Parametrized Universality Problems for One-Counter Nets

Shaull Almagor
Technion, Israel

Udi Boker
Interdisciplinary Center (IDC) Herzliya, Israel

Piotr Hofman
University of Warsaw, Poland

Patrick Totzke
University of Liverpool, UK

Abstract

We study the language universality problem for One-Counter Nets, also known as 1-dimensional Vector Addition Systems with States (1-VASS), parameterized either with an initial counter value, or with an upper bound on the allowed counter value during runs. The language accepted by an OCN (defined by reaching a final control state) is monotone in both parameters. This yields two natural questions: 1) does there exist an initial counter value that makes the language universal? 2) does there exist a sufficiently high ceiling so that the bounded language is universal?

Despite the fact that unparameterized universality is Ackermann-complete and that these problems seem to reduce to checking basic structural properties of the underlying automaton, we show that in fact both problems are undecidable.

We also look into the complexities of the problems for several decidable subclasses, namely for unambiguous, and deterministic systems, and for those over a single-letter alphabet.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Counter net, VASS, Unambiguous Automata, Universality

Funding *Shaull Almagor*: European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 837327.
Udi Boker: Israel Science Foundation grant 1373/16

Acknowledgements We are grateful for fruitful discussions during the Autoboz'2019 workshop.

1 Introduction

One-Counter Nets (OCNs) are finite-state machines equipped with an integer counter that cannot decrease below zero and which cannot be explicitly tested for zero. They are the same as 1-dimensional Vector Addition Systems (or Petri nets with exactly one unbounded place). In order to use them as formal language acceptors we assume that transitions are labelled with letters from a finite alphabet and that some states are marked as accepting.

OCNs are a syntactic restriction of One-Counter Automata – Minsky Machines with only one counter, which can have zero-tests, i.e., transitions that depend on the counter value being exactly zero. If counter updates are restricted to ± 1, the model corresponds to Pushdown automata with a single-letter stack alphabet. OCNs are one of the simplest types of discrete infinite-state systems, which makes them suitable for exploring the decidability border of classical decision problems from automata and formal-language theory.

Universality Problems. The universality problem for a class of automata asks if a given automaton accepts all words over its input alphabet. Due to their lack of an explicit zero-test, OCNs are monotone with respect to counter values: if it is possible to make an a-labelled step from a configuration with state p and counter n to state q with counter $n + d$, written as $(p, n) \xrightarrow{a} (q, n + d)$ here, then the same holds for any larger counter value $m \geq n$: $(p, m) \xrightarrow{a} (q, m + d)$. Consequently, if we define the language via acceptance by reaching a final control state, then for all states s and $n \leq m \in \mathbb{N}$, the language $\mathcal{L}(s, n)$ of the initial configuration (s, n) is included in that of (s, m). This motivates the our first variation of the universality problem. The *Initial-Value Universality* problem asks if there exists a sufficiently large initial counter to make the resulting language universal.

51 **Input**: An OCN with alphabet Σ and an initial state s_0.
Question: Does there exist $c_0 \in \mathbb{N}$ such that $\mathcal{L}(s_0, c_0) = \Sigma^*$?

The next question we consider is the *Bounded Universality* problem, which asks if there exists a large enough upper bound on the counter so that every word can be accepted via a run that remains within this bound. Writing $\mathcal{L}^{\leq b}(s_0, c_0) \subseteq \Sigma^*$ for the b-bounded language from configuration (s_0, c_0), the decision problem is as follows.

56 **Input**: An OCN with alphabet Σ, an initial state s_0, and $c_0 \in \mathbb{N}$.
Question: Does there exist $b \in \mathbb{N}$ such that $\mathcal{L}^{\leq b}(s_0, c_0) = \Sigma^*$?

The motivation for studying these parameterized problems comes from the observation that the "vanilla" universality problem, without existentially quantifying over parameters, is decidable, but Ackermann-complete [16], and the lower bound depends strongly on the assumption that we start with a fixed initial counter (and that its value is not bounded). The two new variants of the universality problem relax these assumptions in an attempt to allow efficient decision procedures via simple cycle analysis or similar.

