the reachability problem for *closed* PTA that use at most one parametric clock (and unboundedly many non-parametric clocks) can be reduced, in exponential time, to the reachability problem for OCA(P,C):

Proposition 6.2 [BO17]**.** *Given a closed PTA $\mathcal{T}$ with one parametric clock and a state s_f of $\mathcal{T}$, one can build, in exponential time, an* OCA(P,C) *$\mathcal{A}$ with a state q_f such that $\mathcal{T} \models \mathit{Reach}(s_f)$ iff $\mathcal{A} \models \mathit{Reach}(q_f)$ and $|\mathcal{A}| = \mathcal{O}(2^{|\mathcal{T}|})$.*

Based on this reduction, Bundala and Ouaknine improved the non-elementary decision procedure in [AHV93] for PTA with a single parametric clock to 2NEXPTIME [BO17] (although only for closed PTA). A NEXPTIME upper bound was then obtained by Benes et al. even in the continuous-time case [BBLS15].

Using Theorem 3.9, together with the hardness result appearing in [BO17], we obtain the following subcase:

Theorem 6.3. PTA-Reachability *for closed PTA with a single parametric clock is* NEXPTIME*-complete.*

7. Conclusion

In this paper, we established the precise complexity of model checking OCA and OCA(S) against flat freeze LTL. To do so, we established a tight link between OCA(P) and alternating two-way automata over words. Exploiting alternation further, it is likely that we can obtain positive results for one-counter games with parameterized tests. Another interesting issue for future work concerns model checking OCA(P), which, in this paper, is defined as the following question: Are there a parameter instantiation γ and a γ-run that satisfies the given formula? In fact, it also makes perfect sense to ask whether there is such a run for *all* parameter instantiations, in particular when requiring that all system runs satisfy a given property. It would also be worthwhile to characterize all the parameter valuations that allow one to reach a given target state.

References

[ABG+14] S. Akshay, B. Bollig, P. Gastin, M. Mukund, and K. Narayan Kumar. Distributed timed automata with independently evolving clocks. *Fundamenta Informaticae*, 130(4):377–407, 2014.

[AHV93] R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proceedings of STOC'93*, pages 592–601. ACM, 1993.

[And19] E. André. What's decidable about parametric timed automata? *STTT*, 21(2):203–219, 2019.

[BBLS15] N. Benes, P. Bezdek, K. G. Larsen, and J. Srba. Language emptiness of continuous-time parametric timed automata. In *Proceedings of ICALP'15, Part II*, volume 9135, pages 69–81. Springer, 2015.

[BDM+11] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27:1–27:26, 2011.

[BMOW08] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. On expressiveness and complexity in real-time model checking. In *Proceedings of ICALP'08, Part II*, volume 5126 of *LNCS*, pages 124–135. Springer, 2008.

[BO17] D. Bundala and J. Ouaknine. On parametric timed automata and one-counter machines. *Inf. Comput.*, 253:272–303, 2017.

[BQS17] B. Bollig, K. Quaas, and A. Sangnier. The complexity of flat freeze LTL. In *Proceedings of CONCUR'17*, volume 85 of *LIPIcs*, pages 33:1–33:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[Cac02] T. Cachat. Two-way tree automata solving pushdown games. In *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *LNCS*, pages 303–317. Springer, 2002.

[CC00] H. Comon and V. Cortier. Flatness is not a weakness. In *Proceedings of CSL'00*, volume 1862 of *LNCS*, pages 262–276. Springer, 2000.

[CCH$^+$16] Dmitry Chistikov, Wojciech Czerwinski, Piotr Hofman, Michal Pilipczuk, and Michael Wehar. Shortest paths in one-counter systems. In *Proceedings of FOSSACS'16*, volume 9634 of *Lecture Notes in Computer Science*, pages 462–478. Springer, 2016.

[DG09] S. Demri and R. Gascon. The effects of bounding syntactic resources on presburger LTL. *J. Log. Comput.*, 19(6):1541–1575, 2009.

