Model Checking Flat Freeze LTL on One-Counter Automata

Antonia Lechner[1], Richard Mayr[2], Joël Ouaknine†[3], Amaury Pouly†[4], and James Worrell[5]

1 **Department of Computer Science, University of Oxford, UK**
antonia.lechner@cs.ox.ac.uk
2 **School of Informatics, LFCS, University of Edinburgh, UK**
http://homepages.inf.ed.ac.uk/rmayr/
3 **Department of Computer Science, University of Oxford, UK**
joel.ouaknine@cs.ox.ac.uk
4 **Department of Computer Science, University of Oxford, UK**
amaury.pouly@cs.ox.ac.uk
5 **Department of Computer Science, University of Oxford, UK**
james.worrell@cs.ox.ac.uk

Abstract

Freeze LTL is a temporal logic with registers that is suitable for specifying properties of data words. In this paper we study the model checking problem for Freeze LTL on one-counter automata. This problem is known to be undecidable in full generality and PSPACE-complete for the special case of deterministic one-counter automata. Several years ago, Demri and Sangnier investigated the model checking problem for the flat fragment of Freeze LTL on several classes of counter automata and posed the decidability of model checking flat Freeze LTL on one-counter automata as an open problem. In this paper we resolve this problem positively, utilising a known reduction to a reachability problem on one-counter automata with parameterised equality and disequality tests. Our main technical contribution is to show decidability of the latter problem by translation to Presburger arithmetic.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases One-counter automata, disequality tests, reachability, freeze LTL, Presburger arithmetic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.29

1 Introduction

Runs of infinite-state machines, such as counter automata, can naturally be seen as *data words*, that is, sequences in which each position is labelled by a letter from a finite alphabet and a datum from an infinite domain. Freeze LTL is an extension of Linear Temporal Logic with registers (or variables) and a binding mechanism, which has been introduced to specify properties of data words [3, 4, 8, 11]. The registers allow to compare data at different positions along the same computation.

An example of a freeze LTL formula is

$$\mathsf{F}(v \wedge {\downarrow_r} \mathsf{XF}(v \wedge {\uparrow_r}))\,. \tag{1}$$

† Supported by ERC grant AVS-ISS (648701).

27th International Conference on Concurrency Theory (CONCUR 2016).
Editors: Josée Desharnais and Radha Jagadeesan; Article No. 29; pp. 29:1–29:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Evaluated on a run of a one-counter automaton, this formula is true if and only if there are at least two different positions in the run which both have control state v and the same counter value. Intuitively the operator $\downarrow_r$ binds the current counter value to register r, while the operator $\uparrow_r$ tests whether the current counter value is equal to the content of register r.

This paper concerns the model checking problem for Freeze LTL on one-counter automata. It is known that this problem is undecidable in general, but PSPACE-complete if one restricts to *deterministic* one-counter automata [5]. Rather than restricting the class of one-counter automata, one can seek to identify decidable syntactic fragments of Freeze LTL. This approach was pursued in [6], which studied the *flat* fragment of Freeze LTL. The flatness condition places restrictions on the occurrence of the binding construct $\downarrow_r$ in relation to the until operator (see Section 2.2 for details). For example, in a flat formula in negation normal form the binding operator $\downarrow_r$ can occur within the scope of F but not G. (Thus formula (1) is flat.) The flatness restriction for Freeze LTL has a similar flavour to the flatness restriction for constraint LTL [2] and for Metric Temporal Logic [1].

Demri and Sangnier [6] considered the decidability of model checking flat Freeze LTL across a range of different counter-machine models. For one-counter automata they showed decidability of model checking for a certain fragment of flat Freeze LTL and they left open the problem of model checking flat Freeze LTL in general.

The approach taken in [6] was to reduce the model checking problem for fragments of Freeze LTL on a class of counter automata to a repeated reachability problem in counter automata of the same class with certain kinds of parameterised tests. In particular, under their approach the model checking problem for flat Freeze LTL on one-counter automata reduces to a repeated reachability problem for the class of one-counter automata extended with parameterised equality and disequality tests. This problem considers one-counter automata whose transitions may be guarded by equality or disequality tests that compare the counter value to integer-valued parameters, and it asks whether there exist parameter values such that there is an infinite computation that visits an accepting location infinitely many times. The main technical contribution of this paper is to show decidability of the latter problem by reduction to the decision problem for Presburger arithmetic.

A related work is [9], which considers one-counter automata with parameterised updates and equality tests. It is shown in [9] that reachability in this model is inter-reducible with the satisfiability problem for quantifier-free Presburger arithmetic with divisibility, and therefore decidable. In contrast to [9], in the present paper the counter automata do not have parameterised updates but they do have parameterised disequality tests. The results in this paper do not appear to be straightforwardly reducible to those of [9] nor *vice versa*. Both reachability problems can be seen as special cases of a long-standing open problem identified by Ibarra *et al.* [10] which asks to decide reachability on a class of automata with a single integer-valued counter, sign tests, and parameterised updates.

2 Preliminaries

2.1 One-Counter Automata with Equality and Disequality Tests

We consider automata with an associated single counter, which ranges over the nonnegative integers, and with both equality and disequality tests on counter values. Formally, a *one-counter automaton* (1-CA) is a tuple $\mathcal{C} = (V, E, \lambda, \tau)$, where V is a finite set of *states*, $E \subseteq V \times V$ is a finite set of *edges* between states, $\lambda : E \to Op$ labels each edge with an element from $Op = \{\mathrm{add}(a) : a \in \mathbb{Z}\} \cup \{\mathrm{eq}(a) : a \in \mathbb{N}\}$, and $\tau : V \to 2^{\mathbb{N}}$ maps each state v to a finite set $\tau(v)$ of *invalid counter values* at state v. Intuitively the operation add(a) adds a

to the counter and $\mathrm{eq}(a)$ tests the counter for equality with a. The association of invalid counter values with each state can be seen as a type of disequality test. This feature is not present in classical presentations of 1-CA, but we include it here to facilitate our treatment of freeze LTL.

For any edge $e = (v, v')$ with $\lambda(e) = \mathrm{op}(a)$, define $\mathrm{start}(e) = v$, $\mathrm{end}(e) = v'$, and $\mathrm{weight}(e) = a$ if $\mathrm{op} = \mathrm{add}$, $\mathrm{weight}(e) = 0$ if $\mathrm{op} = \mathrm{eq}$. A *path* γ is a finite word on the alphabet E: $\gamma = e_1 \cdots e_n$ such that $\mathrm{end}(e_i) = \mathrm{start}(e_{i+1})$ for every i. The *length* of γ, denoted $|\gamma|$, is n. The *state sequence* of γ is $\mathrm{start}(e_1)$, $\mathrm{end}(e_1)$, $\mathrm{end}(e_2)$, …, $\mathrm{end}(e_n)$. The *start* of γ, denoted $\mathrm{start}(\gamma)$, is $\mathrm{start}(e_1)$. The *end* of γ, denoted $\mathrm{end}(\gamma)$, is $\mathrm{end}(e_n)$. A path is *simple* if it contains no repeated vertices. The *weight* of γ, denoted by $\mathrm{weight}(\gamma)$, is $\sum_{i=1}^{n} \mathrm{weight}(e_i)$. A *subpath* γ' of γ is any factor of γ: $\gamma' = e_i e_{i+1} \dots e_j$. If γ and γ' are two paths such that $\mathrm{end}(\gamma) = \mathrm{start}(\gamma')$, $\gamma\gamma'$ is the concatenation of both paths.

A *cycle* ω is a path such that $\mathrm{start}(\omega) = \mathrm{end}(\omega)$. A cycle is *simple* if there are no repeated vertices except for the starting point, which appears twice. A cycle is *positive* if it has positive weight, *negative* if it has negative weight and *zero-weight* if it has weight zero. We denote by $\omega^k = \underbrace{\omega\omega\cdots\omega}_{k \text{ times}}$ the sequence of k iterations of cycle ω.

