Invariants for One-Counter Automata with Disequality Tests

Dmitry Chistikov[1]
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Jérôme Leroux
LaBRI, CNRS, Univ. Bordeaux, France

Henry Sinclair-Banks
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Nicolas Waldburger
IRISA, Université de Rennes, France

Abstract

We study the reachability problem for one-counter automata in which transitions can carry disequality tests. A disequality test is a guard that prohibits a specified counter value. This reachability problem has been known to be NP-hard and in PSPACE, and characterising its computational complexity has been left as a challenging open question by Almagor, Cohen, Pérez, Shirmohammadi, and Worrell [1]. We reduce the complexity gap, placing the problem into the second level of the polynomial hierarchy, namely into the class $\mathsf{coNP}^{\mathsf{NP}}$. In the presence of both equality and disequality tests, our upper bound is at the third level, $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$.

To prove this result, we show that non-reachability can be witnessed by a pair of invariants (forward and backward). These invariants are almost inductive. They aim to over-approximate only a "core" of the reachability set instead of the entire set. The invariants are also leaky: it is possible to escape the set. We complement this with separate checks as the leaks can only occur in a controlled way.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Inductive invariant, Vector addition system, One-counter automaton

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2024.17

Related Version *Full Version*: `https://arxiv.org/abs/2408.11908`

Funding *Dmitry Chistikov*: Supported in part by the Engineering and Physical Sciences Research Council [EP/X03027X/1].
Henry Sinclair-Banks: Supported by EPSRC Standard Research Studentship (DTP), grant number EP/T51794X/1. Also supported in part by the International Emerging Actions grant (IEA'22) and by the ANR grant VeSyAM (ANR-22-CE48-0005).
Nicolas Waldburger: Supported in part by the International Emerging Actions grant (IEA'22) and by the ANR grant VeSyAM (ANR-22-CE48-0005).

Acknowledgements We would like to thank Mahsa Shirmohammadi, without whom this work would not have been possible, for many ideas and encouragement. We are also grateful to Thomas Colcombet, Rayna Dimitrova, and James Worrell for useful discussions.

[1] During the work on this paper, DC was a visitor to the Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern and Saarbrücken, Germany, a visiting fellow at St Catherine's College and a visitor to the Department of Computer Science at the University of Oxford, United Kingdom.

1 Introduction

It is well known that the computational complexity of problems is often sensitive to seemingly minor adjustments in the problem setting. Consider, for example, vector addition systems with states (VASS). Perhaps more commonly presented as Petri nets, VASS are a very simple yet powerful model of concurrency. Many important computational problems in logic, language theory, and formal verification reduce to or are even equivalent to the reachability problem in VASS (see, e.g., [29, 45]). However, a classical result due to Minsky shows that adding the capability to test counters for zero makes the problem undecidable [42].

At the same time, while reachability in VASS is known to be decidable [41], its computational complexity was recently shown to be extremely high, namely Ackermann-complete (see [39] for the upper bound and [19, 20, 38] for the lower bound), so from the practical point of view one might question the significance of the complexity jump arising from zero tests.

More recent, "down-to-earth", and perhaps more striking is the following result on 1-dimensional VASS, which can be thought of as finite-state automata equipped with one counter (capable of storing a nonnegative integer). Reachability in these systems can be decided in NP [28] and is in fact NP-complete. It is not difficult to show, using the standard hill-cutting technique [48], that reachability can also always be witnessed by executions in which all values assumed by the counter are bounded from above by an exponential function in the size of the system and the bit length of counter values of the source and target configurations. Because of this, one might expect that placing an exponential bound on the counter values *upfront* does not change the problem much. But, in fact, the complexity jumps: the problem – which is equivalent to reachability in two-clock timed automata [15] – becomes PSPACE-complete [23]. One may say that, in this case, formal verification toolkit available for the reachability problem is not robust to this change in the problem setting.

In this paper we study a different seemingly benign variation of the standard reachability question. Consider one-counter automata in which transitions may test the value of the counter for *disequality* against a given integer (which depends on the transition). In other words, executions of the system can be blocked by disequality guards, which *prevent* the transition from being fired if the counter value is equal to a specified number. The initial motivation for studying this question comes from a model checking problem for flat Freeze LTL; see Demri and Sangnier [21] and Lechner, Mayr, Ouaknine, Pouly, and Worrell [35]. Additionally, recall that automata can be used for the modeling of imperative code; see, e.g., Hague and Lin [30, 31], as well as discussion in Section 2. Classical Minsky machines encode *if–then* conditionals with equality comparisons to constants, $x = k$. Simulating an *if–then–else* conditional of this type on a Minsky machine seems to require additional $O(k)$ states. If k is large, this growth in size may be exponential, even though *then* branches as well as increments $x \mathrel{+}= k$ and decrements $x \mathrel{-}= k$ can be encoded directly. (If the machine model is extended with $x \leq k$ comparisons, then the asymmetry between *then* and *else* disappears, but reachability becomes PSPACE-complete [23].)

One might expect that, since only a small number of configurations are forbidden (by the disequality guards) in the infinitely large configuration space, the complexity of the problem should not change significantly and existing techniques should be applicable. This conclusion, however, has remained elusive. For the problem of reachability in one-counter automata with disequality tests, the exponential upper bound on counter values necessary for witnessing reachability carries over. But, despite progress on related problems [9, 11] (including the settling of the complexity of the above-mentioned flat Freeze LTL model checking problem [11]), it has not been possible to pin down the complexity of this problem, which has been known to be NP-hard (even without disequality tests) and to belong to PSPACE (thanks to the exponential bound on the counter) [35]. The apparent simplicity of the problem contrasts with the lack of robustness of the available toolbox. It was recently shown by Almagor, Cohen, Pérez, Shirmohammadi, and Worrell [1] that the coverability (or state reachability) problem can be solved in polynomial time for this model, similarly to the standard 1-dimensional VASS without tests. The algorithm and its analysis, however, become sophisticated, and the complexity of reachability was left as a *challenging open problem.*

In the present paper, we make progress on this problem. Existing techniques need to be extended and developed significantly to handle seemingly benign disequality tests. We have been unable to find an easily verifiable witness for reachability, and instead show that *non-reachability* is witnessed by a form of invariants (or, more precisely, separators). The existence of counterexamples that violate such invariants can be checked in NP, thus placing the reachability problem for OCA with disequality tests into the second level of the polynomial hierarchy, namely in $\mathsf{coNP}^{\mathsf{NP}}$. This complexity class captures the complement of synthesis-type questions, which ask to find a single object (say, a circuit) that works correctly for all (exponentially many) inputs. In our problem, one configuration can reach another if and only if every potential invariant (of a form we describe) violates one of the invariance conditions; moreover, this violation can be checked in NP. In the presence of both equality and disequality tests, we need a slightly larger class $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$, at the third level of the hierarchy.

Our contributions. Traditionally, an invariant is an overapproximation of the set of reachable configurations which is inductive, i.e., closed under the transition relation. Our invariants are different in several ways:

1. We capture only some configurations within the reachability set, which form its *core*. Accordingly, we require a tailored notion of closure, namely closure under a restricted form of reachability relation.
2. Our invariants are leaky (*almost* inductive): an execution may escape the set. Allowing leaks is complemented by a separate controlling mechanism (check) that all leaks – which may occur at the interface between strongly connected components of the automaton – are safe.
3. To compose our local inductive invariants, i.e., those restricted to a single strongly connected component, the controlling mechanism for leaks relies on relaxed integer semantics for the execution. More precisely, we extend (to automata with disequality tests) a known technique [28] for lifting $\mathbb{Z}$-executions to actual executions.

Our notion of local invariants requires that we place a certain technical assumption at the interface (entry and exit points) of strongly connected components. To discharge this assumption, we use a combination of two invariants, one for the main (forward) VASS and another for the reverse VASS. Together, these two sets form a separator – a witness for non-reachability.

2 Related Work

Invariants. In formal verification, a forward exploration of countably infinite configuration spaces from the initial configuration, or a symmetrical backward exploration from the target, is a standard approach to reachability problems and targets bug finding. General heuristics can be used to improve such an exploration (see, for instance, the recent directed reachability algorithm [8]). However, in order to prove non-reachability, thus certifying the absence of bugs, an invariant-based approach is more popular.

Many techniques have been developed in the past for computing inductive invariants, depending on the structure of the underlying system based on counterexample-guided abstraction refinement [17], automata [32], property-directed reachability [3,13,14], and more generally in the abstract interpretation framework [18].

In vector addition systems, semilinear invariants [25] are sufficient for the general reachability problem [36]. Even if those invariants are intractable in general, for some instances, namely the control-state reachability problem, the implementation of efficient tools computing invariants (downward-closed sets in that case) is an active research area [7,22] with implementation of tools [6,24,33].