[DK08] C. Dax and F. Klaedtke. Alternation elimination by complementation. In *Proceedings of LPAR'08*, volume 5330 of *LNCS*, pages 214–229. Springer, 2008.

[DL09] S. Demri and R. Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[DLN07] S. Demri, R. Lazic, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. *Inf. Comput.*, 205(1):2–24, 2007.

[DLS10] S. Demri, R. Lazic, and A. Sangnier. Model checking memoryful linear-time logics over one-counter automata. *Theor. Comput. Sci.*, 411(22-24):2298–2316, 2010.

[DS10] S. Demri and A Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In *Proceedings of FoSSaCS'10*, volume 6014 of *LNCS*, pages 176–190. Springer, 2010.

[FLQ15] S. Feng, M. Lohrey, and K. Quaas. Path checking for MTL and TPTL over data words. In *Proceedings of DLT'15*, volume 9168 of *LNCS*, pages 326–339. Springer, 2015.

[Fre03] T. French. Quantified propositional temporal logic with repeating states. In *Proceedings of TIME-ICTL'03*, pages 155–165. IEEE Computer Society, 2003.

[Gal76] Z. Galil. Hierarchies of complete problems. *Acta Inf.*, 6:77–88, 1976.

[GHOW10] S. Göller, C. Haase, J. Ouaknine, and J. Worrell. Model checking succinct and parametric one-counter automata. In *Proceedings of ICALP'10, Part II*, volume 6199 of *LNCS*, pages 575–586. Springer, 2010.

[GHOW12] S. Göller, C. Haase, J. Ouaknine, and J. Worrell. Branching-time model checking of parametric one-counter automata. In *Proceedings of FOSSACS'12*, volume 7213 of *LNCS*, pages 406–420. Springer, 2012.

[GL13] S. Göller and M. Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013.

[Haa12] C. Haase. *On the complexity of model checking counter automata.* PhD thesis, University of Oxford, 2012.

[HKOW09] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *Proceedings of CONCUR'09*, volume 5710 of *LNCS*, pages 369–383. Springer, 2009.

[HLMT16] P. Hofman, S. Lasota, R. Mayr, and P. Totzke. Simulation problems over one-counter nets. *Logical Methods in Computer Science*, 12(1), 2016.

[KV00] O. Kupferman and M. Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Proceedings of CAV'00*, volume 1855 of *LNCS*, pages 36–52. Springer, 2000.

[LLT05] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Proceedings of RTA'05*, volume 3467 of *LNCS*, pages 308–322. Springer, 2005.

[LMO+16] A. Lechner, R. Mayr, J. Ouaknine, A. Pouly, and J. Worrell. Model checking flat freeze LTL on one-counter automata. In *Proceedings of CONCUR'16*, volume 59 of *LIPIcs*, pages 29:1–29:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[LP05] A. Lisitsa and I. Potapov. Temporal logic with predicate lambda-abstraction. In *Proceedings of TIME'05*, pages 147–155. IEEE Computer Society, 2005.

[Mil00] J. S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In Nancy A. Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control, Third International Workshop, HSCC 2000, Pittsburgh, PA, USA, March 23-25, 2000, Proceedings*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer, 2000.

[Ser06] O. Serre. Parity games played on transition graphs of one-counter processes. In *Proceedings of FoSSaCS'06*, volume 3921 of *LNCS*, pages 337–351. Springer, 2006.

[Var98] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 628–641. Springer, 1998.

[VP75] L. G. Valiant and M. S. Paterson. Deterministic one-counter automata. *Journal of Computer and System Sciences*, 10(3):340–350, 1975.

Appendix A. Proof Details for Lemma 3.3

We have to show that $L(\mathcal{T}) = \{w \in \mathcal{W}_\mathcal{X} \mid q_f \text{ is } \gamma_w\text{-reachable}\}$.