A *configuration* of a 1-CA $\mathcal{C} = (V, E, \lambda, \tau)$ is a pair (v, c) with $v \in V$ and $c \in \mathbb{Z}$. Intuitively, (v, c) corresponds to the situation where the 1-CA is in state v with counter value c. Since counter values range over the nonnegative integers, configurations (v, c) with $c \geqslant 0$ are called *valid*, otherwise they are *invalid*. The transition relation E between states with guards λ and τ induces an unlabelled transition relation between configurations: for any two configurations (v, c) and (v', c'), there is a transition $(v, c) \longrightarrow (v', c')$ if and only if there is an edge $e \in E$ with $\lambda(e) = \mathrm{op}(a)$ for some a, $\mathrm{start}(e) = v$, $\mathrm{end}(e) = v'$, and $\mathrm{weight}(e) = c' - c$. We will sometimes write $(v, c) \xrightarrow{e} (v', c')$ for such a transition. The transition is *valid* if $c, c' \geqslant 0$ and $c \notin \tau(v)$, and also $c = a$ if $\mathrm{op} = \mathrm{eq}$. Otherwise it is *invalid*.

A *computation* π is a (finite or infinite) sequence of transitions:

$$\pi = (v_1, c_1) \longrightarrow (v_2, c_2) \longrightarrow (v_3, c_3) \longrightarrow \cdots$$

We write $|\pi|$ for the length of π. If $(v_1, c_1) \xrightarrow{e_1} (v_2, c_2) \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} (v_n, c_n)$ is a finite computation, we will also write it as $(v_1, c_1) \xrightarrow{\gamma}{}^{*} (v_n, c_n)$, where $\gamma = e_1 e_2 \cdots e_{n-1}$, or simply $(v_1, c_1) \longrightarrow^{*} (v_n, c_n)$. A computation π is *valid* if all transitions in the sequence are valid, otherwise it is *invalid*. If π is invalid, an *obstruction* is a configuration (v_i, c_i) such that $(v_i, c_i) \longrightarrow (v_{i+1}, c_{i+1})$ is an invalid transition, or, if π is of finite length $n - 1$, $i = n$ and $c_i < 0$.

Given a path γ and a counter value $c \in \mathbb{Z}$, the *path computation* $\gamma(c)$ is the (finite) computation starting at $(\mathrm{start}(\gamma), c)$ and following the sequence of transitions that correspond to the edges in γ.

A *one-counter automaton with parameterised tests* is a tuple (V, E, X, λ, τ), where V, E and λ are defined as before, X is a set of nonnegative integer variables, $Op = \{\mathrm{add}(a) : a \in \mathbb{Z}\} \cup \{\mathrm{eq}(a), \mathrm{eq}(x) : a \in \mathbb{N}, x \in X\}$, and $\tau : V \to 2^{\mathbb{N} \cup X}$. Note that $\tau(v)$ is still required to be finite for each $v \in V$.

For a given 1-CA $\mathcal{C} = (V, E, \lambda, \tau)$, an initial configuration (v, c) and a target configuration (v', c'), the *reachability* problem asks if there is a valid computation from (v, c) to (v', c'). When $\mathcal{C}$ has sets $F_1, \dots, F_n \subseteq V$ of final states and an initial configuration (v, c), the *generalised repeated reachability* problem asks if there is a valid infinite computation from (v, c) which visits at least one state in each F_i infinitely often.

For a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with parameterised tests with given initial configuration (v, c) and target configuration (v', c'), the *reachability* problem asks if there exist values for the parameters such that there is a computation from (v, c) to (v', c'). Similarly, in the case where $\mathcal{C}$ has sets $F_1, \dots, F_n \subseteq V$ of final states and an initial configuration (v, c), the *generalised repeated reachability* problem asks if there exist values for the parameters such that substituting these values satisfies the generalised repeated reachability condition above.

2.2 Model Checking Freeze LTL on One-Counter Automata

Freeze LTL [5] is an extension of Linear Temporal Logic that can be used to specify properties of data words. Freeze LTL is one of a variety of formalisms that arise by augmenting a temporal or modal logic with variable binding. Given a finite alphabet Σ and set of *registers* R, the formulas of Freeze LTL are given by the following grammar

$$\varphi \quad ::= \quad a \quad | \quad \uparrow_r \quad | \quad \neg\varphi \quad | \quad \varphi \vee \varphi \quad | \quad \mathsf{X}\,\varphi \quad | \quad \varphi \,\mathsf{U}\, \varphi \quad | \quad \downarrow_r \varphi\,,$$

where $a \in \Sigma$ and $r \in R$. We write LTL$^{\downarrow}$ for the set of formulas of Freeze LTL. A *sentence* is a formula in which each occurrence of a subformula $\uparrow_r$ is in the scope of an operator $\downarrow_r$ (for the same register r).

In general, formulas of LTL$^{\downarrow}$ are interpreted over data words. In this paper we are interested in a particular kind of data word, namely those arising from valid computations of 1-CA, and we directly define the semantics of LTL$^{\downarrow}$ over such computations (assuming that the alphabet Σ is the set of control locations of the 1-CA). In this context $\downarrow_r$ can be seen as a binding construct that stores in register r the counter value at the current position in a computation, while $\uparrow_r$ tests whether the counter value at the current position is equal to the content of register r. Formally, define a *register valuation* to be a partial function $f : R \to \mathbb{N}$ and consider a valid infinite computation

$$\pi = (v_1, c_1) \longrightarrow (v_2, c_2) \longrightarrow (v_3, c_3) \longrightarrow \cdots$$

of a 1-CA $\mathcal{C}$. We define a satisfaction relation $\pi, i \models_f \varphi$ specifying when an LTL$^{\downarrow}$ formula φ is satisfied at position i in π under valuation f:

$$\begin{aligned}
\pi, i \models_f a &\overset{\text{def}}{\iff} v_i = a\\
\pi, i \models_f \uparrow_r &\overset{\text{def}}{\iff} c_i = f(r)\\
\pi, i \models_f \mathsf{X}\,\varphi &\overset{\text{def}}{\iff} \pi, i+1 \models_f \varphi\\
\pi, i \models_f \varphi_1 \,\mathsf{U}\, \varphi_2 &\overset{\text{def}}{\iff} \pi, j \models_f \varphi_2 \text{ for some } j \geqslant i \text{ and } \pi, k \models_f \varphi_1 \text{ for all } i \leqslant k < j\\
\pi, i \models_f \downarrow_r \varphi &\overset{\text{def}}{\iff} \pi, i \models_{f[r \mapsto c_i]} \varphi
\end{aligned}$$

where $f[r \mapsto c]$ is the function that maps r to c and is otherwise equal to f. We have omitted the clauses for the Boolean connectives.

An occurrence of a subformula in a LTL$^{\downarrow}$ formula is *positive* if it lies within the scope of an even number of negations, otherwise it is *negative*. The *flat fragment* of LTL$^{\downarrow}$ is the set of LTL$^{\downarrow}$ formulas such that in every positive occurrence of a subformula $\varphi_1 \,\mathsf{U}\, \varphi_2$, the binding operator $\downarrow_r$ does not appear in φ_1, and in every negative occurrence of such a subformula, the binding operator does not appear in φ_2.

The negation of many natural LTL$^{\downarrow}$ specifications yield flat formulas. For example, consider the response property $\mathsf{G}(\downarrow_r (\text{req} \to \mathsf{F}(\text{serve} \wedge \uparrow_r)))$, expressing that every request is followed by a serve with the same associated ticket. Here F and G are the "future" and "globally" modalities, which can be expressed in terms of U in a standard way. The negation of this formula is equivalent to $\mathsf{F}(\downarrow_r (\text{req} \wedge \mathsf{G}(\neg\text{serve} \vee \neg\uparrow_r)))$. The latter is easily seen to be flat after rewriting to the core LTL$^{\downarrow}$ language with only the U temporal operator.