In this paper we focus on 1-dimensional VASS in the presence of equality and disequality tests; we call them *one-counter automata* (OCA) *with tests.* The notion of local inductive invariants with leaks, which we propose, provides a way to reduce the search space of inductive invariants, by specifying the shape of the "core" of the invariant (a union of arithmetic progressions within "bounded chains"), as well as restricting the problem to each strongly connected component one by one. We view this as a compositional approach for computing inductive invariants. As a theoretical application, we prove that the reachability problem for one-counter automata with tests is between NP and $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$, and in fact in $\mathsf{coNP}^{\mathsf{NP}}$ if only disequality tests are present.

The previous work on OCA with disequality tests by Almagor et al. [1] enables us to focus (subject to a technical assumption) on configurations in a small number of bounded chains (see Section 4). The structure of the set of reachable configurations in these chains admits a short description. At the core of our invariants are exactly such sets, and we need an appropriate notion of "inductiveness", a condition to control "leaks" that violate the assumption above, and a verification mechanism for all these conditions.

One-counter automata. OCA can be seen as an abstraction of pushdown automata, a widely used model of recursive systems. Conceptually very simple, OCA are at the heart of a number of results in formal verification; see, e.g., [2,4,34]. Multi-counter automata are used to model imperative code with numerical data types [30,31]; roughly speaking, a reachability query is expressed in logic, as a formula in existential linear integer arithmetic. In these two references an additional pushdown stack is available, capturing recursive function calls. We refer the reader to [16] for a retrospective on underlying "pumping" results for OCA, crucial for many of the recent results. There is also a rich history of research on behavioural equivalences and model checking for a variety of one-counter processes and systems; see, e.g., [10,27,47,48].

The above-mentioned result that reachability in OCA is NP-complete, by Haase, Kreutzer, Ouaknine, and Worrell [28], has recently been built upon to give a representation of the entire reachability relation in existential linear integer arithmetic, with an implementation available online, by Li, Chen, Wu, and Xia [40]. The idea of "lifting" candidate runs to actual runs, which is shown in [28] and which we develop further by adding support for disequality tests, has been used in other settings as well [5,37,41]. For example, a construction similar to our Lemma 7.1 is an element of the proof of a tight upper bound on the length of shortest runs in OCA without disequality tests [16]. In comparison to the latter paper, our construction need not consider divisibility properties of run lengths, but at the same time applies in a more general scenario: the updates of our OCA are specified in binary notation (that is, succinctly); and, naturally, our OCA may have disequality tests.

We already mentioned above that, despite appearing atypical at first glance, the disequality tests do in fact contribute to the modeling power: namely, when modeling code, these tests enable the simulation of the *else* branch in conditional statements comparing an integer variable for equality with some constant. The framework of Hague and Lin [30, 31] assumes that each counter variable can undergo at most k reversals (i.e., changes between "increasing" and "decreasing"), where k is fixed. This assumption is strong; without it, a reachability instance would require a logical formula of exponential size. Results of Haase et al. [28] and Li et al. [40] avoid this assumption, but, for the standard syntax of one-counter automata, *if–then–else* conditionals remain out of reach – or rather require an exponential expansion of the automaton. Our leaky invariants technique allows us to handle such conditionals with equality tests on counters, without assuming any bound on the number of reversals.

3 OCA with Equality and Disequality Tests

We denote by $\mathbb{Z}$ and $\mathbb{N}$ the set of all integers and all nonnegative integers, respectively.

A *constraint* is either an *equality test* of the form $x = k$ with $k \in \mathbb{N}$, a *disequality test* of the form $x \neq k$ with $k \in \mathbb{N}$, or simply `true`; x denotes here our counter, which is a nonnegative integer variable. Let $\mathcal{C}$ denote the set of all possible constraints. A *one-counter automaton (OCA) with equality and disequality tests* is a triplet $\mathcal{A} = (Q, \Delta, \tau)$, where Q is a finite set of *states*, $\Delta \subseteq Q \times \mathbb{Z} \times Q$ is a finite set of *transitions* and $\tau \colon Q \to \mathcal{C}$ is the *constraint function*. The automaton $\mathcal{A}$ is an *OCA with disequality tests* if the constraint function τ does not have any equality tests. We sometimes refer to the constraints as *guards*.[2]

Syntactically, $\mathcal{A}$ can be seen as an integer-weighted graph with directed edges between states. Viewed this way, $\mathcal{A}$ can be decomposed into a set of *strongly connected components (SCCs)*. The automaton $\mathcal{A}$ is *strongly connected* when it has one strongly connected component only. A *path* π in $\mathcal{A}$ is a sequence $\pi = (t_1, t_2, \dots, t_n)$ of transitions, where $t_i = (q_{i-1}, a_i, q_i)$ for each i and $n \geq 0$. We may refer to π as a q_0–q_n path. The *length* of such a path is $\mathsf{len}(\pi) \overset{\text{def}}{=} n$. The *effect* of π is $\mathsf{eff}(\pi) \overset{\text{def}}{=} \sum_{i=1}^{n} a_i$. A *cycle* is a path starting and ending at the same state; for $q \in Q$, a q-*cycle* is a q–q path. A path or cycle is *simple* if it contains no repetition of states, except that a *simple cycle* has the same starting and ending state. Every simple cycle has length less than or equal to $|Q|$.

The size of an OCA $\mathcal{A}$ is the bit size of its encoding, where all numbers are written in binary. We write $\|\Delta\|$ and $\|\tau\|$ to refer to the maximum absolute value of a transition update and test, respectively.

Configurations and runs. The semantics of $\mathcal{A}$ is defined based on the set of valid configurations and the reachability relation, as follows.

A *configuration* is a pair (q, z) comprising a state $q \in Q$ and a nonnegative integer $z \in \mathbb{N}$; we may refer to z as the *counter value*. We say that (q, z) is a *valid configuration* if it respects the constraint $\tau(q)$. Write $\mathit{Conf} \overset{\text{def}}{=} Q \times \mathbb{N}$ for the set of all configurations. Given two configurations $(q, z), (q', z')$ and $t \in \Delta$, we write $(q, z) \xrightarrow{t} (q', z')$ when $t = (q, z' - z, q')$; we denote by $(q, z) \to (q', z')$ the existence of such a transition. A *run* of $\mathcal{A}$ is a sequence $(q_0, z_0), \dots, (q_n, z_n)$ of valid configurations, for $n \geq 0$, such that there exists a path $(t_1, \dots, t_n)$ with $(q_{i-1}, z_{i-1}) \xrightarrow{t_i} (q_i, z_i)$. We say that (q_n, z_n) is *reachable* from (q_0, z_0)

[2] We use automata with constraints on states. Automata with constraints on transitions are, for our purposes, equivalent.

if there exists a run from (q_0, z_0) to (q_n, z_n). We write $(q_0, z_0) \xrightarrow{*} (q_n, z_n)$ to denote the existence of such a run. Given a path π, we write $(q_0, z_0) \xrightarrow{\pi} (q_n, z_n)$ if π *yields* a run from (q_0, z_0) to (q_n, z_n).

A path π has no hope to yield a run from (q, z) if $z + \mathsf{eff}(\pi') < 0$ for some prefix π' of π. We denote by $\mathsf{drop}(\pi)$ the maximum of $-\mathsf{eff}(\pi')$ over all prefixes π' of π, and call it the *drop* of π. Intuitively, $\mathsf{drop}(\pi)$ is the smallest counter value $z \in \mathbb{N}$ such that π, when applied from (q, z), remains nonnegative; note that hitting a guard is not a consideration here.

We use the following standard operators: $\mathsf{Post}(c) \stackrel{\text{def}}{=} \{c' \in \mathit{Conf} \mid c \to c'\}$ and $\mathsf{Pre}(c) \stackrel{\text{def}}{=} \{c' \in \mathit{Conf} \mid c' \to c\}$. For $X \subseteq \mathit{Conf}$, we write $\mathsf{Post}(X) \stackrel{\text{def}}{=} \bigcup_{c \in X} \mathsf{Post}(c)$ and $\mathsf{Pre}(X) \stackrel{\text{def}}{=} \bigcup_{c \in X} \mathsf{Pre}(c)$. Also, $\mathsf{Post}^*(X) \stackrel{\text{def}}{=} \{d \mid \exists c \in X : c \xrightarrow{*} d\}$ and $\mathsf{Pre}^*(X) \stackrel{\text{def}}{=} \{c \mid \exists d \in X : c \xrightarrow{*} d\}$.

For an OCA $\mathcal{A} = (Q, \Delta, \tau)$, we define the *reverse of* $\mathcal{A}$ as $\mathcal{A}^R \stackrel{\text{def}}{=} (Q, \Delta^R, \tau)$ where $(q, a, q') \in \Delta^R$ if and only if $(q', -a, q) \in \Delta$. Given configurations c and d and a path π in $\mathcal{A}$, we have $c \xrightarrow{*} d$ in $\mathcal{A}$ if and only if $d \xrightarrow{*} c$ in $\mathcal{A}^R$.

The reachability problem. We consider the following decision problem.