$\supseteq$: Consider $w = a_0a_1a_2 \cdots \in \mathcal{W}_\mathcal{X}$ such that q_f is γ-reachable where $\gamma = \gamma_w$. Then, there is an initialized run $\rho = (q_0, v_0) \longrightarrow_\gamma \cdots \longrightarrow_\gamma (q_n, v_n)$ of $\mathcal{A}$ with $q_n = q_f$. Suppose that the global transitions are witnessed by transitions $(q_i, \mathsf{op}_i, q_{i+1}) \in \Delta$, for $i \in \{0, \ldots, n-1\}$. We construct a (tree) run χ of $\mathcal{T}$ on w. For $v \in \mathbb{N}$, let $\mathsf{enc}(v)$ denote the unique position $i \in \mathbb{N}$ of w such that $a_i = \Box$ and $v = |a_0 \ldots a_{i-1}|_\Box$. Thus, $\mathsf{enc}(v)$ is the position in w that corresponds to counter value v. Moreover, for $x \in \mathcal{X}$, let $\mathsf{pos}(x)$ be the unique position i such that $a_i = x$. Thus, we have $\mathsf{enc}(\gamma(x)) < \mathsf{pos}(x) < \mathsf{enc}(\gamma(x) + 1)$.

Starting from the root of χ, which is labeled with $(s_{\mathsf{in}}, 0)$, we will first generate a finite branch that takes care of simulating ρ. More precisely, we create a linear sequence of nodes by replacing a global transition $(q_i, v_i) \longrightarrow_\gamma (q_{i+1}, v_{i+1})$, with $i \in \{0, \ldots, n-1\}$, by a sequence of configurations (proof obligations) of $\mathcal{T}$ as follows (here, $\Longrightarrow$ denotes an edge of χ, possibly with the letter read at the corresponding transition):

- If $\mathsf{op}_i \in \{0\} \cup \mathrm{Tests}(\mathcal{X})$:
$$(q_i, \mathsf{enc}(v_i)) \Longrightarrow (q_{i+1}, \mathsf{enc}(v_{i+1}))$$
- If $\mathsf{op}_i = +1$:
$$\begin{aligned}(q_i, \mathsf{enc}(v_i)) &\overset{\Box}{\Longrightarrow} (\mathsf{right}(q_{i+1}), \mathsf{enc}(v_i) + 1)\\ &\overset{x_1}{\Longrightarrow} (\mathsf{right}(q_{i+1}), \mathsf{enc}(v_i) + 2)\\ &\overset{x_2}{\Longrightarrow} \cdots\\ &\overset{x_n}{\Longrightarrow} (\mathsf{right}(q_{i+1}), \mathsf{enc}(v_{i+1}))\\ &\overset{\Box}{\Longrightarrow} (q_{i+1}, \mathsf{enc}(v_{i+1}))\end{aligned}$$
where $x_i \neq \Box$ are parameters.

- If $\mathsf{op}_i = -1$:

$$\begin{aligned}(q_i, \mathsf{enc}(v_i)) &\Longrightarrow (\mathsf{left}(q_{i+1}), \mathsf{enc}(v_i) - 1)\\ &\Longrightarrow \cdots\\ &\Longrightarrow (\mathsf{left}(q_{i+1}), \mathsf{enc}(v_{i+1}) + 1)\\ &\Longrightarrow (q_{i+1}, \mathsf{enc}(v_{i+1}))\end{aligned}$$

Note that the resulting finite branch is both well-defined (as w is a parameter word) and accepting (due to the transition $q_n \xrightarrow{\Box} \texttt{true}$ of $\mathcal{T}$). Now, for all $i \in \{0, \ldots, n-1\}$ such that $\mathsf{op}_i \in \mathrm{Tests}(\mathcal{X}) \setminus \{=0\}$, we append another branch rooted at the node that arises from position i and is labeled with $(q_i, \mathsf{enc}(v_i))$:

- If $\mathsf{op}_i = (=x)$:

$$\begin{aligned}(q_i, \mathsf{enc}(v_i)) &\overset{\Box}{\Longrightarrow} (\mathsf{present}(x), \mathsf{enc}(v_i) + 1)\\ &\overset{x_1}{\Longrightarrow} (\mathsf{present}(x), \mathsf{enc}(v_i) + 2)\\ &\overset{x_2}{\Longrightarrow} \cdots\\ &\overset{x}{\Longrightarrow} (\mathsf{present}(x), \mathsf{pos}(x))\end{aligned}$$

  where $x_i \neq x$ are parameters. Since ρ is a run, we have $v_i = \gamma(x)$ and, therefore, $\mathsf{enc}(v_i) < \mathsf{pos}(x) < \mathsf{enc}(v_i + 1)$. Thus, this finite branch is well-defined and corresponds to a sequence of transitions of $\mathcal{T}$. It is accepting, as $a_{\mathsf{pos}(x)} = \Box$ and there is a transition $\mathsf{present}(x) \xrightarrow{x} \texttt{true}$.

- If $\mathsf{op}_i = (<x)$:

$$\begin{aligned}(q_i, \mathsf{enc}(v_i)) &\overset{\Box}{\Longrightarrow} (\mathsf{search}^+(x), \mathsf{enc}(v_i) + 1) \Longrightarrow \cdots \Longrightarrow (\mathsf{search}^+(x), \mathsf{enc}(v_i + 1))\\ &\overset{\Box}{\Longrightarrow} (\mathsf{search}(x), \mathsf{enc}(v_i + 1) + 1) \Longrightarrow \cdots \Longrightarrow (\mathsf{search}(x), \mathsf{pos}(x))\\ &\overset{x}{\Longrightarrow} (\checkmark_x, \mathsf{pos}(x) + 1) \Longrightarrow (\checkmark_x, \mathsf{pos}(x) + 2) \Longrightarrow \cdots\end{aligned}$$

  Since ρ is a run, $v_i < \gamma(x)$. Therefore, $\mathsf{enc}(v_i + 1) < \mathsf{pos}(x)$. Thus, the branch corresponds to the transition relation of $\mathcal{T}$. It is infinite, since w is a parameter word, and accepting, as $\checkmark_x \in S_f$ (i.e., $\checkmark_x$ is an accepting state).

- The case $\mathsf{op}_i = (>x)$ is similar.

Finally, we add the branches that check that the input word is a parameter word. For every $x \in \mathcal{X}$, there is a branch $(s_{\mathsf{in}}, 0) \overset{\Box}{\Longrightarrow} (\mathsf{search}(x), 0) \Longrightarrow (\mathsf{search}(x), 1) \Longrightarrow \cdots \Longrightarrow (\mathsf{search}(x), \mathsf{pos}(x)) \overset{x}{\Longrightarrow} (\checkmark_x, \mathsf{pos}(x) + 1) \Longrightarrow (\checkmark_x, \mathsf{pos}(x) + 2) \Longrightarrow \cdots$ starting at the root of χ. This infinite branch is well-defined, as w is indeed a parameter word. Again, it is accepting since $\checkmark_x \in S_f$.

This concludes the construction of the accepting run χ of $\mathcal{T}$ on w.

$\subseteq$: For the converse direction, suppose that $w = a_0 a_1 a_2 \cdots \in L(\mathcal{T})$. Thus, there is an accepting run χ of $\mathcal{T}$ on w whose root is labeled with $(s_{\mathsf{in}}, 0)$. We assume that χ is minimal in the sense that removing a subtree is no longer a run on w. The initial transition $s_{\mathsf{in}} \xrightarrow{\Box} (q_{\mathsf{in}}, 0) \wedge \bigwedge_{x \in \mathcal{X}} (\mathsf{search}(x), +1)$ checks that the first letter is $\Box$. It also spawns a copy for each parameter x, checking that letter x occurs exactly once. Thus, we can assume that $w \in \mathcal{W}_{\mathcal{X}}$ (i.e., w is a parameter word) so that we may use the definitions $\mathsf{enc}(v)$ and $\mathsf{pos}(x)$ from the other direction. Moreover, the root has a successor u labeled with $(q_{\mathsf{in}}, 0)$. Browsing through χ, we successively build a γ_w-run ρ. We start at u and initialize ρ to $(q_{\mathsf{in}}, 0)$.