The main subject of this paper is the decidability of the following model checking problem: given a 1-CA $\mathcal{C}$, a valid configuration (v, c) of $\mathcal{C}$, and a flat sentence $\varphi \in \mathrm{LTL}^{\downarrow}$, does there exist a valid infinite computation π of $\mathcal{C}$, starting at (v, c), such that $\pi, 1 \vDash_{\emptyset} \varphi$? Note that, following [6], we have given an existential formulation of the model checking problem. The problem above is equivalent to asking whether $\neg\varphi$ holds along all infinite computations starting at (v, c).

The model checking problem for flat $\mathrm{LTL}^{\downarrow}$ on 1-CA was reduced to the generalised repeated reachability problem for 1-CA with parameterised tests in [6, Theorem 15]. The idea of the reduction is, given a 1-CA $\mathcal{C}$ and a flat $\mathrm{LTL}^{\downarrow}$ sentence φ, to construct a 1-CA with parameterised tests which is the product of $\mathcal{C}$ and φ. This product automaton includes a parameter x_r for each register r that is mentioned in a subformula of φ of type $\downarrow_r \varphi'$. This is where the restriction to the flat fragment of $\mathrm{LTL}^{\downarrow}$ is crucial, since it allows us to assume, without loss of generality, that the value stored in a register is never overwritten along any computation of $\mathcal{C}$, so that it can be represented by precisely one parameter. An occurrence of the binding operator $\downarrow_r$ in φ is represented in the product automaton by an equality test $\mathrm{eq}(x_r)$. A positive occurrence of a formula of the type $\uparrow_r$ is likewise represented by an equality test $\mathrm{eq}(x_r)$, while a negative occurrence of such a subformula is represented by a disequality test $\tau(v_r) = \{x_r\}$.

Note that the definition of 1-CA with parameterised tests in [6] includes parameterised equality and disequality tests (as in the present paper) together with parameterised inequality tests, i.e., testing whether the counter value is less than or greater than the value of a parameter. However, it is clear from the details of the reduction that only equality and disequality tests are needed, and thus we do not consider inequality tests in this paper. Note also that in the previous section we defined 1-CA to have equality tests on edges and disequality tests on states. On the other hand, the 1-CA considered in [6] have both kinds of tests on edges and allow multiple edges between the same pair of states. It is easy to see that both models are equivalent with respect to reachability, i.e., there are reductions in both directions between reachability problems in the two models.

2.3 Presburger Arithmetic

Presburger arithmetic is the first-order logic over $\langle \mathbb{Z}, +, <, 0, 1\rangle$, where $+$ and $<$ are the standard addition and ordering of integers. Presburger arithmetic is known to be decidable [12]. Using shorthand notation, we can assume that the atomic formulas of Presburger arithmetic are equalities or inequalities between linear polynomials with integer coefficients.

3 Normal Form for Paths

In this section, we show that any valid finite computation of a 1-CA can be rewritten to a normal form whose shape only depends on the automaton. Informally, any such computation can be described as a sequence of "take this edge" and "take this cycle k times". We show that the maximum length of a description of this kind is independent of the original computation.

First we show that without loss of generality, any computation can be broken down into a small number of segments that do not contain any transitions with equality tests. The idea is that any segment between two identical equality tests can be omitted.

▶ **Lemma 1** (Equality test isolation). *Let π be a valid finite computation from (v, c) to (v', c'). Then there exists a path γ such that $\gamma(c)$ is a valid computation from (v, c) to (v', c') and γ*

is of the form $\gamma = \gamma_0 e_1 \gamma_1 e_2 \cdots e_n \gamma_n$*, where* e_i *is an edge with an equality test,* γ_i *is a path without equality tests and* $n \leqslant |E|$*.*

We give a proof of Lemma 1 in Appendix A.

We need to introduce some terminology to formalise our notion of normal form. Given a state v, $\mathrm{SC}(v)$ (resp. $\mathrm{SC}^+(v)$, $\mathrm{SC}^-(v)$) denotes the *set of equality-free simple* (resp. *positive simple, negative simple*) *cycles* starting at v. The set of all equality-free simple (resp. positive simple, negative simple) cycles from all vertices is SC (resp. SC^+, SC^-). Note that each equality-free simple cycle is counted several times in SC: once for each state in the cycle.

The *cycle alphabet*, denoted C, consists of symbols of the form $\underline{\omega^k}$ where $\omega \in \mathrm{SC}$ and $k \in \mathbb{N}$. Note that this alphabet is infinite. Also note that $\underline{\omega^k}$ is a single symbol, underlined to indicate the difference from the cycle ω^k, which consists of $|\omega|k$ symbols from E. For convenience, $\underline{\omega}$ is a shorthand for $\underline{\omega^1}$. We naturally define the start and end of symbol $\underline{\omega^k}$ by the start of ω: $\mathrm{start}(\underline{\omega^k}) = \mathrm{end}(\underline{\omega^k}) = \mathrm{start}(\omega)$.

A *folded path* χ is a word on the alphabet $E \cup C$: $\chi = s_1 \cdots s_n$ such that $\mathrm{end}(s_i) = \mathrm{start}(s_{i+1})$ for every $i < n$. We also define the natural unfolding of a folded path as a monoid homomorphism $\mathrm{unfold} : (E \cup C)^* \to E^*$ such that $\mathrm{unfold}(e) = e$ for $e \in E$ and $\mathrm{unfold}(\underline{\omega^k}) = \omega^k$ for $\underline{\omega^k} \in C$. The weight of a folded path is the weight of its unfolding.

From now on, until Theorem 8 at the end of this section, we fix an initial counter value $c \in \mathbb{N}$ and we only consider computations starting at c which do not include equality tests. We refer to a folded path χ as being valid if $\mathrm{unfold}(\chi)(c)$ is a valid computation.

Define the following nondeterministic rewriting system on folded paths. Each rule of the system has a name, a pattern to match against, a condition which must be satisfied for the rule to apply and the result of the rule. We denote by $\chi \rightsquigarrow \chi'$ the fact that χ rewrites to χ'.

Rule	**Pattern**	**Result**	**Condition**
`fold`	$\psi\omega\phi$	$\psi\underline{\omega}\phi$	ω is a simple cycle of nonzero weight.
`simplify`	$\psi\rho\phi$	$\psi\phi$	Nonempty ρ, $\mathrm{weight}(\mathrm{unfold}(\rho)) = 0$ and $\mathrm{end}(\psi) = \mathrm{start}(\phi)$.
`gather`$^+$	$\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$	$\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi$	Result is valid, ω is a positive simple cycle and $\ell > 0$.
`gather`$^-$	$\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$	$\psi\underline{\omega^{k-1}}\rho\underline{\omega^{\ell+1}}\phi$	Result is valid, ω is a negative simple cycle and $k > 0$.

▶ **Lemma 2** (Soundness). *If* χ *is valid and rewrites to* χ' *then* χ' *is valid. Furthermore,* χ *and* χ' *start and end at the same state and* $\mathrm{weight}(\mathrm{unfold}(\chi)) = \mathrm{weight}(\mathrm{unfold}(\chi'))$*.*

The proof of Lemma 2 is included in Appendix A.

▶ **Lemma 3** (Termination). *There are no infinite chains of rewriting.*

A proof of Lemma 3 can be found in Appendix A. Here we give an informal explanation. The first thing to notice is that the length of a folded path (over alphabet $E \cup C$) never increases after a rewriting operation. The second thing is that the length of a folded path over E (i.e., ignoring symbols from C) never increases either. Since rule `simplify` decreases the length, it can only be applied finitely many times. Similarly, rule `fold` decreases the length over E because it replaces a symbol from E by one from C. Rules `gather`$^\pm$ are more difficult to analyse because they only reorder the path by replacing symbols from C. But as it can be seen, a symbol $\underline{\omega}$, where ω is a positive cycle, can only move left, and similarly a negative cycle can only move right. Intuitively, this process must be finite because once a positive (negative) cycle reaches the leftmost (rightmost) position, it cannot move anymore.