<u>REACHABILITY</u>
Input: An OCA $\mathcal{A}$ with equality and disequality tests, a valid initial configuration src, and a valid target configuration trg.
Output: Does $\mathsf{src} \xrightarrow{*} \mathsf{trg}$ hold?

The model of OCA with disequality tests has been studied in [35] and [1]. The latter paper provides polynomial-time algorithms for the *coverability problem*: "given src and a state q, does there exist z such that $\mathsf{src} \xrightarrow{*} (q, z)$?" and the related *unboundedness problem*: "is the set of configurations reachable from src infinite?". The reachability problem, however, is NP-hard even without tests, see Figure 1 (Left).

Equality tests. In the reachability problem in OCA with equality and disequality tests, the main technical challenge stems from disequality tests. Indeed, a state with an equality test only has one valid configuration hence need not be visited more than once.

> We now work with OCA with disequality tests only. We will discuss in Section 7.5 how our techniques are affected by the addition of equality tests.

4 Getting Familiar with Disequality Tests

In this section, we fix an OCA $\mathcal{A} = (Q, \Delta, \tau)$ with disequality tests and two valid configurations src and trg.

A configuration c is *bounded* when $\mathsf{Post}^*(c)$ is finite, and *unbounded* otherwise. It is known, although far from trivial, that one can decide boundedness in polynomial time.

▶ **Lemma 4.1** (see [1, Theorem 19]). *Given an OCA with disequality tests and a configuration c, it is decidable in polynomial time whether c is bounded or unbounded.*

A *candidate run* is a run except that neither the nonnegativity condition nor the disequality tests are necessarily respected. Formally, a candidate run is simply a sequence $(q_0, z_0), \ldots, (q_n, z_n)$ where all $(q_i, z_i) \in Q \times \mathbb{Z}$, $n \geq 0$, and such that there exist transitions $(q_{i-1}, a_i, q_i) \in \Delta$ with $z_{i+1} = z_i + a_i$ for all $i \in \{1, \ldots, n\}$. We write $(q_0, z_0) \xrightarrow[\mathbb{Z}]{*} (q_n, z_n)$.

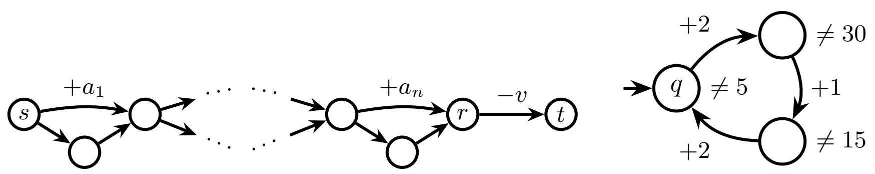

Figure 1 Left. This OCA without tests is constructed from an instance of the subset sum problem $(a_1, \ldots, a_n, v)$; this is in fact how the reachability problem in OCA without tests is proved to be NP-hard in [28]. Configuration $(t, 0)$ can be reached from configuration $(s, 0)$ whenever there exists a subset of $\{a_1, \ldots, a_n\}$ whose elements sum up to v. Note that all unlabelled transitions have update zero. The set of configurations reachable from $(s, 0)$ can have size exponential in n, and its structure is unwieldy.
Right. (For Section 4.) The named state q belongs to Q_+ since there is a simple q-cycle with positive effect. There are six bounded chains of configurations at q. The disequality test $\neq 5$ bounds the counter values with residue 0 modulo 5, so $\{(q, 0)\}$ and $\{(q, 5)\}$ are bounded chains. The disequality test $\neq 30$ bounds the counter values with residue 3 modulo 5, so $\{(q, 3), (q, 8), (q, 13), (q, 18), (q, 23), (q, 28)\}$ is a bounded chain. The disequality test $\neq 15$ bounds the counter values with residue 2 modulo 5, so $\{(q, 2), (q, 7), (q, 12)\}$ is a bounded chain.

An ingredient of the NP upper bound for reachability in OCA *without* disequality tests [28] is establishing conditions under which a candidate run can be *lifted* to a run. We adapt the argument to OCA with disequality tests.

▶ **Lemma 4.2.** *Let $\mathcal{A}$ be a strongly connected OCA with disequality tests. If* src *is unbounded in $\mathcal{A}$ and* trg *is unbounded in $\mathcal{A}^R$, then there is a run from* src *to* trg *in $\mathcal{A}$* ($\mathsf{src} \xrightarrow{*} \mathsf{trg}$) *if and only if there is a candidate run from* src *to* trg *in $\mathcal{A}$* ($\mathsf{src} \xrightarrow[\mathbb{Z}]{*} \mathsf{trg}$).

The hypothesis that $\mathcal{A}$ is strongly connected is crucial. Indeed, if $\mathcal{A}$ is strongly connected and $\mathsf{src} = (s, v)$ is unbounded in $\mathcal{A}$, then there is a cycle of positive effect that, from src, can be applied infinitely often to reach (s, z) with z arbitrarily large. If $\mathcal{A}$ is not strongly connected, it could be that the positive cycles that make src unbounded are in another SCC and that the set $\{z \mid (s, v) \xrightarrow{*} (s, z)\}$ is finite.

Let $Q_+ \subseteq Q$ be the set of states $q \in Q$ such that there exists a q-cycle γ with $\mathsf{len}(\gamma) \leq |Q|$ and with $\mathsf{eff}(\gamma) > 0$. For each $q \in Q_+$, let γ_q be such a q-cycle with minimal drop. **We fix this choice for the remainder of the paper.** We define

$$\mathit{Conf}_+ \stackrel{\text{def}}{=} \{(q, z) : q \in Q_+ \text{ and } z \geq \mathsf{drop}(\gamma_q)\}.$$

▶ **Lemma 4.3.** *There is a polynomial-time algorithm to identify Q_+ and to choose cycles γ_q for all $q \in Q_+$. Moreover, membership in Conf_+ can be decided in polynomial time.*

The proof of Lemma 4.3 can be found in Appendix A.

▶ Remark 4.4. Our choice of Q_+ differs slightly from the definition found in [1]: we use short cycles ($\mathsf{len}(\gamma) \leq |Q|$) rather than simple cycles. For simple cycles, the ability to compute, in polynomial time, the minimal drop of a positive-effect simple q-cycle (for each $q \in Q$) is not justified in [1]. In fact, in Appendix B, we prove that deciding, for a given OCA without tests $\mathcal{A}$ and a given state q, whether there exists a positive-effect simple q-cycle in $\mathcal{A}$ is an NP-complete problem. However, all constructions and arguments of [1] appear to be insensitive to the replacement of "simple cycles" by "short cycles". As a result, we can still use polynomial-time algorithms for coverability and for unboundedness in OCA with disequality tests (1-VASS with disequality tests).

The set of all $(q,z) \in \mathit{Conf}_+$ can be partitioned into *q-chains*. For each $q \in Q_+$, let $\mathit{Conf}_+(q) = (\{q\} \times \mathbb{N}) \cap \mathit{Conf}_+$. A q-chain C is a maximal non-empty subset $C \subseteq \mathit{Conf}_+(q)$ such that, for every two distinct $c, c' \in C$, either c is reachable from c' by iterating γ_q, or vice versa. In other words, C is a non-empty minimal subset of $\mathit{Conf}_+(q)$ (with respect to set inclusion) such that, for all $c \in C$ and all $c' \in \mathit{Conf}$, if $c \xrightarrow{\gamma_q} c'$ or $c' \xrightarrow{\gamma_q} c$ then $c' \in C$.

A q-chain is *bounded* if it is a finite set, otherwise it is *unbounded*. Note that configurations in unbounded chains are all themselves unbounded, but configurations in bounded chains need not be bounded (they may be unbounded). Because the number of disequality guards that a cycle γ_q may encounter is small, so is the total number of bounded chains.

▶ **Lemma 4.5** (see [1, Remark 6]). *There are at most $2|Q|^2$ bounded chains.*

Given a chain C, the counter values z of every $(q,z) \in C$ have the same remainder modulo $\mathsf{eff}(\gamma_q)$. Henceforth, a bounded q-chain can be described as $[\ell, u] \cap (r + \mathsf{eff}(\gamma_q) \cdot \mathbb{N})$ where $[\ell, u]$ is an interval of nonnegative integers and $r + \mathsf{eff}(\gamma_q) \cdot \mathbb{N}$ is an arithmetic progression with initial term r and difference $\mathsf{eff}(\gamma_q)$. Since the OCA $\mathcal{A}$ is encoded in binary, the values of l, u, r, and $|\gamma_q|$ may be exponential in the size of $\mathcal{A}$. See Figure 1 (Left) for an example.

5 Pessimistic Reachability

In this section, we exhibit a family run of runs, namely pessimistic runs, that are guaranteed to admit an NP certificate. This will already enable us to prove, in Section 6, that the reachability problem is in $\mathsf{coNP}^{\mathsf{NP}}$ in the special case where the OCA is strongly connected.