Our Results. We show that both initial-value universality and bounded universality are undecidable (Section 3). The proofs use techniques from weighted automata [13, 5], reducing the halting problem of two-counter machines to our setting. In a nutshell, the idea (for e.g., initial-value universality) is to construct, given a two-counter machine $\mathcal{M}$, an OCN that reads encodings of runs of $\mathcal{M}$. Then, the OCN checks whether the encoding indeed represents a prefix of the run of $\mathcal{M}$. If it does not, the word is accepted. Otherwise, if the prefix becomes too long (depending on the initial value), the counter of the OCN becomes negative, and the word is not accepted. Thus, if $\mathcal{M}$ halts, then there exists a large enough initial value (namely larger than the length of $\mathcal{M}$'s run) for which every word is accepted, and if $\mathcal{M}$ does not halt, then a long enough prefix of its run will be rejected.

In light of these negative results, we proceed to study restricted classes of OCNs, for which the problems become decidable, as we elaborate below. In most cases, the complexity crucially depends on how transition updates are encoded: we consider both the case of "succinct", binary-encoded updates, and the case of unary-encoded updates, which corresponds to systems where transitions can only update the counter by ± 1.

The most intricate and interesting case is that of OCNs over a single-letter alphabet (Section 4). In order to analyze this model, we split universality to criteria on "short" words, and on longer words that admit a cyclic behavior. In particular, we devise a canonical representation of "pumpable" paths, akin to the so-called linear-path schemes [19, 7]. We show that the complexity of some of the problems is coNP complete, where others range between coNP and coNP^{NP} (see Tables 1 and 2).

We then consider deterministic, and unambiguous OCNs (Sections 5 and 6, respectively). For such systems, deciding (bounded) universality problems mostly reduces to checking

Table 1 The complexity of the universality problems of one-counter nets in which weights are encoded in unary.

Unary encoding	Universality		Initial-Value Universality		Bounded Universality	
	Singleton Alphabet	General Alphabet	Singleton Alphabet	General Alphabet	Singleton Alphabet	General Alphabet
Deterministic	L Theorem 28	NL-comp. Theorem 26	L Theorem 28	NL-comp. Theorem 26	L Theorem 28	NL-comp. Theorem 26
Unambiguous	NL Theorem 31	NC^2; NL-hard [12]	NL Theorem 34	NC^2 Theorem 34	NL Theorem 36	NC^2 Theorem 36
Non-deterministic	coNP-comp. Theorem 10	Ackermann [16]	coNP-comp. Theorem 15	Undecidable Theorem 1	coNP-comp. Theorem 22	Undecidable Theorem 2

Table 2 The complexity of the bounded universality problems of one-counter nets in which weights are encoded in binary.

Binary encoding	Universality		Initial-Value Universality		Bounded Universality	
	Singleton Alphabet	General Alphabet	Singleton Alphabet	General Alphabet	Singleton Alphabet	General Alphabet
Deterministic	NC^2 Theorem 28	NC Theorem 26	NC^2 Theorem 28	NC^2 Theorem 34	NC^2 Theorem 28	NC Theorem 26
Unambiguous	coNP-comp. Theorem 12	PSPACE; coNP-hard [12]	NC^2 Theorem 34	NC^2 Theorem 34	$coNP^{NP}$ Theorem 22	PSPACE Theorem 36
Non-deterministic	$coNP^{NP}$ Theorem 12	Ackermann [16]	coNP-comp. Theorem 15	Undecidable Theorem 1	$coNP^{NP}$ Theorem 22	Undecidable Theorem 2

simple conditions on the cyclic structure of the control automaton underlying the OCN. Based on known (but in some cases very recent) results on unambiguous finite automata and vector-addition systems, we derive relatively low complexity upper bounds, in polynomial time (assuming unary encoding) and space (assuming binary encoding). Tables 1 and 2 summarize the status quo, following our results.