▶ **Lemma 4** (Size of cycle-free subpaths). *If* $\psi\rho\phi$ *is such that* $\rho \in E^*$ *and no rule applies, then* $|\rho| < |V|$*.*

Proof. Assume the contrary: if ρ only consists of edges and has length $\geqslant |V|$, then some state is repeated in the state sequence of ρ. Thus ρ contains a cycle and thus a simple cycle. So rule `fold` applies if the cycle has nonzero weight, or rule `simplify` applies if it has weight zero. ◀

For $S \subseteq \mathbb{Z}$ and $x \in \mathbb{Z}$, we use $S - x$ to denote $\{y - x \mid y \in S\}$. The idea of the next lemma is to show that given a state v, some counter values prevent reordering of cycles within the folded path. These counter values act as a "barrier" for the `gather` rules and increase the size of the normal form. We call these values *critical* for positive (resp. negative) cycles and denote them by $B^+(v)$ (resp. $B^-(v)$). Formally, $B^+(v)$ contains:

- $\tau(v) - \mathrm{weight}(\omega)$, for every positive (resp. negative) simple cycle ω,
- $\tau(\mathrm{end}(\gamma)) - \mathrm{weight}(\gamma)$ for every prefix[1] γ of every positive (resp. negative) simple cycle ω starting at v.

▶ **Lemma 5** (Obstructions in irreducible paths with cycles). *Let ω be a positive cycle and assume that rule `gather⁺` (resp. `gather⁻`) does not apply on $\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$ (which we assume is valid and $k, \ell > 0$) for this particular pattern. Then there exists a (potentially empty) prefix μ of ρ such that $\mathrm{unfold}(\psi\underline{\omega^k}\mu)(c)$ has the form $(v, c) \longrightarrow^* (v', c')$ where c' is critical for v' for positive (resp. negative) cycles,* i.e. *$c' \in B^+(v')$ (resp. $c' \in B^-(v')$). Furthermore $B^+(v')$ and $B^-(v')$ only depends on the automaton and*

$$|B^+(v')| \leqslant |\,\mathrm{SC}^+\,| \sum_{v\in V} |\tau(v)| \quad \textit{and} \quad |B^-(v')| \leqslant |\,\mathrm{SC}^-\,| \sum_{v\in V} |\tau(v)|.$$

Proof. We first show the result for positive cycles. Let $\pi = \mathrm{unfold}(\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi)(c)$ and $\pi' = \mathrm{unfold}(\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi)(c)$. To make things slightly easier to understand, note that:

$$\begin{aligned}\pi &= [\mathrm{unfold}(\psi)\omega^k\,\mathrm{unfold}(\rho)\omega\omega^{\ell-1}\,\mathrm{unfold}(\phi)](c)\\ \pi' &= [\mathrm{unfold}(\psi)\omega^k\omega\,\mathrm{unfold}(\rho)\omega^{\ell-1}\,\mathrm{unfold}(\phi)](c).\end{aligned}$$

Since $\mathrm{unfold}(\rho)\omega$ and $\omega\,\mathrm{unfold}(\rho)$ have the same weight, it is clear that the first $(\mathrm{unfold}(\psi)\omega^k)$ and last $(\omega^{\ell-1}\,\mathrm{unfold}(\phi))$ parts of the computation are the same in π and π', i.e., they have the same counter values. Consequently, if they are valid in π, the same parts are also valid in π'. Since by the hypothesis `gather⁺` does not apply, π' is invalid. So there must be an obstruction (u, d) in the middle part $(\omega\,\mathrm{unfold}(\rho))$ of π'. There are two possibilities.

The first case is when the obstruction (u, d) is in the $\mathrm{unfold}(\rho)$ part of π'. Note that $d = c^* + \mathrm{weight}(\omega)$, where (u, c^*) is the corresponding configuration in the $\mathrm{unfold}(\rho)$ part of π. Since ω is a positive cycle, $d > c^*$ cannot be negative (since (u, c^*) occurs in π, which is valid). Since we assumed that all computations are free of equality tests, the obstruction must be because of a disequality, i.e., it must be that $d = c^* + \mathrm{weight}(\omega) \in \tau(u)$. Thus $c^* \in \tau(u) - \mathrm{weight}(\omega)$ and c^* is critical for u. Then there exists a prefix μ of ρ such that $\mathrm{unfold}(\psi\underline{\omega^k}\mu)(c) = (v, c) \longrightarrow^* (u, c^*)$ and this shows the result.

The second case is when (u, d) is in the ω part of the middle part $(\omega\,\mathrm{unfold}(\rho))$ of π'. Again, it is impossible that the counter value d be negative. Indeed, remember that ω is a

[1] Other than ω and the empty prefix. Indeed the empty prefix is impossible because $c_2 \notin \tau(v_1)$ as π is valid. And ω correspond to the previous case of the definition.

positive cycle and $k > 0$, thus

$$\begin{aligned}\pi' &= [\mathrm{unfold}(\psi)\omega^{k+1}\,\mathrm{unfold}(\rho)\omega^{\ell-1}\,\mathrm{unfold}(\phi)](c)\\ &= [\mathrm{unfold}(\psi)\omega^{k-1}\omega\omega\,\mathrm{unfold}(\rho)\omega^{\ell-1}\,\mathrm{unfold}(\phi)](c)\\ &= (v,c) \xrightarrow{\mathrm{unfold}(\psi)\omega^{k-1}}{}^{*} (v_1,c_1) \xrightarrow{\omega}{}^{*} (v_1,c_2) \xrightarrow{\omega}{}^{*} (v_1,c_3) \xrightarrow{\mathrm{unfold}(\rho)\omega^{\ell-1}\,\mathrm{unfold}(\phi)}{}^{*} (v'',c'').\end{aligned}$$

We already argued that $(v,c) \longrightarrow^{*} (v_1,c_2)$ is valid, so in particular $(v_1,c_1) \xrightarrow{\omega}{}^{*} (v_1,c_2)$ is valid. Note that the obstruction is in the second iteration of ω: $(v_1,c_2) \xrightarrow{\omega}{}^{*} (v_1,c_3)$. Since ω is a positive cycle, $c_2 > c_1$. Note that initially the cycle ω was feasible (with the counter not going negative) starting with a lower counter value (c_1) so the counter cannot possibly become negative on the second iteration starting with a higher counter value (c_2). Thus, again, the obstruction happens because of a disequality. That is, we can write $\omega = \gamma\gamma'$ such that:

$$\pi' = (v,c) \xrightarrow{\mathrm{unfold}(\psi)\omega^{k}}{}^{*} (v_1,c_2) \xrightarrow{\gamma}{}^{*} (u,d) \xrightarrow{\gamma'}{}^{*} (v_1,c_3) \xrightarrow{\mathrm{unfold}(\rho)\omega^{\ell-1}\,\mathrm{unfold}(\phi)}{}^{*} (v'',c'')$$

and the obstruction happens because $d \in \tau(u)$. Note however that $d = c_2 + \mathrm{weight}(\gamma)$ and thus $c_2 \in \tau(u) - \mathrm{weight}(\gamma)$. In this case, c_2 is critical for v_1. Choose μ to be the empty word, so that $\mathrm{unfold}(\psi\underline{\omega^{k}}\mu)(c) = (v,c) \longrightarrow^{*} (v_1,c_2)$ to show the result.

Observe that the definition of critical values only depend on the automaton itself. Furthermore, the size of $B^{+}(v)$ can easily be bounded. Indeed, there are $|\,\mathrm{SC}^{+}\,|$ positive simple cycles, so in the first case of the definition, there are at most $|\,\mathrm{SC}^{+}\,||\tau(v')|$ values. In the second case, since the cycle ω is simple, each prefix γ of ω ends at a different state. Thus each state is visited at most once, and v is not visited because the prefix is not empty or equal to ω. So the second case includes at most an additional $|\,\mathrm{SC}^{+}\,|\sum_{u\neq v}|\tau(u)|$ values. Finally the total bound is $|\,\mathrm{SC}^{+}\,|\sum_{v\in V}|\tau(v)|$.