Let $\mathcal{A}$ be an OCA with disequality tests. We call a run of $\mathcal{A}$ *pessimistic* if none of its configurations are in Conf_+, except possibly the first one. Of course, some pessimistic runs may be exponentially long relative to the size of $\mathcal{A}$; however, we provide a way to handle them. For $S \subseteq \mathit{Conf}_+$, we write $\mathsf{Post}^*_-(S)$ for the set of configurations reachable from S using only pessimistic runs. In particular, $S \subseteq \mathsf{Post}^*_-(S)$.

Consider the following decision problem:

PESSIMISTIC REACHABILITY

Input: An OCA $\mathcal{A}$ with disequality tests, and two configurations src and trg.

Output: Is there a pessimistic run from src to trg in $\mathcal{A}$?

Pessimistic runs turn out to be very handy, not least because we can adapt an existing “flow” technique [28] to decide pessimistic reachability.

▶ **Lemma 5.1.** *The pessimistic reachability problem is in* NP.

In a nutshell, the idea [28] is to guess how many times the run traverses each transition. The guessed numbers are subject to polynomial-time checkable balance and connectivity conditions, akin to, e.g., [46]. However, we cannot check whether the (possibly very long) run constructed from the flow violates disequality constraints, so the technique cannot be applied directly.

Our solution uses the pessimism of the run. Let $x \neq g$ be a guard on state q. We split the run in two: in the first part, all visits to q are above g; then the run *jumps* the guard so that, in the second part, all visits are below g. This way, with at most $|Q|$ splits, we can reduce the problem to the case in which the run does not jump *any* disequality guard (always staying above or below each of them).

6 Reachability in Strongly Connected OCA

In this section, for pedagogical purposes, we study the particular case where the OCA is strongly connected. The case with multiple SCCs presented in Section 7 is more technical but relies on the same key idea.

▶ **Theorem 6.1.** *The reachability problem for strongly connected OCA with disequality tests belongs to the complexity class* $\mathsf{coNP}^{\mathsf{NP}}$.

We sketch the proof of Theorem 6.1 below. Throughout the section, we fix a strongly connected OCA with disequality tests $\mathcal{A}$ and two configurations src and trg, and we are interested in whether $\mathsf{src} \xrightarrow{*} \mathsf{trg}$.

6.1 Ruling Out the Unbounded Case

By Lemma 4.1, given an instance of reachability, we can check in polynomial time whether src is unbounded in $\mathcal{A}$ and trg is unbounded in $\mathcal{A}^R$. If both are true, then, by Lemma 4.2, it suffices to determine whether there exists a candidate run from src to trg. The existence of a candidate run can be decided in NP (e.g., using integer linear programming, see [12]). This case will not affect our complexity result because $\mathsf{NP} \subseteq \mathsf{coNP}^{\mathsf{NP}}$. Thus, without loss of generality, we may assume that src is bounded in $\mathcal{A}$ or trg is bounded in $\mathcal{A}^R$. Moreover, if trg is bounded in $\mathcal{A}^R$, we symmetrically work with $\mathcal{A}^R$ instead of $\mathcal{A}$.

In the remainder of this section, we assume that src is bounded in $\mathcal{A}$.

6.2 Inductive Invariants in the Bounded Case

We will show that $\mathsf{src} \overset{*}{\not\to} \mathsf{trg}$ if and only if there exists a certificate of a particular shape witnessing this non-reachability. This certificate takes the form of an inductive invariant separating src and trg. The exact set of configurations comprising this inductive invariant is unwieldy, so we concentrate on its *core* instead. This set of core configurations admits a short representation, as follows.

We call an *arithmetic progression* on state $q \in Q$ a set of configurations $\{(q, v) \mid \ell \leq v \leq L \wedge \exists k \in \mathbb{N}, v = kp + s\}$ with $p, s, \ell, L \in \mathbb{N}$. An arithmetic progression can be specified by writing q and the numbers p, s, ℓ, L. A set of configurations *has a concise description* if it is a union of at most $2|Q|^2 + 1$ arithmetic progressions whose configurations have counter value bounded by $2|Q| \cdot \|\Delta\| \cdot \|\tau\|$. Such a set can be described in polynomial space.

The set of all configurations in bounded chains has a concise description. This also holds for the set R of all *reachable* configurations in bounded chains: indeed, if a configuration of a chain can be reached, the same is true for all configurations above in the same chain. Because src is bounded, unbounded chains cannot be reached, and this R is in fact the set of reachable configurations in all chains. (Observe that runs that reach configurations from R may well visit the complement of Conf_+.)

▶ **Lemma 6.2.** *The set R of reachable configurations in Conf_+ has a concise description.*

Intuitively, R is our desired "core invariant", and the desired invariant is the set $\mathsf{Post}^*_{-}(R)$. However, when given a set I, it is not easy to check whether I is actually equal to R. Instead, the following theorem defines possible invariant cores by 3 conditions.

Conditions involving src and trg are self-explanatory. Set inclusion $\mathsf{Post}(\mathsf{Post}^*_-(I)) \subseteq \mathsf{Post}^*_-(I)$ would express inductiveness (closure of the set under $\mathsf{Post}(\cdot)$). However, verifying this condition is computationally expensive, and we replace it with a version that "focuses" on the core only, and thus has I rather than $\mathsf{Post}^*_-(I)$ on the right-hand side.

▶ **Theorem 6.3.** *Suppose* src *is bounded in* $\mathcal{A}$. *Then* $\mathsf{src} \overset{*}{\nrightarrow} \mathsf{trg}$ *if and only if there exists a set* $I \subseteq \mathit{Conf}_+ \cup \{\mathsf{src}\}$ *with concise description such that:*
(Cond1) $\mathsf{src} \in I$*,*
(Cond2) $\mathsf{trg} \notin \mathsf{Post}^*_-(I)$*, and*
(Cond3) $\mathsf{Post}(\mathsf{Post}^*_-(I)) \cap \mathit{Conf}_+ \subseteq I$*.*

Proof. First, assume that there is such a set I. Because $\mathsf{trg} \notin \mathsf{Post}^*_-(I)$ by **(Cond2)**, it suffices to prove that $\mathsf{Post}^*(\mathsf{src}) \subseteq \mathsf{Post}^*_-(I)$. We proceed by induction on the length of the run from src to $c \in \mathsf{Post}^*(\mathsf{src})$. The base of induction is **(Cond1)**. Assume that we have $d \in \mathsf{Post}^*_-(I)$ and $d \to c$. If $c \notin \mathit{Conf}_+$ then we have a pessimistic run from I to c, so $c \in \mathsf{Post}^*_-(I)$. If $c \in \mathit{Conf}_+$ then $c \in \mathsf{Post}(\mathsf{Post}^*_-(I)) \cap \mathit{Conf}_+$, hence $c \in I$ by **(Cond3)**.
For the other direction, assume that trg is not reachable from src. Let $I \overset{\text{def}}{=} R \cup \{\mathsf{src}\}$; by Lemma 6.2, I has a concise description. **(Cond1)** and **(Cond2)** are trivially satisfied. Moreover, $\mathsf{Post}(\mathsf{Post}^*_-(I)) \cap \mathit{Conf}_+ \subseteq \mathsf{Post}^*(\mathsf{src}) \cap \mathit{Conf}_+ \subseteq I$, hence **(Cond3)** is satisfied. ◀

6.3 The Complexity of Reachability in Strongly Connected OCA

We now prove that reachability is in $\mathsf{coNP}^{\mathsf{NP}}$ by, equivalently, proving that *non*-reachability is in $\mathsf{NP}^{\mathsf{NP}}$. Roughly speaking, a problem is in $\mathsf{NP}^{\mathsf{NP}}$ whenever this problem is solvable in non-deterministic polynomial time by a Turing machine which has access to an oracle for some NP-complete problem. The oracle is a black box that may provide the answer to any problem in NP (and therefore to any problem in coNP).

As argued in Section 6.1, we assume with no loss of generality that src is bounded. By Theorem 6.3, we have $\mathsf{src} \overset{*}{\nrightarrow} \mathsf{trg}$ if and only if there exists I satisfying the three conditions **(Cond1)**, **(Cond2)**, and **(Cond3)**. Moreover, by the same theorem, I can be assumed to have a concise description. Thus, we can guess such a set I in non-deterministic polynomial time. It remains to prove that the verification that a set I satisfies the three conditions can be performed using an NP oracle. To this end, we prove that this verification is a coNP problem. Indeed, I does *not* satisfy the three conditions when:

- either $\mathsf{src} \notin I$ (which can be checked efficiently),
- or $\mathsf{trg} \in \mathsf{Post}^*_-(I)$ (this is when there is a *small* configuration c such that $c \in I$ and $\mathsf{trg} \in \mathsf{Post}^*_-(c)$),
- or there are some *small* configurations c and d such that $c \in I$, $d \in \mathsf{Post}^*_-(c)$, and some successor of d belongs to Conf_+ but not to I.