Related work. The undecidability of language universality for pushdown automata is textbook. In his 1973 PhD thesis [25], Valiant showed that the problem remains undecidable for the strictly weaker model of one-counter automata (OCA, with zero tests) by recognizing the complement of all accepting runs of a two-counter machine. Language inclusion is undecidable for the further restricted model of OCNs [15]. If one considers ω-regular languages defined by OCNs with Büchi acceptance condition then the resulting universality problem is undecidable [8].

On the positive side, universality is decidable for vector addition systems [17] and Ackermann-complete for the special case of OCNs [16]. One-counter systems have received some attention in regards to checking bisimulation and simulation relations, which under-approximate language equivalence (and inclusion, respectively) and are computationally simpler. For OCAs/OCNs, bisimulation is PSPACE-complete [9], while weak bisimulation is undecidable for OCNs [20]. Both strong and weak simulation are PSPACE-complete for OCNs, and checking if an OCN simulates an OCA is decidable [1].

Universality problems for OCNs over single-letter alphabets are related to the termination problem for VASS, which asks if there exists an infinite run. Non-termination naturally corresponds to the property that $a^n \in \mathcal{L}(s_0, \mathbf{v_0})$, i.e., all finite words are accepted, assuming that all states are accepting. Termination reduces to boundedness (finiteness of the reachability set) which is EXPSPACE-complete [22, 14] in general and PSPACE-complete for systems with fixed dimensions [23]. In contrast, the *structural* termination problem (there exists no infinite run, regardless of the initial configuration) is equivalent to finding an executable cycle that is non-decreasing on all dimensions, and can be solved in polynomial time [18]. Finally, the idea to existentially quantify over some initial resource is commonplace in the formal verification literature. Examples include unknown initial-credit problems for energy games [10, 1] and R-Automata [3], timed Petri nets [2], and inclusion problems for weighted automata [13, 5].

Due to space constraints, most proofs appear in the Appendix.

2 Preliminaries

One-Counter Nets. A *one-counter net* (OCN) is a finite directed graph where edges carry both an integer weight and a letter from a finite alphabet. We write $\mathcal{A} = (\Sigma, Q, s_0, \delta, F)$ for the net $\mathcal{A}$ where Q is a finite set of *states*, Σ is a finite set of *letters*, $s_0 \in Q$ is an *initial state*, $\delta \subseteq Q \times \Sigma \times \mathbb{Z} \times Q$ is the *transition* relation, and $F \subseteq Q$ are the *accepting* states.

For a transition $t = (s, a, e, s') \in \delta$ we write $\mathit{effect}(t) \stackrel{def}{=} e$ for its (counter) *effect*, and write $\|\delta\|$ for the largest absolute effect among all transitions. By the *underlying automaton* of an OCN we mean the NFA obtained from the OCN by disregarding the transition effects.

A path in the OCN is a sequence $\pi = (s_1, a_1, e_1, s_2)(s_2, a_2, e_2, s_3) \dots (s_k, a_k, e_k, s_{k+1}) \in \delta^*$. Such a path π is a *cycle* if $s_1 = s_{k+1}$, and is a *simple cycle* if no other cycle is a proper infix of it. We say that the path above *reads* word $a_1 a_2 \dots a_k \in \Sigma^*$ and is accepting if $s_{k+1} \in F$. Its $\mathit{effect}(\pi) \stackrel{def}{=} \sum_{i=1}^{k} e_i$ is the sum of its transition effects . Its *height* is the maximal effect of any prefix and, similarly, its *depth* is the inverse of the minimal effect of any prefix.

An OCN naturally induces an infinite-state labelled transition system in which each *configuration* is a pair $(s, c) \in Q \times \mathbb{N}$ comprising a state and a non-negative integer. We call such a configuration *final*, or *accepting*, if $s \in F$. Every letter $a \in \Sigma$ induces a step relation $\xrightarrow{a} \subseteq (Q \times \mathbb{N})^2$ between configurations where, for every two configurations (s, c) and (s', c'),

$$(s, c) \xrightarrow{a} (s', c') \iff (s, a, d, s') \in \delta \quad \text{and } c' = c + d.$$