The proof is exactly the same in the negative case except for one detail. This time we move negative cycles to the right so that the middle part of π' ($\mathrm{unfold}(\rho)\omega$) can only get higher counter values than the middle part of π ($\omega\,\mathrm{unfold}(\rho)$), as in the positive case. ◀

▶ **Lemma 6** (Length of irreducible paths). *Let χ be a folded path such that no rule applies on χ. Let $Y = \mathrm{SC}^{+}$ or $Y = \mathrm{SC}^{-}$. Then for every $\omega \in Y$, the number of symbols in χ of the form $\underline{\omega^{\cdot}}$ (the exponent does not matter) is bounded by*

$$|V||Y|\Big(1 + \sum_{v\in V}|\tau(v)|\Big).$$

Proof. Without loss of generality, we show the result for $X = \mathrm{SC}^{+}$. First note that if $\underline{\omega^{k}}$ appears in χ and no rule applies, then $k > 0$, otherwise we could apply `simplify` to remove $\underline{\omega^{0}}$. We can thus decompose the path as:

$$\chi = \phi_0\underline{\omega^{k_1}}\phi_1\underline{\omega^{k_2}}\phi_2\cdots\phi_{n-1}\underline{\omega^{k_n}}\phi_n$$

where $k_i > 0$ and ϕ_i does not contain any $\underline{\omega^{\cdot}}$ symbol. Since no rule applies, by Lemma 5, there exist prefixes $\mu_1, \mu_2, \ldots, \mu_{n-1}$ of $\phi_1, \phi_2, \ldots, \phi_{n-1}$ respectively, such that for each i:

$$(v,c) \xrightarrow{\phi_0\underline{\omega^{k_1}}\phi_1\cdots\phi_{i-1}\underline{\omega^{k_i}}\mu_i}{}^{*} (v_i,c_i) \quad \text{where} \quad c_i \in B^{+}(v_i).$$

Assume for a contradiction that there is a repeated configuration among the (v_i, c_i). Then there exists $i < j$ such that $v_i = v_j$ and $c_i = c_j$. Let $\phi_i = \mu_i\rho$ and $\phi_j = \mu_j\rho'$, and observe that:

$$(v, c) \xrightarrow{\phi_0\underline{\omega}^{k_1}\phi_1\cdots\phi_{i-1}\underline{\omega}^{k_i}\mu_i}{}^* (v_i, c_i) \xrightarrow{\rho\underline{\omega}^{k_{i+1}}\phi_{i+1}\cdots\phi_{j-1}\underline{\omega}^{k_j}\mu_j}{}^* (v_i, c_i) \xrightarrow{\rho'\underline{\omega}^{k_{j+1}}\phi_{j+1}\cdots\phi_{n-1}\underline{\omega}^{k_n}\phi_n}{}^* (v', c').$$

Thus the subpath $\rho\underline{\omega}^{k_{i+1}}\phi_{i+1}\cdots\phi_{j-1}\underline{\omega}^{k_j}\mu_j$ has weight 0 and rule `simplify` must apply:

$$\chi \quad \rightsquigarrow \quad \phi_0\underline{\omega}^{k_1}\phi_1\cdots\phi_{i-1}\underline{\omega}^{k_i}\mu_i\rho'\underline{\omega}^{k_{j+1}}\phi_{j+1}\cdots\phi_{n-1}\underline{\omega}^{k_n}\phi_n$$

which is a contradiction because we assumed that no rule can apply on χ.

Consequently, for any $i \neq j$, we have $(v_i, c_i) \neq (v_j, c_j)$. But remember that $c_i \in B^+(v_i)$, thus $(v_i, c_i) \in A$ where:

$$A = \bigcup_{v\in V} \{v\} \times B^+(v).$$

This shows that $n - 1 \leqslant |A|$. Indeed, by the pigeonhole principle, some pair (v_i, c_i) would be repeated if $n - 1 > |A|$. We can easily bound the size of A using the bound on $B^+(v)$ from Lemma 5:

$$|A| \leqslant \sum_{v\in V} |B^+(v)| \leqslant |V| |\operatorname{SC}^+| \sum_{v\in V} |\tau(v)|.$$

Finally we have

$$n \leqslant |V| |\operatorname{SC}^+| \sum_{v\in V} |\tau(v)| + 1 \leqslant |V| |\operatorname{SC}^+| \Big(1 + \sum_{v\in V} |\tau(v)|\Big)$$

because $|V| \geqslant 1$ and $|\operatorname{SC}^+| \geqslant 1$ unless there are no positive cycles, in which case $n = 0$ anyway. ◀

▶ **Lemma 7** (Length of equality-free computations). *Let π be a valid finite computation (without equality tests) from (v, c) to (v', c'). Then there exists a folded path χ such that* $\operatorname{unfold}(\chi(c))$ *is a valid computation from (v, c) to (v', c'), the length of* $\operatorname{unfold}(\chi(c))$ *at most that of π and the word length of χ is bounded by:*

$$|V| + |V|^2 |\operatorname{SC}|^2 \Big(1 + \sum_{v\in V} |\tau(v)|\Big)$$

Proof. Let χ_0 be the path defined by π: it is a word over alphabet E and is thus a (trivial) folded path. By definition $\operatorname{unfold}(\chi_0(c)) = \pi$ is a valid computation from (v, c) to (v', c') and the length of $\operatorname{unfold}(\chi_0(c))$ is equal to that of π. Let χ be any rewriting of χ_0 such that no rule applies on χ: it exists because there are no infinite rewriting chains by Lemma 3. By Lemma 2, $\operatorname{unfold}(\chi(c))$ is still a valid computation from (v, c) to (v', c'). Let ω be a simple cycle: note that it is either positive or negative, because rule `simplify` removes zero-weight cycles. Then by Lemma 6, the number of symbols of the form $\underline{\omega}^{\cdot}$ appearing in χ is bounded by[2]:

$$|V| |\operatorname{SC}| \Big(1 + \sum_{v\in V} |\tau(v)|\Big) \tag{2}$$

[2] Since obviously $\max(|\operatorname{SC}^+|, |\operatorname{SC}^-|) \leqslant |\operatorname{SC}|$.

and thus the total number of symbols in χ of the form $\underline{\omega^{\cdot}}$ for any ω is bounded by:

$$|V||\,\mathrm{SC}\,|^2\Big(1+\sum_{v\in V}|\tau(v)|\Big). \tag{3}$$

Furthermore, inbetween symbols of the form $\underline{\omega^{\cdot}}$, there can be subpaths consisting of symbols in E only, so χ is of the form

$$\chi = \phi_0\underline{\omega_1^{k_1}}\phi_1\underline{\omega_2^{k_2}}\cdots\underline{\omega_n^{k_n}}\phi_n$$

where $\phi_i \in E^*$ and $\omega_i \in \mathrm{SC}$ for all i. By the reasoning above, $n \leqslant (3)$. Furthermore, by Lemma 4, $\phi_i < |V|$ for all i. It follows that the total length of χ is bounded by

$$\begin{aligned}(n+1)(|V|-1)+n &\leqslant |V|+n|V|\\ &\leqslant |V|+|V|^2|\,\mathrm{SC}\,|^2\Big(1+\sum_{v\in V}|\tau(v)|\Big).\end{aligned}$$

Finally the length of $\mathrm{unfold}(\chi(c))$ at most that of π because the rewriting system does not increase the length of the path and the length of $\mathrm{unfold}(\chi_0(c))$ is equal to that of π. ◀

The main result of this section shows that any valid computation has an equivalent valid computation given by a folded path whose length only depends on the automaton.

▶ **Theorem 8** (Length of computations). *Let π be a valid finite computation from (v,c) to (v',c'). Then there exists a folded path χ such that $\chi(c)$ is a valid computation from (v,c) to (v',c'), the length of* $\mathrm{unfold}(\chi(c))$ *is at most that of π and the word length of χ is bounded by:*

$$|E|\left(1+|V|+|V|^2|\,\mathrm{SC}\,|^2\Big(1+\sum_{v\in V}|\tau(v)|\Big)\right).$$

Proof. Apply Lemma 1 to isolate the equality tests (at most $|E|$ of them) and apply Lemma 7 to each equality-free subcomputation. We can improve the bound slightly by noticing that there can only be up to $|E|$ equality-free subcomputations (and not $|E|+1$). Indeed, if there are $|E|$ different equality tests in the path, there are no further edges available for equality-free computations, and the word length is at most $|E|$. ◀

4 Reachability with Parameterised Tests

In this section we will show that both the reachability problem and the generalised repeated reachability problem for 1-CA with parameterised tests are decidable, via a symbolic encoding of folded paths, making use of the normal form from the previous section. The result of this encoding is a formula of Presburger arithmetic.