The adjective *small* should here be understood as "bounded by an exponential in the size of $\mathcal{A}$, src, and trg". In fact, it is fairly easy to obtain an exponential bound on configurations to consider. Thanks to Lemma 5.1, verification of both whether there is a $c \in I$ such that $\mathsf{trg} \in \mathsf{Post}^*_-(c)$ and whether there exist $c \in I$, $d \in \mathsf{Post}^*_-(c)$, and $e \in \mathsf{Post}(d)$ such that $e \notin I$ are in NP. Since membership in Conf_+ can be checked in polynomial time by Lemma 4.3, the entire third condition is also an NP condition. This completes the proof of Theorem 6.1.

7 Combining Strongly Connected Components

In this section, we extend the techniques from Section 6 to the general case in which the OCA is not assumed to be strongly connected. We fix an OCA $\mathcal{A}$ with disequality tests and two configurations src and trg.

We first highlight why the techniques developed above do not apply to this general case. In Section 6, the hypothesis that $\mathcal{A}$ is strongly connected was necessary for the application of Lemma 4.2. When $\mathcal{A}$ is not strongly connected, knowing that src is unbounded is no longer satisfactory. Indeed, it no longer implies the existence of a positive cycle involving its state, as the positive cycle allowing us to pump up could be in another SCC. We need to be able to specify whether a configuration is unbounded *within its own SCC* or not.

7.1 Locally Bounded Configurations and Runs

Locally bounded configurations. Given a SCC S of $\mathcal{A}$, we denote by $\mathcal{A}_S$ the automaton obtained when restricting $\mathcal{A}$ to states and transitions within S. A configuration c is *locally bounded* if c is bounded in $\mathcal{A}_S$ where S is the SCC of c. We denote by L the set of all locally bounded configurations (and by L^R in $\mathcal{A}^R$). Configurations that are not locally bounded are referred to as *locally unbounded*. We generalise the lifting technique from Lemma 4.2.

▶ **Lemma 7.1** (Lifting). *For all $c \notin L$ and $d \notin L^R$, we have $c \xrightarrow{*} d$ if and only if $c \xrightarrow[\mathbb{Z}]{*} d$.*

Locally bounded runs. A run $c \xrightarrow{\pi} d$ is said to be *locally bounded* if all configurations visited by the run are locally bounded. We denote such a run by $c \xrightarrow[L]{\pi} d$, and denote its existence by $c \xrightarrow[L]{*} d$. Notice that a locally bounded run may go through several SCCs. Moreover, a run starting from a locally bounded configuration is not always locally bounded: once it goes to a new SCC, it may visit configurations that are not locally bounded. We define the locally bounded counterpart LPost^*_- of the pessimistic post-star operator: $d \in \mathsf{LPost}^*_-(c)$ if there is a pessimistic and locally bounded run from c to d. We extend this definition to sets of configurations X in the usual way. Dually, we also define, for every set of configurations X, the set $\mathsf{LPre}^*_+(X)$ as the same notion but in the reverse OCA $\mathcal{A}^R$. We extend Lemma 5.1 to these new operators.

▶ **Lemma 7.2.** *Given $c, d \in Conf$, deciding whether $d \in \mathsf{LPost}^*_-(c)$ is in NP.*

Proof sketch. We split the run on its transitions between SCCs. We apply Lemma 5.1 on the portions remaining in one SCC. Since the run is pessimistic, we can bound all the counter values in it. The run is locally bounded when the first configuration visited in each SCC is locally bounded, which is checked using Lemma 4.1. ◀

7.2 Leaky Invariants

Unlike in the strongly-connected case, a single invariant construction is not sufficient for our needs. Indeed, if src is locally bounded but unbounded, then one could imagine the invariant technique from Theorem 6.3 applied to the SCC S of src, but then this invariant would not apply to other SCCs. For example, there could be runs that are locally bounded in the SCC S_{src} of src but not in the SCC S_{trg} of trg, making the invariant inapplicable. Instead, assuming that trg is locally bounded in $\mathcal{A}^R$, one may consider in the SCC of trg an invariant constructed in the reverse automaton $\mathcal{A}^R$. We therefore employ a pair of invariants, one for $\mathcal{A}$ (the *forward invariant*) and another one for its reverse $\mathcal{A}^R$ (the *backward invariant*). The two invariants will induce two sets of configurations that, in a negative instance of the reachability problem, separate the source and target.

The following lemma will allow us to avoid treating src and trg separately. The set $Conf^R_+$ is defined as the counterpart of $Conf_+$ in $\mathcal{A}^R$.

▶ **Lemma 7.3.** *We may assume that* $\mathsf{src} \in \mathit{Conf}_+ \cap L$ *and* $\mathsf{trg} \in \mathit{Conf}_+^R \cap L^R$.

We now define our notion of a leaky invariant. As in Section 6, we represent the invariants using *core* sets of configurations that can be succinctly described, denoted by I and J. Our invariants must be inductive in the following weak sense:

▶ **Condition 7.4.** *Let* $I \subseteq \mathit{Conf}_+ \cap L$ *and* $J \subseteq \mathit{Conf}_+^R \cap L^R$ *be sets of configurations. The pair* (I, J) *is* ***inductive*** *if*

(Ind)
$$\mathsf{Post}(\mathsf{LPost}_-^*(I)) \cap \mathit{Conf}_+ \cap L \subseteq I \quad \textit{and}$$
$$\mathsf{Pre}(\mathsf{LPre}_+^*(J)) \cap \mathit{Conf}_+^R \cap L^R \subseteq J.$$

Notice that I and J play symmetric roles in $\mathcal{A}$ and $\mathcal{A}^R$. We now provide some intuition for the (forward) inductive condition for I. The set I only contains configurations from $\mathit{Conf}_+ \cap L$, because the set $\mathit{Conf}_+ \cap L$ has a regular structure thanks to bounded chains. The set I is, again, only the *core* of the invariant. The full invariant[3] is $\mathsf{Post}_-^*(I) \cup \mathsf{Post}(\mathsf{Post}_-^*(I))$, but this set is not easily described (see Remark 7.8). This explains why we use the composition $\mathsf{Post}(\mathsf{LPost}_-^*(\cdot))$ instead of the single-step $\mathsf{Post}(\cdot)$ operator traditionally used to define inductiveness.

▶ Remark 7.5. We refer to our invariants as *leaky*, because they are not inductive in the traditional sense. Indeed, our invariants are "focused" on locally bounded configurations, and can be escaped by transitions to locally unbounded configurations. This leak may, however, only happen with transitions going from one SCC to another.

▶ **Condition 7.6.** *Let* $\mathcal{I}, \mathcal{J} \subseteq \mathit{Conf}$ *be sets of configurations. The pair* $(\mathcal{I}, \mathcal{J})$ *is a* ***separator*** *if, for all* $c \in \mathcal{I}$ *and* $d \in \mathcal{J}$,
(Sep1) $c \not\to d$; *and*
(Sep2) *if* $c \notin L$ *and* $d \notin L^R$, *then* $c \underset{\mathbb{Z}}{\overset{*}{\not\to}} d$.

Firstly, **(Sep1)** will forbid $\mathcal{I}$ and $\mathcal{J}$ from being connected by a single step. Secondly, **(Sep2)** will forbid connection between $\mathcal{I}$ and $\mathcal{J}$ using the lifting technique of Lemma 7.1. This does not, in general, prevent the existence of runs from $\mathcal{I}$ to $\mathcal{J}$; it will do so, however, for our leaky invariants that combine Conditions 7.4 and 7.6.

▶ **Definition 7.7.** *Let* $I \subseteq \mathit{Conf}_+ \cap L$ *and* $J \subseteq \mathit{Conf}_+^R \cap L^R$. *Consider the sets*

$$\mathcal{I} := \mathsf{Post}_-^*(I) \cup \mathsf{Post}(\mathsf{Post}_-^*(I)) \quad \textit{and}$$
$$\mathcal{J} := \mathsf{Pre}_+^*(J) \cup \mathsf{Pre}(\mathsf{Pre}_+^*(J)).$$

We call the pair (I, J) *a* non-reachability witness *for* src *and* trg *if* (I, J) *is inductive,* $(\mathcal{I}, \mathcal{J})$ *is a separator,* $\mathsf{src} \in I$, *and* $\mathsf{trg} \in J$.

The pair of sets (I, J) forms the *core* of the invariant, namely I represents the *forward leaky invariant* and J represents the *backward leaky invariant*. In this case we also say that (I, J) *induces* the separator $(\mathcal{I}, \mathcal{J})$. A visualisation of a pair (I, J) and its induced separator $(\mathcal{I}, \mathcal{J})$ can be seen in Figure 2. A helpful intuition is that $\mathcal{I}$ is approximately $\mathsf{Post}_-^*(I)$ (and similarly $\mathcal{J}$ is approximately $\mathsf{Pre}_+^*(J)$). One additional step of $\mathsf{Post}(\cdot)$ (and $\mathsf{Pre}(\cdot)$, respectively) ensures that the "outer boundary" of this closure should also be included in the set.