A *run* on a word $w = a_1 a_2 \dots a_k \in \Sigma^*$ is a path in this induced infinite system; that is, a sequence $\rho = (s_0, c_0), (s_1, c_1), (s_2, c_2), \dots (s_k, c_k)$ such that $(s_{i-1}, c_{i-1}) \xrightarrow{a_i} (s_i, c_i)$ holds for all $1 \le i \le k$. Naturally, a run uniquely describes a path in the underlying finite OCN. Conversely, for every such path and initial counter value c_0, there is at most one corresponding run: A path π is *executable from* c_0 if its depth is at most c_0 (that is, we do not allow the counter to become negative). A run as above is called a (simple) *cycle* if its underlying path is a (simple) cycle. It is *accepting* if it ends in an accepting configuration. We call a run *bounded* by $b \in \mathbb{N}$ if $c_i \le b$ for all $0 \le i \le k$.

For any fixed initial configuration (s, c), we define its *language* $\mathcal{L}_{\mathcal{A}}(s, c) \subseteq \Sigma^*$ to contain exactly all words on which an accepting run starting in (s, c) exists. (We omit the subscript $\mathcal{A}$ if the OCN is clear from context.) Similarly, the *b-bounded language* $\mathcal{L}^{\le b}(s, c)$ is the set of those words on which there is a b-bounded run starting in (s, c).

The OCN is *deterministic* if for every pair $(s, a) \in Q \times \Sigma$ there is at most one pair $(d, q) \in \mathbb{N} \times Q$ with $(s, a, d, s') \in \delta$. A net as above together with an initial configuration (s_0, c_0) is *unambiguous* if for every word $w \in \Sigma^*$ there is at most one accepting run starting in (s_0, c_0).

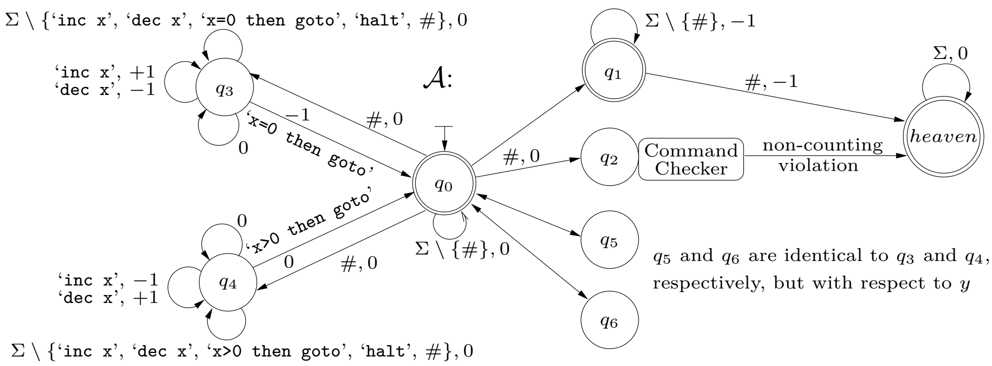

Figure 1 The one-counter net $\mathcal{A}$ from the proof of Theorem 1.

Two-Counter Machines. A two-counter machine (Minsky Machine) $\mathcal{M}$ is a sequence $(l_1, \dots, l_n)$ of commands involving two counters x and y. We refer to $\{1, \dots, n\}$ as the *locations* of the machine. There are five possible forms of commands: `inc(c)`, `dec(c)`, `goto` l_i, `halt`, `if c=0 goto` l_i `else goto` l_j, where $c \in \{x, y\}$ is a counter and $1 \le i, j \le n$ are locations. The counters are initially set to 0. Since we can always check whether $c = 0$ before a `dec(c)` command, we assume that the machine never reaches `dec(c)` with $c = 0$. That is, the counters never have negative values.

3 Undecidability

We show that both initial-value universality and bounded universality are undecidable by reduction from the halting problem of two-counter machines, which is known to be undecidable [21]. The proofs use techniques similar to those used in [13] and [5].