Recall that $C = \{\underline{\omega^k} : \omega \in \mathrm{SC}, k \in \mathbb{N}\}$. Let $C' = \{\underline{\omega^{\cdot}} : \omega \in \mathrm{SC}\}$. We define a *path shape* to be a word over the alphabet $E \cup C'$: $\xi = t_1 \dots t_n$ such that $\mathrm{end}(t_i) = \mathrm{start}(t_{i+1})$, where $\mathrm{start}(\underline{\omega^{\cdot}}) = \mathrm{end}(\underline{\omega^{\cdot}}) = \mathrm{start}(\omega)$. Given a path shape $\xi = \gamma_0\underline{\omega_1^{\cdot}}\gamma_1\dots\underline{\omega_n^{\cdot}}\gamma_n$ with $\gamma_i \in E^*$, we write $\xi(k_1,\dots,k_n)$ for the folded path $\gamma_0\underline{\omega_1^{k_1}}\gamma_1\dots\underline{\omega_n^{k_n}}\gamma_n$. The advantage of working with path shapes rather than folded paths is that the former are words over a finite alphabet.

▶ **Lemma 9** (Encoding computations). *Given a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with parameterised tests and configurations (v,c) and (v',c'), and given a path shape $\xi = t_1t_2\dots t_n \in (E \cup C')^*$, there exists a Presburger arithmetic formula $\varphi_{comp}^{(\xi),(v,c),(v',c')}(\boldsymbol{k},\boldsymbol{x})$, with free variables $\boldsymbol{x}$ corresponding to the parameters X and $\boldsymbol{k}$ corresponding to exponents to be substituted in ξ, which evaluates to true if and only if* $\mathrm{unfold}(\xi(\boldsymbol{k}))(c)$ *is a valid computation from (v,c) to (v',c').*

Proof. Assume first that ξ does not include any equality tests. We define a formula $\varphi^{(t)}_{valid,noeq}(\boldsymbol{k},\boldsymbol{x},y)$ which, given an equality-free symbol $t\in E\cup C'$ and an integer y, evaluates to true if and only if $\mathrm{unfold}(t(\boldsymbol{k}))(y)$ is a valid computation. There are two cases:

- $t\in E$. Then $\varphi^{(t)}_{valid,noeq}(\boldsymbol{x},y)\equiv y\geqslant 0\wedge y+\mathrm{weight}(t)\geqslant 0\wedge y\notin\tau(\mathrm{start}(t))$.
- $t\in C'$, i.e., $t(\boldsymbol{k})=\underline{\omega^k}$ for some simple cycle $\omega=e_1e_2\dots e_\ell$ and $k\in\boldsymbol{k}$. Then

$$\varphi^{(t)}_{valid,noeq}(\boldsymbol{k},\boldsymbol{x},y)\equiv\forall k'\,(0\leqslant k'<k)\Rightarrow\bigwedge_{i=1}^{\ell}\left(y+k'\,\mathrm{weight}(\omega)+\sum_{j=1}^{i-1}\mathrm{weight}(e_j)\geqslant 0\wedge\right.$$
$$\left.y+k'\,\mathrm{weight}(\omega)+\sum_{j=1}^{i-1}\mathrm{weight}(e_j)\notin\tau(\mathrm{start}(e_i))\right)\wedge y+k\,\mathrm{weight}(\omega)\geqslant 0.$$

Note that for each edge $e\in E$, $\mathrm{weight}(e)$ is a constant, given by the automaton, and $\mathrm{weight}(\omega)$ is a shorthand for $\sum_{i=1}^{\ell}\mathrm{weight}(e_i)$, which is also a constant. So the only type of multiplication in the formula is by a constant. A formula of the form $a\notin\tau(u)$ is a shorthand for $\bigwedge_{b\in\tau(u)}a\neq b$, which is clearly a Presburger arithmetic formula. Since $\mathcal{C}$ has parameterised tests, in general some of these disequalities include variables from $\boldsymbol{x}$. We can now define a formula with the required property in the case where ξ does not include any equality tests:

$$\varphi^{(\xi),(v,c),(v',c')}_{comp,noeq}(\boldsymbol{k},\boldsymbol{x})\equiv\left(\bigwedge_{i=1}^{n-1}\mathrm{end}(t_i)=\mathrm{start}(t_{i+1})\right)\wedge\mathrm{start}(t_1)=v\wedge\mathrm{end}(t_n)=v'\wedge$$
$$\sum_{i=1}^{n}\mathrm{weight}(t_i(\boldsymbol{k}))=c'-c\wedge\bigwedge_{i=1}^{n}\varphi^{(t_i)}_{valid,noeq}(\boldsymbol{k},\boldsymbol{x},c+\sum_{j=1}^{i-1}\mathrm{weight}(t_j(\boldsymbol{k}))),$$

where we use the shorthand $\mathrm{weight}(s)$ for $s\in E\cup C$: if $s\in E$ then $\mathrm{weight}(s)$ is a constant as above, and if $s\in C$ then it is of the form $\underline{\omega^k}$ and $\mathrm{weight}(s)=k\sum_{e\in\omega}\mathrm{weight}(e)$. Again, the only multiplications are by constants, so the resulting formula is a formula of Presburger arithmetic.

Finally, in the case where ξ includes equality tests, we split $\mathrm{unfold}(\xi)$ at the t_i which are equality tests, and construct a formula $\varphi_{comp,noeq}$ as above for each equality-free part of ξ. $\varphi^{(\xi),(v,c),(v',c')}_{comp}(\boldsymbol{k},\boldsymbol{x})$ is the conjunction of these formulas. ◀

▶ Remark 10 (Removing the universal quantification). For simplicity, we have used a universal quantifier in $\varphi^{(t)}_{valid,noeq}(\boldsymbol{k},\boldsymbol{x},y)$ to express that k iterations of a cycle yield a valid computation. In fact it is possible to rewrite $\varphi^{(t)}_{valid,noeq}(\boldsymbol{k},\boldsymbol{x},y)$ as a purely existential formula, with a polynomial blowup. Let $\omega=e_1\cdots e_\ell$ be a cycle and suppose we want to check that $\omega^k(y)$ is a valid computation. Let $u=\mathrm{start}(e_i)$ be a state on the cycle. First we need to express that the counter value at u is never negative along $\omega^k(y)$. Since the counter value at u is monotone during the k iterations of the cycle (it increases if ω is positive and decreases if ω is negative), we only need check that it is nonnegative at the first and last iteration:

$$y+\sum_{j=1}^{i-1}\mathrm{weight}(e_j)\geqslant 0\wedge y+(k-1)\,\mathrm{weight}(\omega)+\sum_{j=1}^{i-1}\mathrm{weight}(e_j)\geqslant 0.$$

Next, for each $b\in\tau(u)$, we need to check that the cycle avoids b in u. Without loss of generality, assume that ω is positive. Then the counter value at u increases after each iteration. We can now perform a case analysis on the three ways to satisfy a disequality test during the k iterations of ω:

- The value at the first iteration is already bigger than b:

$$y + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) > b.$$

- The value at the last iteration is less than b:

$$y + (k-1)\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) < b.$$

- There is an iteration k', with $0 \leqslant k' < k-1$, at which the counter value is less than b, but where at the next iteration $k'+1$ the counter value is bigger than b:

$$\exists k'\,(0 \leqslant k' < k-1) \wedge y + k'\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) < b$$
$$\wedge\, y + (k'+1)\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) > b.$$

Finally, we can use a conjunction over all vertices in ω to get a formula which is equivalent to $\varphi^{(t)}_{valid,noeq}(\boldsymbol{k}, \boldsymbol{x}, y)$ but has no universal quantifiers.