[3] Our invariant is $\mathsf{Post}_-^*(I) \cup \mathsf{Post}(\mathsf{Post}_-^*(I))$ but the operator that appears in Condition 7.4 is $\mathsf{LPost}_-^*(\cdot)$. In Appendix C, we discuss the issues encountered if $\mathsf{Post}_-^*(\cdot) \cup \mathsf{Post}(\mathsf{Post}_-^*(\cdot))$ was used in Condition 7.4.

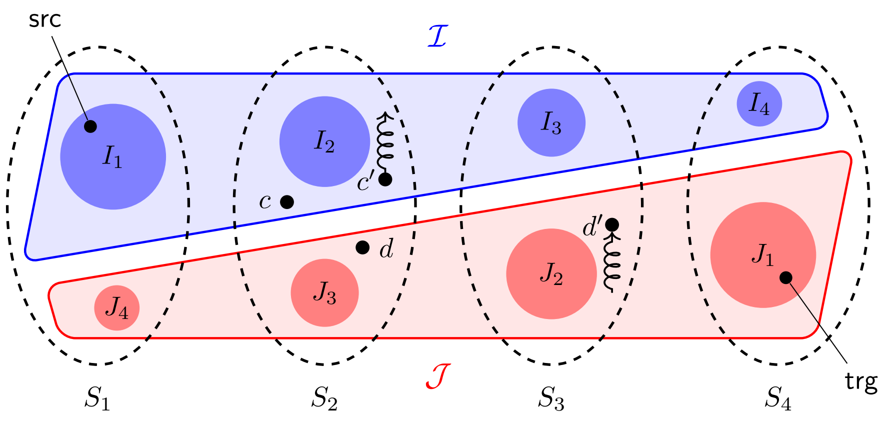

Figure 2 Core inductive sets and the separator they induce. The core of the forward leaky invariant is $I = I_1 \cup I_2 \cup I_3 \cup I_4$ (the blue circular sets) and the core of the backward leaky invariant is $J = J_1 \cup J_2 \cup J_3 \cup J_4$ (the red circular sets). The induced separator $(\mathcal{I}, \mathcal{J})$ is shown as blue and red rounded quadrilaterals containing the core sets. Notice that $\mathsf{src} \in I$ and $\mathsf{trg} \in J$. The upwards coiled arrow from c' represents that c' is locally unbounded in the SCC S_2 and the downwards coiled arrow from d' represents that d' is locally unbounded in the SCC S_3^R. Note also the separator conditions: Condition 7.6 **(Sep1)** means that configurations $c \in \mathcal{I}$ and $d \in \mathcal{J}$ cannot reach one another by one transition, so $c \not\to d$; Condition 7.6 **(Sep2)** means that unbounded configurations $c' \in \mathcal{I}$ and $d' \in \mathcal{J}$ cannot reach one another via a candidate run, so $c \underset{\mathbb{Z}}{\overset{*}{\not\to}} d$.

▶ Remark 7.8. As in Section 6, our representation of invariants refers to their core only, i.e., the pair (I, J). The example in Fig. 1 (left) demonstrates that the set $\mathsf{Post}^*_-(I) \cup \mathsf{Post}(\mathsf{Post}^*_-(I))$ does not always have a tractable description. The set of all possible sums of subsets has no convenient description, therefore we want it to be captured by $\mathsf{Post}^*_-(I) \cup \mathsf{Post}(\mathsf{Post}^*_-(I))$ only and not by I itself.

▶ **Theorem 7.9.** *In an OCA $\mathcal{A}$ with disequality tests,* trg *is* not *reachable from* src *if and only if there exists a non-reachability witness. Moreover, in this case, there is always a non-reachability witness with a concise description.*

In Section 7.3, we define "perfect cores", which we use in Section 7.4 to sketch a proof of Theorem 7.9 (details can be found in the full version).

7.3 Perfect Cores

Condition **(Ind)** on the core of leaky invariants captures a weak inductiveness property, which is central to our approach. We will now discuss two features of this condition that are used in the proof of Theorem 7.9.

Our conditions capture a specific invariant, which we now define. Consider the set

$$B \overset{\text{def}}{=} \{c \in \mathit{Conf} : \mathsf{src} \xrightarrow[L]{*} c\}$$

In words, B contains all configurations reachable from src using locally bounded runs. In line with ideas from Section 6, we do not want to store B entirely, so we will restrict the core to configurations in bounded chains. We call *perfect core* the set $B \cap \mathit{Conf}_+$; the term *perfect* is motivated by the fact that Definition 7.7 *aims* to capture this set exactly. Similarly, let $B^R := \{c \in \mathit{Conf} : c \xrightarrow[L^R]{*} \mathsf{trg}\}$. The *perfect core* in the reverse automaton is $B^R \cap \mathit{Conf}_+^R$.

The two features of **(Ind)** are summarised in the following two lemmas. We use the words "sound" and "complete" to characterise the relationship between Condition 7.4 (as part of Definition 7.7) and the perfect cores defined above. Completeness expresses that in *every instance* of non-reachability, the perfect cores defined above induce a non-reachability invariant. Conversely, soundness states that *every invariant* must contain all configurations from the perfect cores. (Thus, the perfect core are the smallest possible invariants.)

▶ **Lemma 7.10** (Soundness). *For all $I \subseteq Conf_+ \cap L$ and $J \subseteq Conf_+^R \cap L^R$ such that* $\mathsf{src} \in I$ *and* $\mathsf{trg} \in J$*, if (I, J) is inductive (Condition 7.4), then $B \cap Conf_+ \subseteq I$ and $B^R \cap Conf_+^R \subseteq J$.*

Proof sketch. Condition 7.4 for I gives $\mathsf{Post}(\mathsf{LPost}_-^*(I)) \cap Conf_+ \cap L \subseteq I$. Let $c \in B \cap Conf_+$ be a configuration of the perfect core. Thus, src reaches c by a locally bounded run. It is not always true that $c \in \mathsf{Post}(\mathsf{LPost}_-^*(\mathsf{src}))$ because this run does not have to be pessimistic: it may observe configurations in $Conf_+$. We prove by induction that all configurations in $Conf_+$ along the run are in I, using the property that $\mathsf{Post}(\mathsf{LPost}_-^*(I)) \cap Conf_+ \cap L \subseteq I$ once for each such configuration; this eventually proves that $c \in I$. The proof is analogous for J. ◀

▶ **Lemma 7.11** (Completeness). *If $I = B \cap Conf_+$ and $J = B^R \cap Conf_+^R$, then (I, J) is inductive (Condition 7.4).*

Proof sketch. We prove that $\mathsf{Post}(\mathsf{LPost}_-^*(B \cap Conf_+)) \cap Conf_+ \cap L \subseteq B \cap Conf_+$. Let $c \in Conf_+$ be locally bounded and belong to $\mathsf{Post}(\mathsf{LPost}_-^*(B \cap Conf_+))$. All configurations in $\mathsf{LPost}_-^*(B \cap Conf_+)$ are in B by the definition of B, so c can be reached in one step from a configuration $d \in B$. By definition, d is reachable from src with a locally bounded run; since c is itself locally bounded, this is also true for c, and so $c \in B$. The case of J is similar. ◀

7.4 Non-reachability Witnesses and Their Complexity

▶ **Theorem 7.9.** *In an OCA $\mathcal{A}$ with disequality tests,* trg *is* not *reachable from* src *if and only if there exists a non-reachability witness. Moreover, in this case, there is always a non-reachability witness with a concise description.*

Proof sketch. First, if $\mathsf{src} \overset{*}{\not\rightarrow} \mathsf{trg}$ then the perfect cores $I = B \cap Conf_+$ and $J = B^R \cap Conf_+^R$ form a non-reachability witness. Indeed, by Lemma 7.11, (I, J) is inductive. Moreover, the induced $\mathcal{I} = \mathsf{Post}_-^*(I) \cup \mathsf{Post}(\mathsf{Post}_-^*(I))$ and $\mathcal{J} = \mathsf{Pre}_+^*(J) \cup \mathsf{Pre}(\mathsf{Pre}_+^*(J))$ form a separator. We have $\mathcal{I} \subseteq \mathsf{Post}^*(\mathsf{src})$ and $J \subseteq \mathsf{Pre}^*(\mathsf{trg})$, proving Condition 7.6 **(Sep1)**. If Condition 7.6 **(Sep2)** fails, Lemma 7.1 yields a contradiction. Moreover, I and J have a concise description thanks to bounded chains.