The idea underlying both undecidability results is that the initial counter value, or the bound on the allowed counter, prescribes a bound on the number of steps until the OCN must make a decision weather the input word, which encodes a run of the two-counter machine, either halts or cheats. After this decision the OCN is reset and continues to read the remaining word within an adjusted bound. If the decision was correct then the bound remains the same and otherwise, it is strictly reduced. The existence of a halting run of the two-counter machine now implies that its length corresponds to a sufficient initial bound for this simulating OCN to be universal. In particular, if the run of the machine does not halt then for every bound – after which the OCN must declare termination or cheat – there exists a non-cheating, and non-terminating prefix run. Repeating this prefix n times witnesses non-universality for the simulating OCN.

3.1 Initial-Value Universality

Given a two-counter machine $\mathcal{M}$, we construct a one-counter net $\mathcal{A}$ as follows (see Figure 1). Intuitively, an input word w to $\mathcal{A}$ is a sequence of segments separated by #, where each segment is a sequence of commands from $\mathcal{M}$. Accordingly, the alphabet of $\mathcal{A}$ consists of # and all possible commands of $\mathcal{M}$.

We build $\mathcal{A}$ to accept w, once starting with a big enough initial counter value, if one of the following conditions hold: i) one of w's segments is shorter than the length of the (legal halting) run of $\mathcal{M}$; or ii) one of w's segments does not respect the control structure underlying $\mathcal{M}$, which is called a "non-counting cheat" here; or iii) all of w's segments do not describe a prefix of the run of $\mathcal{M}$, making "counting cheats". The OCN reads every segment in between two #'s starting in, and returning to, a central state q_0.

Non-counting cheats are easy to verify—for every line l of $\mathcal{M}$, there is a corresponding state q in $\mathcal{A}$, and when $\mathcal{A}$ is at state q and reads a letter a, $\mathcal{A}$ checks if a matches the command in l. For example, if $l =$'`goto i`' and $a =$ '`inc x`', the transition from q goes to a forever accepting state (*heaven*), and if $a =$'`goto i`', it goes to the state of $\mathcal{A}$ that corresponds to the line l_i. This is the "command-checker gadget" of $\mathcal{A}$.

Counting cheats are more challenging to verify, as OCNs cannot branch according to a counter value. We consider separately "positive cheats" and "negative cheats". The former stands for the case that the input letter is '`x=0 then goto`' (or '`y=0 then goto`') while the value of x (or y) in the legal run of $\mathcal{M}$ should be positive. The latter stands for the case that the input letter is '`x>0 then goto`' (or '`y>0 then goto`') while the value of x (or y) in the legal run of $\mathcal{M}$ should be 0.

Positive cheats can be verified by directly simulating the respective counter of $\mathcal{M}$ using the counter in $\mathcal{A}$ (states q_3 and q_5 in Figure 1). Once the cheat occurs, $\mathcal{A}$ can return to q_0 with a penalty of -1, and since the counter in $\mathcal{M}$ is positive, we are guaranteed that the counter in $\mathcal{A}$ did not decrease since leaving q_0, allowing $\mathcal{A}$ to continue the run.

For verifying a negative cheat, we simulate the counting of $\mathcal{M}$ by an "opposite-counting" in $\mathcal{A}$ (states q_4 and q_6 in Figure 1), whereby an increment of the counter in $\mathcal{M}$ results in a decrement of the counter in $\mathcal{A}$, and vice versa—once the cheat occurs, $\mathcal{A}$ can return to q_0 with no penalty, and since the counter in $\mathcal{M}$ is 0, we are guaranteed that the counter in $\mathcal{A}$ did not decrease since leaving q_0, allowing $\mathcal{A}$ to continue the run.

Formally, we construct $\mathcal{A}$ from $\mathcal{M}$ as follows.