▶ **Lemma 11** (Encoding reachability). *Let $\mathcal{C} = (V, E, X, \lambda, \tau)$ be a 1-CA with parameterised tests, and let (v, c) and (v', c') be given configurations of $\mathcal{C}$. Then there exists a Presburger arithmetic formula $\varphi^{(v,c),(v',c')}_{reach}(\boldsymbol{x})$ which evaluates to true if and only if there is a valid computation from (v, c) to (v', c') in $\mathcal{C}$, as well as a formula $\varphi^{(v,c),(v',c')}_{reach_+}$ which is true if and only if there is such a computation of length at least* 1.

Proof. Note that the bounds on the length of computations in 1-CA from the previous section do not depend on the values occurring in equality or disequality tests. That is, if there is a valid computation $(v, c) \xrightarrow{\pi}{}^{*} (v', c')$ for any given values of the parameters, then there is a folded path χ of word length at most $p(\mathcal{C})$ such that $(v, c) \xrightarrow{\mathrm{unfold}(\chi(c))}{}^{*} (v', c')$ is a valid computation, where p is the polynomial function given in Theorem 8. Equivalently, there is a path shape ξ of word length at most $p(\mathcal{C})$ and there exist values $\boldsymbol{k}$ such that $(v, c) \xrightarrow{\mathrm{unfold}(\xi(\boldsymbol{k})(c))}{}^{*} (v', c')$ is a valid computation.

Since path shapes are words over a finite alphabet, we can express this property as a finite disjunction

$$\varphi^{(v,c),(v',c')}_{reach}(\boldsymbol{x}) \equiv \exists \boldsymbol{k} \bigvee_{|\xi| \leqslant p(\mathcal{C})} \varphi^{(\xi),(v,c),(v',c')}_{comp}(\boldsymbol{k}, \boldsymbol{x}).$$

For φ_{reach_+}, we simply change the disjunction to be over all ξ such that $1 \leqslant |\xi| \leqslant p(\mathcal{C})$. ◀

▶ **Lemma 12** (Encoding repeated reachability). *Let $\mathcal{C} = (V, E, X, \lambda, \tau)$ be a 1-CA with parameterised tests, let $F \subseteq V$ be a set of final states, and let (v, c) be the initial configuration of $\mathcal{C}$. Then there exists a Presburger arithmetic formula $\varphi^{(v,c),(F)}_{rep\text{-}reach}(\boldsymbol{x})$ which evaluates to true if and only if there is a valid infinite computation π which starts in (v, c) and visits at least one state in F infinitely often.*

Proof. Suppose there is an infinite computation which starts in (v, c) and visits a state $u \in F$ infinitely often. Equivalently, there is a counter value $d \in \mathbb{N}$ such that $(v, c) \longrightarrow^* (u, d)$ is a valid (finite) computation, and there is a cycle ω with $\mathrm{start}(\omega) = u$ such that $\omega^k(d)$ is a valid computation for all $k \in \mathbb{N}$. There are two possible cases:

- $\mathrm{weight}(\omega) = 0$, so $\omega^k(d)$ is valid for all k if and only if $\omega(d)$ is valid.
- $\mathrm{weight}(\omega) > 0$, so it might be possible to start from (u, d) and follow the edges of ω a finite number of times before an obstruction occurs. However, if ω can be taken an arbitrary number of times, then the counter value will tend towards infinity, so we are free to choose ω to be an equality-free simple cycle, and d to be high enough to guarantee that if ω can be taken once without obstructions, it can be taken infinitely many times.

The resulting formula is then

$$\varphi^{(v,c),(F)}_{\textit{rep-reach}}(\boldsymbol{x}) \equiv \exists d \bigvee_{u \in F} \Bigg(\varphi^{(v,c),(u,d)}_{\textit{reach}}(\boldsymbol{x}) \wedge \Big(\varphi^{(u,d),(u,d)}_{\textit{reach}_+}(\boldsymbol{x}) \vee$$

$$(d > M(\boldsymbol{x}) \wedge \exists d' \bigvee_{\omega \in \mathrm{SC}^+} \varphi^{(\underline{\omega^{\cdot}}),(u,d),(u,d')}_{\textit{comp,noeq}}(1, \boldsymbol{x}))\ \Big) \Bigg)$$

where $M(\boldsymbol{x}) = \max\left(\bigcup_{v \in V} \tau(v)\right) - \sum\{\mathrm{weight}(e) : e \in E, \mathrm{weight}(e) < 0\}$. The sum over negative edge weights ensures that the counter always stays above $\max\left(\bigcup_{v \in V} \tau(v)\right)$ along the computation $\omega(d)$, since each edge is taken at most once in ω. Since ω is a positive cycle, this implies that the counter always stays above all bad values along $\omega(d^k)$ for each $k \in \mathbb{N}$, so no obstructions can occur. ◀

▶ **Theorem 13** (Decidability of reachability problems). *Both the reachability problem and the generalised repeated reachability problem are decidable for 1-CA with parameterised tests.*

Proof. Given a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with parameterised tests and configurations (v, c) and (v', c'), to check if there exist values for the parameters X such that there is a valid computation from (v, c) to (v', c'), we use Lemma 11 to construct the formula $\exists \boldsymbol{x}\, \varphi^{(v,c),(v',c')}_{\textit{reach}}(\boldsymbol{x})$.

To solve the generalised repeated reachability problem for a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with sets of final states $F_1, \ldots, F_n \subseteq V$ and initial configuration (v, c), note that this problem can easily be reduced to the simpler case where $n = 1$, using a translation similar to the standard translation from generalised Büchi automata to Büchi automata. In the case where $n = 1$, we can use Lemma 12 to construct the formula $\exists \boldsymbol{x}\, \varphi^{(v,c),(F_1)}_{\textit{rep-reach}}(\boldsymbol{x})$. ◀

▶ **Corollary 14** (Decidability of model checking flat Freeze LTL). *The existential model checking problem for flat Freeze LTL on 1-CA is decidable.*

5 Conclusion

The main result of this paper is that the model checking problem for the flat fragment of Freeze LTL on one-counter automata is decidable. We have concentrated on showing decidability rather than achieving optimal complexity. For example, we have reduced the model checking problem to the decision problem for the class of sentences of Presburger arithmetic with quantifier prefix $\exists^*\forall^*$. We explained in Remark 10 that in fact the reduction can be refined to yield a (polynomially larger) purely existential sentence.

Another important determinant of the complexity of our procedure is the dependence of the symbolic encoding of computations (via path shapes) in Section 4 on the number of simple cycles in the underlying control graph of the one-counter automaton. The number of such cycles may be exponential in the number of vertices. It remains to be seen whether it is possible to give a more compact symbolic representation, e.g., in terms of the Parikh image of paths. As it stands, our procedure works as follows. From the flat LTL$^{\downarrow}$ formula, we build a 1-CA with parameterised tests (of exponential size). We then guess the normal form of the path shapes (of exponential size in the size the automaton). We finally check the resulting existential Presburger formula. Since the Presburger formula has size double exponential in the size of the original LTL$^{\downarrow}$ formula, we get a naive upper bound of 2NEXPTIME for our algorithm. Improving this bound is a subject of ongoing work.

Another interesting complexity question concerns configuration reachability in one-counter automata with non-parameterised equality and disequality tests. For automata with only equality tests and with counter updates in binary, reachability is known to be NP-complete [9]. If inequality tests are allowed then reachability is PSPACE-complete [7]. Now automata with equality and disequality tests are intermediate in expressiveness between these two models and the complexity of reachability in this case is open as far as we know.

References

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

2 H. Comon and V. Cortier. Flatness is not a weakness. In *Proceedings of CSL*, volume 1862 of *LNCS*. Springer, 2000.

3 S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *Proceedings of LICS*, pages 17–26. IEEE Computer Society, 2006.

4 S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. In *Proceedings of TIME*, pages 113–121, 2005.

5 S. Demri, R. Lazić, and A. Sangnier. Model checking freeze LTL over one-counter automata. In *Proceedings of FOSSACS*, volume 4962 of *LNCS*, pages 490–504, 2008.