Conversely, suppose there is a non-reachability witness (I, J). Assume for the sake of contradiction that $\mathsf{src} \xrightarrow{*} \mathsf{trg}$. By Lemma 7.10, since (I, J) is inductive, $B \cap Conf_+ \subseteq I$ and $B^R \cap Conf_+^R \subseteq J$. Consider a run from src to trg. It must leave $\mathcal{I} = \mathsf{Post}_-^*(I) \cup \mathsf{Post}(\mathsf{Post}_-^*(I))$ therefore it visits locally unbounded configurations. Let c be the first such configuration visited. Similarly, let d be the last visited configuration that is locally unbounded in $\mathcal{A}^R$. First, if c occurs before d, then Condition 7.6 **(Sep2)** is violated. Second, if c occurs after d, then there is an overlap in the runs from src to c and from d to trg. The overlap must be in $\mathcal{I} \cap \mathcal{J}$, leading to a violation of Condition 7.6 **(Sep1)**. ◀

▶ **Theorem 7.12.** *The reachability problem for OCA with disequality tests belongs to the complexity class* $\mathsf{coNP}^{\mathsf{NP}}$.

Proof sketch. We use Theorem 7.9, deciding the existence of a non-reachability witness in $\mathsf{NP}^{\mathsf{NP}}$. Recall that $\mathsf{NP}^{\mathsf{NP}}$ is introduced in Section 6.3. Let a pair (I, J) be given; we show that a violation of the conditions for being a non-reachability witness can be checked in NP.

- One can check in polynomial time whether $\mathsf{src} \in I$ and $\mathsf{trg} \in J$.
- Violation of Condition 7.4 **(Ind)** is an NP property. Indeed, this follows because membership in $\mathsf{LPost}^*_-(\cdot)$ is in NP (by Lemma 7.2) and membership in Conf_+ and L is polynomial-time checkable (by Lemma 4.3 and Lemma 4.1, respectively).
- Violation of Condition 7.6 **(Sep1)** is in NP, because $\mathcal{I} := \mathsf{Post}^*_-(I) \cup \mathsf{Post}(\mathsf{Post}^*_-(I))$, $\mathcal{J} := \mathsf{Pre}^*_+(J) \cup \mathsf{Pre}(\mathsf{Pre}^*_+(J))$, and membership of given configurations in the pessimistic post-star (optimistic pre-star, respectively) is in NP by Lemma 5.1. This assumes that we have an exponential bound on relevant configurations.
- To check violation of Condition 7.6 **(Sep2)** in NP, we again use Lemma 4.1 for L, as well as the fact that the existence of a candidate run is in NP (by integer programming). ◀

7.5 Adding Equality Tests

The previous techniques have been developed for OCA with disequality tests only. In particular, the lifting argument of Lemma 7.1 does not hold in the presence of equality tests: candidate runs that visit a state with an equality test cannot be lifted to greater counter values. However, at the cost of increasing the complexity, one can handle equality tests.

Complexity class $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$ consists of decision problems solvable in polynomial time with access to an $\mathsf{NP}^{\mathsf{NP}}$ oracle (which can solve $\mathsf{NP}^{\mathsf{NP}}$ problems in one step).

▶ **Corollary 7.13.** *The reachability problem for OCA with equality and disequality tests belongs to the complexity class* $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$.

Proof. Let $\mathcal{A}$ be such an OCA with tests, and src and trg two configurations. Denote by $\mathit{Conf}_=$ the set of valid configurations at states with equality tests; $|\mathit{Conf}_=|$ does not exceed the number of states in $\mathcal{A}$. Consider the OCA with disequality tests $\mathcal{A}'$ that is obtained by deleting all states with equality tests (and incident transitions) from $\mathcal{A}$. By Theorem 7.12, with an $\mathsf{NP}^{\mathsf{NP}}$ oracle we can build a graph with vertex set $\mathit{Conf}_= \cup \{\mathsf{src}, \mathsf{trg}\}$ and edge set $\{(c, d) \mid c \xrightarrow{*} d \text{ in } \mathcal{A}'\}$. Depth-first search in this graph for a path from src to trg takes polynomial time. ◀

8 Conclusions

We have looked at the reachability problem for one-counter automata with equality and disequality tests. We have proposed the idea of local inductive invariants and combined them with the notion of unboundedness within an SCC. Our construction circumvents the lack of computationally tractable descriptions: indeed, in the subset sum example (Fig. 1 (left) and Remark 7.8) the reachability set has exponential size, depending on $a_1, \ldots, a_n$. There is no obvious means of compression available, and guessing/storing a traditional invariant is prohibitively expensive even for moderate n.

An outstanding theoretical question is characterisation of complexity of reachability in OCA with disequality tests. We have placed the problem in $\mathsf{coNP}^{\mathsf{NP}}$ and, in the presence of equality tests, in $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$. Both problems have already been known to be NP-hard. Are they NP-complete or coNP-hard too? We also leave it open whether our technique can be extended to other systems and settings, e.g., to parameter synthesis questions (see, e.g., [26, 35, 43]).

In a more practical direction, while the general invariant-based effective procedure for (non-)reachability in vector addition systems [36] has not, to the best of our knowledge, been implemented, our work identifies these potentially practical ways to reduce the search space for invariants in VASS. The idea of restricting invariant sets to just a small "core" (in our case: a union of arithmetic progressions), combined with the compositionality of invariants, can help to direct an exploration of the search space, or assist a learning algorithm.

References

1 Shaull Almagor, Nathann Cohen, Guillermo A. Pérez, Mahsa Shirmohammadi, and James Worrell. Coverability in 1-VASS with disequality tests. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

2 Rajeev Alur and Pavol Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 599–610, 2011.

3 Nicolas Amat, Silvano Dal-Zilio, and Thomas Hujsa. Property directed reachability for generalized Petri nets. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 505–523. Springer, 2022.

4 Pascal Baumann, Moses Ganardi, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Context-bounded verification of context-free specifications. *Proc. ACM Program. Lang.*, 7(POPL):2141–2170, 2023.

5 Michael Blondin, Matthias Englert, Alain Finkel, Stefan Göller, Christoph Haase, Ranko Lazic, Pierre McKenzie, and Patrick Totzke. The reachability problem for two-dimensional vector addition systems with states. *J. ACM*, 68(5):34:1–34:43, 2021.

6 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. Approaching the coverability problem continuously. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9636 of *Lecture Notes in Computer Science*, pages 480–496. Springer, 2016.

7 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. The logical view on continuous Petri nets. *ACM Trans. Comput. Log.*, 18(3):24:1–24:28, 2017.

8 Michael Blondin, Christoph Haase, and Philip Offtermatt. Directed reachability for infinite-state systems. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II*, volume 12652 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2021.

9 Michael Blondin, Tim Leys, Filip Mazowiecki, Philip Offtermatt, and Guillermo A. Pérez. Continuous one-counter automata. *ACM Trans. Comput. Log.*, 24(1):3:1–3:31, 2023.

10 Stanislav Böhm, Stefan Göller, and Petr Jancar. Equivalence of deterministic one-counter automata is NL-complete. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 131–140. ACM, 2013.

11 Benedikt Bollig, Karin Quaas, and Arnaud Sangnier. The complexity of flat freeze LTL. *Log. Methods Comput. Sci.*, 15(3), 2019.

12 Itshak Borosh and Leon Bruce Treybig. Bounds on positive integral solutions of linear Diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, 1976.

13 Aaron R. Bradley. SAT-based model checking without unrolling. In Ranjit Jhala and David A. Schmidt, editors, *Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings*, volume 6538 of *Lecture Notes in Computer Science*, pages 70–87. Springer, 2011.

14 Aaron R. Bradley. Understanding IC3. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2012.

15 Daniel Bundala and Joël Ouaknine. On parametric timed automata and one-counter machines. *Inf. Comput.*, 253:272–303, 2017.

16 Dmitry Chistikov, Wojciech Czerwinski, Piotr Hofman, Michal Pilipczuk, and Michael Wehar. Shortest paths in one-counter systems. *Log. Methods Comput. Sci.*, 15(1), 2019.

17 Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In E. Allen Emerson and A. Prasad Sistla, editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2000.

18 Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977.

19 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. *J. ACM*, 68(1):7:1–7:28, 2021.

20 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021.

21 Stéphane Demri and Arnaud Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In C.-H. Luke Ong, editor, *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6014 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2010.

22 Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. An SMT-based approach to coverability analysis. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014.

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

24 Alain Finkel, Serge Haddad, and Igor Khmelnitsky. Minimal coverability tree construction made complete and efficient. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 237–256. Springer, 2020.

25 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966. `doi:10.2140/pjm.1966.16.285`.

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

27 Stefan Göller and Markus Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013.

28 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009.

29 Michel Hack. *Decidability questions for Petri nets.* PhD thesis, MIT, 1975. URL: `http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-161.pdf`.

30 Matthew Hague and Anthony Widjaja Lin. Model checking recursive programs with numeric data types. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 743–759. Springer, 2011.

31 Matthew Hague and Anthony Widjaja Lin. Synchronisation- and reversal-bounded analysis of multithreaded programs with counters. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2012.

32 Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Software model checking for people who love automata. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2013. `doi:10.1007/978-3-642-39799-8_2`.

33 Alexander Kaiser, Daniel Kroening, and Thomas Wahl. Efficient coverability analysis by proof minimization. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 500–515. Springer, 2012.