Suppose that we already constructed the run $\rho = (q_{\mathsf{in}}, 0) \longrightarrow_{\gamma_w} \cdots \longrightarrow_{\gamma_w} (q, v)$ up to some (q, i)-labeled node u. As an invariant, we maintain that $i = \mathsf{enc}(v)$. If u does not have any successor, then $q = q_f$, because the only transition that allows us to accept is $q_f \xrightarrow{\Box} \mathtt{true}$. In that case, q_f is γ_w-reachable, witnessed by ρ. Now, suppose that u has some successors. By the definition of δ, there is a transition $(q, \mathsf{op}, q') \in \Delta$ of $\mathcal{A}$ such that one of the following hold:

- We have $\mathsf{op} = 0$ and there is a successor u' of u labeled (q', i). Obviously, we have $(q, v) \longrightarrow_{\gamma_w} (q', v)$ and we set $\rho' := \rho \longrightarrow_{\gamma_w} (q', v)$.
- We have $\mathsf{op} = +1$ and, starting from u, there is a path of the form $(q, i) \overset{\Box}{\Longrightarrow} (\mathsf{right}(q'), i+1) \Longrightarrow \ldots \Longrightarrow (\mathsf{right}(q'), \mathsf{enc}(v+1)) \overset{\Box}{\Longrightarrow} (q', \mathsf{enc}(v+1))$. Let u' be the last node of that path. Clearly, it holds $(q, v) \longrightarrow_{\gamma_w} (q', v+1)$, and we set $\rho' := \rho \longrightarrow_{\gamma_w} (q', v+1)$.
- The case $\mathsf{op} = -1$ is similar.
- We have $\mathsf{op} = {=}0$ and there is a successor u' of u labeled $(q', 0)$ (due to $q \xrightarrow{\mathsf{first?}} (q', 0)$). Then, $v = 0$ and $(q, v) \longrightarrow_{\gamma_w} (q', v)$ so that we can set $\rho' := \rho \longrightarrow_{\gamma_w} (q', v)$.
- We have $\mathsf{op} = ({=}x)$ and, starting from u, there are a successor labeled with (q', v) and a finite branch of the following form:
  $$(q, i) = (q, \mathsf{enc}(v)) \Longrightarrow (\mathsf{present}(x), \mathsf{enc}(v) + 1) \Longrightarrow \cdots \Longrightarrow (\mathsf{present}(x), \mathsf{pos}(x))$$
  By the latter, we have $\mathsf{enc}(v) < \mathsf{pos}(x) < \mathsf{enc}(v+1)$, which implies $v = \gamma_w(x)$. Thus, $(q, v) \longrightarrow_{\gamma_w} (q', v)$ and we set $\rho' := \rho \longrightarrow_{\gamma_w} (q', v)$.
- We have $\mathsf{op} = {<}\,x$ and, starting from u, there are a successor u' labeled with (q', v) and another (accepting) infinite branch of the following form:
  $$\begin{aligned}(q, i) = (q, \mathsf{enc}(v)) &\overset{\Box}{\Longrightarrow} (\mathsf{search}^+(x), \mathsf{enc}(v) + 1) \Longrightarrow \cdots \Longrightarrow (\mathsf{search}^+(x), \mathsf{enc}(v+1)) \\ &\overset{\Box}{\Longrightarrow} (\mathsf{search}(x), \mathsf{enc}(v+1) + 1) \Longrightarrow \cdots \Longrightarrow (\mathsf{search}(x), \mathsf{pos}(x)) \\ &\overset{x}{\Longrightarrow} (\checkmark_x, \mathsf{pos}(x) + 1) \Longrightarrow (\checkmark_x, \mathsf{pos}(x) + 2) \Longrightarrow \cdots\end{aligned}$$
  By the latter, we have $\mathsf{enc}(v+1) < \mathsf{pos}(x)$, which implies $v < \gamma_w(x)$. Thus, $(q, v) \longrightarrow_{\gamma_w} (q', v)$ and we set $\rho' := \rho \longrightarrow_{\gamma_w} (q', v)$.
- The case $\mathsf{op} = {>}\,x$ is similar.