6 S. Demri and A. Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In *Proceedings of FOSSACS*, volume 6014 of *LNCS*, pages 176–190, 2010.

7 John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015.

8 T. French. Quantified propositional temporal logic with repeating states. In *Proceedings of TIME-ICTL*, pages 155–165. IEEE Computer Society, 2003.

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

10 O. H. Ibarra, T. Jiang, N. Tran, and H. Wang. New decidability results concerning two-way counter machines and applications. In *Proceedings of ICALP*, volume 700 of *LNCS*. Springer, 1993.

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

12 M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I congrés de Mathématiciens des Pays Slaves. Warsaw*, pages 92–101, 1929.

A Appendix

▶ Lemma 1 (Equality test isolation). Let π be a valid finite computation from (v,c) to (v',c'). Then there exists a path γ such that $\gamma(c)$ is a valid computation from (v,c) to (v',c') and γ is of the form $\gamma = \gamma_0 e_1 \gamma_1 e_2 \cdots e_n \gamma_n$, where e_i is an edge with an equality test, γ_i is a path without equality tests and $n \leqslant |E|$.

Proof. Let π' be the shortest valid computation from (v,c) to (v',c'). We can decompose it as

$$\pi' = (v,c) \xrightarrow{\gamma_0} (v_1,c_1) \xrightarrow{e_1} (v'_1,c_1) \xrightarrow{\gamma_1} (v_2,c_2) \cdots (v'_n,c_n) \xrightarrow{\gamma_n} (v',c')$$

where for every i, γ_i is a path without any equality tests and $e_i = (v_i, eq(c_i), v'_i) \in E$ is an equality test. Then clearly $\pi = \gamma(c)$ where

$$\gamma = \gamma_0 e_1 \gamma_1 e_2 \cdots e_n \gamma_n.$$

Assume for a contradiction that $n > |E|$. Then by the pigeonhole principle, there exists $i < j$ such that $e_i = e_j$. But since π' is a valid computation, the two transitions $(v_i,c_i) \xrightarrow{eq(e_i)} (v'_i,c_i)$ $(v_j,c_j) \xrightarrow{eq(c_i)} (v'_j,c_j)$ are the same and $(v_i,c_i) = (v_j,c_j)$. Thus we can delete part of the computation and define

$$\gamma' = \gamma_0 e_1 \gamma_1 \cdots e_i \gamma_j e_{j+1} \cdots e_n \gamma_n.$$

Then $\gamma'(c)$ is a valid computation from (v,c) to (v',c') and is shorter than π', which is a contradiction. ◀

▶ Lemma 2 (Soundness). If χ is valid and rewrites to χ' then χ' is valid. Furthermore, χ and χ' start and end at the same vertex and $\mathrm{weight}(\mathrm{unfold}(\chi)) = \mathrm{weight}(\mathrm{unfold}(\chi'))$.

Proof. This is easily checked for each rule:

- `fold`: Clearly $\mathrm{unfold}(\chi) = \mathrm{unfold}(\chi')$.
- `simplify`: First note that the result is well-formed because of the condition on start and end. The unfolding of the first part (ψ) of the path is unchanged, so it remains valid and with the same starting vertex. Since the second part of the path (ρ) has weight 0, the counter value is the same at the beginning and end of ρ, so the unfolding of the third part (ϕ) stays the same, and thus valid with the same end vertex. The weight of the unfolded path remains unchanged as the removed part ρ has weight 0.
- `gather`$^{\pm}$: The condition ensures the result is valid. The start and end vertex clearly do not change, and neither does the weight, since $\mathrm{unfold}(\chi')$ contains the same edges as $\mathrm{unfold}(\chi)$, only in a different order.

◀

▶ Lemma 3 (Termination). There are no infinite chains of rewriting.

Proof. We will define a valuation over folded paths and show that it decreases after each application of a rule. First, for any folded path χ and any given simple cycle ω, define the *ω-projection* $p_\omega(\chi)$ of χ as the subword only consisting of symbols of the form $\underline{\omega^k}$:

$$p_\omega(e\chi) = p_\omega(\chi) \text{ if } e \in E \qquad p_\omega(\underline{\omega^k}\chi) = \underline{\omega^k} p_\omega(\chi) \qquad p_\omega(\underline{\theta^k}\chi) = p_\omega(\chi) \text{ if } \theta \neq \omega.$$

For any folded path χ, define:

$$(\!|\chi|\!) = (|\chi|, |\chi|_E, \sigma(\chi)) \qquad \text{where} \qquad \sigma(\chi) = \sum_{\omega \in \mathrm{SC}} \sigma_\omega(p_\omega(\chi))$$

where $|\chi|$ is the word length of χ (over alphabet $E \cup C$), $|\chi|_E$ is the word length of χ only counting symbols in E and $\sigma_\omega(p_\omega(\chi))$ is defined as follows:

$$\sigma_\omega\left(\underline{\omega^{k_1}}\,\underline{\omega^{k_2}}\cdots\underline{\omega^{k_n}}\right) = \begin{cases} \sum_{i=1}^{n} i k_i & \text{if } \operatorname{weight}(\omega) > 0 \\ 0 & \text{if } \operatorname{weight}(\omega) = 0 \\ \sum_{i=1}^{n} (n+1-i) k_i & \text{if } \operatorname{weight}(\omega) < 0. \end{cases}$$

We will now show that $(\!|\chi|\!)$ decreases in lexicographic order each time a rule is applied. In the case of rule `fold`, if $|\omega| > 2$ then clearly $|\chi|$ decreases because we replace several symbols with just one. If $|\omega| = 1$ then $|\chi|$ stays constant but $|\chi|_E$ decreases by one because we replace one symbol from E by one symbol from C. Similarly, rule `simplify` decreases $|\chi|$ because we remove a nonzero-length subpath. Since rules `gather+` and `gather-` are symmetric, we only prove it for `gather+`. Note that the rule does not change $|\chi|$ or $|\chi|_E$ because it only replaces symbols from C with different symbols from C, so we are only concerned with $\sigma(\chi)$.

Assume the rule rewrites $\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$ into $\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi$. First note that if $\theta \neq \omega$ is a simple cycle, then the θ-projection is the same before and after the rule because the rule does not replace any symbols of the form $\underline{\theta^k}$, so σ_θ does not change. The case of σ_ω is slightly more involved and we need to introduce some notations:

$$p_\omega(\psi) = \underline{\omega^{u_1}}\cdots\underline{\omega^{u_n}}, \qquad p_\omega(\rho) = \underline{\omega^{u_{n+2}}}\cdots\underline{\omega^{u_m}}, \qquad p_\omega(\phi) = \underline{\omega^{u_{m+2}}}\cdots\underline{\omega^{u_q}}$$

and

$$u_{n+1} = k, \quad u_{m+1} = \ell, \quad u'_{n+1} = k+1, \quad u'_{m+1} = \ell - 1, \quad u'_i = u_i \text{ if } i \neq n+1, m+1.$$

Then we can observe that:

$$\sigma_\omega\left(p_\omega\left(\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi\right)\right) = \sigma_\omega\left(\underline{\omega^{u_1}}\cdots\underline{\omega^{u_q}}\right) = \sum_{i=1}^{q} i u_i, \tag{4}$$

$$\sigma_\omega\left(p_\omega\left(\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi\right)\right) = \sigma_\omega\left(\underline{\omega^{u'_1}}\cdots\underline{\omega^{u'_q}}\right) = \sum_{i=1}^{q} i u'_i. \tag{5}$$

Thus:

$$\begin{aligned} (4) - (5) &= \sum_{i=1}^{q} i(u_i - u'_i) \\ &= (n+1)(u_{n+1} - u'_{n+1}) + (m+1)(u_{m+1} - u'_{m+1}) \\ &= -(n+1) + (m+1) \\ &> 0 \text{ because } m > n. \end{aligned}$$

Thus $\sigma_\omega(\chi)$ decreases after the rule is applied and thus $\sigma(\chi)$ also decreases. ◀