34 Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In Jürgen Giesl, editor, *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2005.

35 Antonia Lechner, Richard Mayr, Joël Ouaknine, Amaury Pouly, and James Worrell. Model checking flat freeze LTL on one-counter automata. *Log. Methods Comput. Sci.*, 14(4), 2018.

36 Jérôme Leroux. The general vector addition system reachability problem by Presburger inductive invariants. *Log. Methods Comput. Sci.*, 6(3), 2010.

37 Jérôme Leroux. Distance between mutually reachable Petri net configurations. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

38 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021.

39 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019.

40 Xie Li, Taolue Chen, Zhilin Wu, and Mingji Xia. Computing linear arithmetic representation of reachability relation of one-counter automata. In Jun Pang and Lijun Zhang, editors, *Dependable Software Engineering. Theories, Tools, and Applications - 6th International Symposium, SETTA 2020, Guangzhou, China, November 24-27, 2020, Proceedings*, volume 12153 of *Lecture Notes in Computer Science*, pages 89–107. Springer, 2020.

41 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

42 Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., USA, 1967.

43 Guillermo A. Pérez and Ritam Raha. Revisiting parameter synthesis for one-counter automata. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 33:1–33:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

44 Louis E. Rosier and Hsu-Chun Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *J. Comput. Syst. Sci.*, 32(1):105–135, 1986. `doi:10.1016/0022-0000(86)90006-1`.

45 Sylvain Schmitz. The complexity of reachability in vector addition systems. *SIGLOG News*, 3(1):4–21, 2016.

46 Helmut Seidl, Thomas Schwentick, Anca Muscholl, and Peter Habermehl. Counting in trees for free. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 1136–1149. Springer, 2004.

47 Alistair Stewart, Kousha Etessami, and Mihalis Yannakakis. Upper bounds for Newton's method on monotone polynomial systems, and P-time model checking of probabilistic one-counter automata. *J. ACM*, 62(4):30:1–30:33, 2015.

48 Leslie G. Valiant and Mike Paterson. Deterministic one-counter automata. *J. Comput. Syst. Sci.*, 10(3):340–350, 1975.

A Proof of Lemma 4.3

The proof will use 1-dimensional vector addition systems with states (1-VASS). These are one-counter automata (as defined in this paper) without any tests:

- Syntactically, a 1-VASS is a pair (Q, T), where Q is the set of states and $T \subseteq Q \times \mathbb{Z} \times Q$ is the set of transitions.
- The semantics is the same as that of an OCA with tests $(Q, T, \texttt{true})$, where the map $\texttt{true}$ assigns true to all states in Q.

We will consider an auxiliary problem which takes as input a 1-VASS (Q, T), a state $q \in Q$, and a natural number d (encoded in binary); the problem asks whether there is a positive-effect q-cycle γ of length at most $|Q|$ such that $\mathsf{drop}(\gamma) \leq d$. We show that this problem can be solved in polynomial time, given that coverability in 1-VASS can be decided in polynomial time. Coverability is placed in P by a standard reduction from coverability to unboundedness (see, e.g., [1, Lemma 1]) and a polynomial-time algorithm for unboundedness by Rosier and Yen [44, Theorem 3.4]. A stronger result from [1] is that coverability in 1-VASS is in fact in $\mathsf{NC}^2 \subseteq \mathsf{P}$.

Reduction of the auxiliary problem to coverability. Let $n = |Q|$. We can construct an instance of coverability as follows. Consider the unfolding (Q', T') where $Q' = \{q^{(i)} : q \in Q \text{ and } i \in [0, n]\}$ and $T' = \{(p^{(i-1)}, a, q^{(i)}) : (p, a, q) \in T \text{ and } i \in [1, n]\}$. Observe that there exists a positive-effect q-cycle whose length is at most $|Q|$ and with a drop bounded by d in (Q, T) if and only if $(q^{(i)}, d + 1)$ can be covered from $(q^{(i)}, d)$ in (Q', T') for some $i \in [1, n]$. Moreover, in that case, such a cycle is obtained directly from a path in the unfolded 1-VASS that witnesses coverability.

Polynomial-time algorithms for minimum drop and membership in Conf_+. We complete the proof of Lemma 4.3:

- First, observe that the minimum drop can be computed by a binary search for d. Let $m = \max\{|a| : (p, a, q) \in T\}$. By starting from an upper bound of $n(m + 1)$, d can be computed using a polynomial number (at most $\lceil \log(n(m + 1)) \rceil$) of coverability queries.
- Second, to decide membership of a configuration (q, v) in Conf_+, it suffices to check that $q \in Q_+$, to compute $\mathsf{drop}(\gamma_q)$, and to check that $v \geq \mathsf{drop}(\gamma_q)$. ◀

B Finding Positive-Effect Simple Cycles is NP-hard

▶ **Proposition B.1.** *Deciding, for a given OCA without tests $\mathcal{A}$ and a given state q, whether there exists a positive-effect simple q-cycle in $\mathcal{A}$ is an* NP*-complete problem.*

Proof. Membership in NP is obtained by using the q-cycle itself as a certificate. To prove NP-hardness, we provide a reduction from the Hamiltonian path problem. Let $G = (V, E)$ be a directed graph and let $s, t \in V$ be two distinct vertices. A path from s to t is *Hamiltonian* if it is simple and visits every vertex in the graph. The Hamiltonian path problem takes as input a directed graph $G = (V, E)$ and two vertices $s, t \in V$ and asks whether there is a Hamiltonian path from s to t in G.

For the remainder of this proof, we fix an instance of this problem formed by $G = (V, E)$ and $s, t \in V$. Let $n = |V|$. We will now construct an OCA without tests (a 1-VASS) $\mathcal{A} = (Q, \Delta)$. Define $Q := V \cup \{q\}$, where $q \notin V$ is a new state, and

$$\Delta := \{(u, 1, v) : (u, v) \in E\} \cup \{(q, 0, s), (t, -(n - 2), q)\}.$$

The construction of $\mathcal{A}$ takes polynomial time. We claim that there exists a Hamiltonian path from s to t in G if and only if there exists a positive-effect simple q-cycle in $\mathcal{A}$.

Suppose there exists a Hamiltonian path π from s to t in G. Since π visits every vertex in G, we have $\mathsf{len}(\pi) = n - 1$. Consider the path σ in $\mathcal{A}$ that is obtained from π by replacing each edge $(u, v) \in E$ with the corresponding transition $(u, 1, v) \in \Delta$ as well as prepending the transition $(q, 0, s)$ and appending the transition $(t, -(n - 2), q)$. Given that π is a simple path in G, we know that σ is a simple q-cycle in $\mathcal{A}$. Furthermore, given that $\mathsf{len}(\pi) = n - 1$, we know that $\mathsf{eff}(\sigma) = 0 + n - 1 - (n - 2) = 1$, so σ has positive effect.

Conversely, suppose there exists a positive-effect simple q-cycle σ in $\mathcal{A}$. This σ must begin with $(q, 0, s)$, the only outgoing transition from q, and end with $(t, -(n - 2), q)$, the only transition leading back to q. Let $\sigma = (q, 0, s)\, \sigma'\, (t, -(n - 2), q)$ for some σ'. Given that $\mathsf{eff}(\sigma) \geq 1$ and all other transitions in $\mathcal{A}$ have effect 1, we know that $\mathsf{len}(\sigma') \geq n - 1$. Since the cycle σ is simple and $|Q \setminus \{q, s, t\}| = n - 2$, we conclude that σ' visits each of these $n - 2$ states exactly once. So the path π obtained from σ' by replacing each transition $(u, 1, v) \in \Delta$ with the corresponding edge $(u, v) \in E$ is a Hamiltonian path from s to t in G. ◀

C Discussion of the Choice of Operators

In Condition 7.4, we made the choice of using operator $\mathsf{LPost}^*_-(\cdot)$, and not $\mathsf{Post}^*_-(\cdot) \cup \mathsf{Post}(\mathsf{Post}^*_-(\cdot))$ as in Section 6. Indeed, if we had used $\mathsf{Post}^*_-(\cdot) \cup \mathsf{Post}(\mathsf{Post}^*_-(\cdot))$ instead, then in order to obtain completeness (Lemma 7.11), one would have to change the perfect core. We want the perfect core to be contained in $L \cap \mathit{Conf}_+$ so that it has a short representation; the natural candidate would be to take $\mathsf{Post}^*(\mathsf{src}) \cap L \cap \mathit{Conf}_+$ (and symmetrically in $\mathcal{A}^R$). This perfect core would satisfy the inductive property; however, this choice would break soundness (Lemma 7.10). Indeed, this invariant could contain a locally bounded configuration c that is reached from src using a run that visits many locally unbounded configurations in $\mathit{Conf} \setminus \mathit{Conf}_+$ before coming back to L. In this case, it could be that c is not captured by the inductive property, so one could find an inductive invariant I that does not contain c.