We constructed a longer run ρ' up to node u' and continue this procedure. Since χ is accepting, we eventually find a γ_w-run of $\mathcal{A}$ ending in a configuration (q, i) with $q = q_f$. Thus, q_f is γ_w-reachable.

Appendix B. Proof Details for Theorem 3.8

We give more detailed arguments why condition (4) in the proof of Theorem 3.8 can be verified in polynomial time.

Recall from (4) that, given two configurations (q, v) and (q', v') with $v, v' \in D$, we need to verify whether there exists a run $(q, v) \longrightarrow_{\gamma}^{*} (q', v')$ such that, (strictly) between the configurations (q, v) and (q', v'), the counter never takes a value in D. Assume $v = d_i$ for some $i \in \{0, \ldots, n, n+1\}$. We have to distinguish the cases $v' = d_i$, $v' = d_{i-1}$ (if $i \geq 1$), and $v' = d_{i+1}$ (if $i \leq n$). Moreover, if $v' = d_i = v$, the run can proceed below or above v.

Towards the case $v' = d_{i+1}$, we transform $\mathcal{A}$ into an OCA $\mathcal{M}$ as follows:

- remove all transitions where the label op is of the form
  - $=0$,
  - $=x$ for some $x \in \mathcal{X}$,
  - $>x_j$ for some $j \in \{i+1,\dots,n\}$, or
  - $<x_j$ for some $j \in \{1,\dots,i\}$,
- replace op in all transitions by true, if op is of the form
  - $<x_j$ for some $j \in \{i+1,\dots,n\}$, or
  - $>x_j$ for some $j \in \{1,\dots,i\}$,
- keep all other transitions.

It is easy to see that we have $(q,v) \longrightarrow_{\gamma}^{*} (q',v')$ in $\mathcal{A}$ such that (strictly) between the configurations (q,v) and (q',v') the counter never takes a value in D iff there is a (v,v')-run from (q,v) to (q',v') in $\mathcal{M}$. The latter condition can be tested in polynomial time according to Lemma 3.7.

The other cases are handled similarly.

Appendix C. Proof Details for Lemma 4.1

Soundness. Assume there exist some parameter instantiation γ' and some finite γ'-run ρ' in $\mathcal{A}'$ ending in $\hat{s}$. We distinguish the only two possible cases:

(1) The suffix of ρ' is of the form
$$(t,v) \longrightarrow_{\gamma'} (t_1,v) \longrightarrow_{\gamma'} (t_2,v) \longrightarrow_{\gamma'} \dots \longrightarrow_{\gamma'} (t_n,v) \longrightarrow_{\gamma'} (\hat{s},v)$$
for some $t \in T$ and $v \in \mathbb{N}$. Recall from the definition of Δ that the corresponding sequence of transitions is $(t,0,t_1)$, $(t_i,>x_i,t_{i+1})$ for all $1 \le i < n$, and $(t_n,>x_n,\hat{s})$. This implies $v > \gamma'(x)$ for every $x \in \mathcal{X}$. By definition of T, there exists some infinite run $\hat{\rho}$ in $\mathcal{A}$ starting in $(t,0)$ and visiting q_f infinitely often. Further, $\hat{\rho}$ does not contain any transition originating from a transition in $\mathcal{A}$ with an operation of the form $=0$, $=x$ or $<x$ for some $x \in \mathcal{X}$. Hence, using the parameter instantiation defined by $\gamma(x) = \gamma'(x)$ for every $x \in \mathcal{X}$, we can use the corresponding sequence of transitions of $\hat{\rho}$ to construct an infinite γ-run of $\mathcal{A}$ that starts in (t,v) and visits q_f infinitely often: note that even tests of the form $>x$ (that were replaced in $\hat{\rho}$ by 0) are satisfied because $v > \gamma(x)$ for all $x \in \mathcal{X}$. Concatenating this run with the prefix of ρ' (until (t,v)) yields the infinite γ-run we aim to construct.

(2) The last transition in ρ' is of the form $(\hat{q}_f,v) \longrightarrow_{\gamma'} (\hat{s},v)$, for some $v \in \mathbb{N}$, resulting from the transition $(\hat{q}_f,=y,\hat{s}) \in \Delta'$. By definition of $\mathcal{A}'$, ρ' consists of a prefix run of the form
$$(q_{\mathsf{in}},0) \longrightarrow_{\gamma'}^{*} (q_f,v) \longrightarrow_{\gamma'} (s,v),$$
where the last transition results from the transition $(q_f,=y,s) \in \Delta'$, which implies $\gamma'(y) = v$, and some suffix run of the form
$$(s,v) \longrightarrow_{\gamma'} (\hat{q}_1,v_1) \longrightarrow_{\gamma'} \cdots \longrightarrow_{\gamma'} (\hat{q}_k,v_k) \longrightarrow_{\gamma'} (\hat{q}_f,v') \longrightarrow_{\gamma'} (\hat{s},v')$$
where the last transition again results from the transition $(\hat{q}_f,=y,s) \in \Delta$, which implies $v' = v$. Define γ by $\gamma(x) = \gamma'(x)$ for every $x \in \mathcal{X}$. The construction of $\mathcal{A}'$ implies that in $\mathcal{A}$ there must be finite γ-runs ρ_{prefix} and ρ_{suffix} of the form
$$\rho_{\mathsf{prefix}} = (q_{\mathsf{in}},0) \longrightarrow_{\gamma}^{*} (q_f,v)$$

and

$$\rho_{\mathsf{suffix}} = (q_f, v) \longrightarrow_\gamma (q_1, v_1) \longrightarrow_\gamma \cdots \longrightarrow_\gamma (q_k, v_k) \longrightarrow_\gamma (q_f, v),$$

yielding an infinite γ-run defined by $\rho_{\mathsf{prefix}}(\rho_{\mathsf{suffix}})^\omega$, visiting q_f infinitely often.

Completeness. Assume there exist some parameter instantiation γ and some infinite γ-run ρ in $\mathcal{A}$ visiting q_f infinitely often. We distinguish two (not necessarily disjoint) cases:

(1) Run ρ visits q_f infinitely often with the same counter value v, for some $v \in \mathbb{N}$. Define γ' by $\gamma'(x) = \gamma(x)$ for all $x \in \mathcal{X}$, and $\gamma'(y) = v$. It can be easily seen that, from ρ, we can construct a finite γ'-run ρ' that ends in $\hat{s}$: when, in ρ, the configuration (q_f, v) is visited for the first time, we use the transition $(q_f, =y, p)$, from which $\mathcal{A}'$ can simulate ρ until it meets $(\hat{q}_f, v)$ for the next time.

(2) Run ρ visits q_f infinitely often with unbounded counter values. Define $M = \max\{\gamma(x) \mid x \in \mathcal{X}\}$ to be the maximum value that a parameter is instantiated with. We distinguish two cases:

(a) Run ρ visits infinitely often a counter value $m \leq M$. Then, ρ must necessarily contain a subrun of the form

$$(q, m) \longrightarrow_\gamma^* (q_f, v) \longrightarrow_\gamma^* (q, m)$$

for some $q \in Q$ and $v \in \mathbb{N}$. Note that, then, there must necessarily exist an infinite γ-run that visits (q_f, v) infinitely often, so that we can proceed with case (1) above.

(b) There exists some configuration (t, v) in ρ, where $v > M$, such that the infinite γ-run ρ' starting in (t, v) never visits the counter value M anymore. Hence, ρ' cannot contain any transitions with operations of the form $=0$, $=x$, or $<x$ for some $x \in \mathcal{X}$, and transition guards of the form $>x$ for some $x \in \mathcal{X}$ are satisfied. Clearly, ρ' is visiting q_f infinitely often, and hence $t \in T$. It is easily seen that, from ρ, we can construct a finite γ'-run (where the fresh parameter y is indeed never used) that ends in $\hat{s}$ using the sequence of transitions $(t, 0, r_1), (t_1, > x_1, r_2), \ldots, (t_n, > x_n, \hat{s})$.