- The alphabet Σ of $\mathcal{A}$ consists of # and the descriptive commands for the counter machine $\mathcal{M}$: '`inc x`', '`inc y`', '`dec x`', '`dec y`', '`halt`', and for every line i of $\mathcal{M}$, the commands '`goto i`', '`x=0 then goto i`', '`y=0 then goto i`', '`x>0 then goto i`', and '`y>0 then goto i`'.
- The initial state q_0 is accepting, it has a self transition over $\Sigma \setminus \{\#\}$ and nondeterministic transitions to the states $q_1 \dots q_6$ over #, all with weight 0.
- There is a *heaven* state, which is accepting, and has a self loop over Σ with weight 0.
- The state q_1 is accepting and intuitively allows to accept short segments between consequent #'s: It has a self transition over $\Sigma \setminus \{\#\}$ and a transition to *heaven* over #, all with weight -1.
- The state q_2 starts the command-checker gadget, which looks for a non-counting violation of $\mathcal{M}$'s commands (which is a simple regular check). Once reaching a violation it goes to *heaven*. All of its transitions are with weight 0. If it does not find a violation, it cannot continue the run.
- The state q_3 is a positive-cheat checker for $\mathcal{M}$'s counter x. It has a self loop over '`inc x`' with weight $+1$ and over '`dec x`' with weight -1. Over '`x=0 then goto`' it can nondeterministically choose between a self loop with weight 0 and a transition to q_0 with weight -1. Over the rest of the alphabet lettres, except for '`halt`' and #, it has a self loop with weight 0. (Over '`halt`' and # it cannot continue the run.)
- The state q_4 is a negative-cheat checker for $\mathcal{M}$'s counter x. It has a self loop over '`inc x`' with weight -1 and over '`dec x`' with weight $+1$. Over '`x>0 then goto`' it can nondeterministically choose between a self loop with weight 0 and a transition to q_0 with weight 0. Over the rest of the alphabet lettres, except for '`halt`' and #, it has a self loop with weight 0.

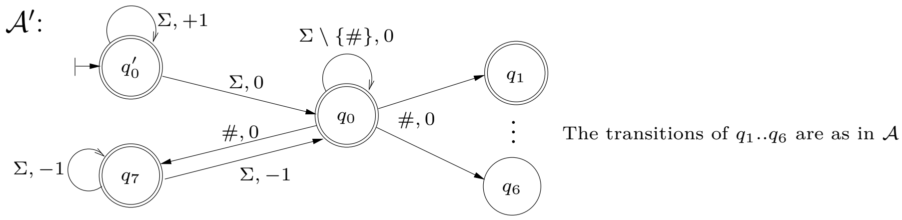

Figure 2 The one-counter net $\mathcal{A}'$ from the proof of Theorem 2.

- The states q_5 and q_6 provide positive-cheat checker and negative-cheat checker for $\mathcal{M}$'s counter y, respectively, analogously to states q_3 and q_4.

We can show (see Appendix A) that $\mathcal{M}$ halts if and only if $\mathcal{A}$ is initial-value universal.

▶ **Theorem 1.** *The initial-value universality problem for one-counter nets is undecidable.*

3.2 Bounded Universality

We show that the problem is undecidable by making some changes to the undecidability proof of the initial-value universality problem.

Given a two-counter machine $\mathcal{M}$, we construct a one-counter net $\mathcal{A}'$ that is similar to $\mathcal{A}$, as constructed above, except for the following changes (see Figure 2):

- There is an additional state q'_0 that is accepting, it is the new initial state, and it has a nondeterministic choice over Σ of either taking a self loop with weight $+1$ or going to q_0 with weight 0.
- The state q_0 is no longer initial, and it has an additional transition over # to a new state q_7 with weight 0.
- The state q_7 is accepting, and it has nondeterministic choice over Σ of either taking a self loop with weight -1 or going to q_0 with weight -1.

Now $\mathcal{M}$ halts if and only if $\mathcal{A}'$ is bounded universal for an initial counter value 0.

▶ **Theorem 2.** *The bounded universality problem for one-counter nets is undecidable.*

4 Singleton Alphabet

In this section we study universality problems on OCN over singleton alphabets. The universality problem for NFA over singleton alphabets is already coNP-hard [24], a lower bound which trivially carries over to all problems considered here[1].

For simplicity, we identify languages $L \subseteq \{a\}^*$ with their Parikh image, so that the universality problems ask if the (bounded) language of a given OCN equals $\mathbb{N}$. Throughout this section, fix an OCN $\mathcal{A} = (\Sigma, Q, s_0, \delta, F)$.

We start by sketching our approach. Observe that the language of an OCN is not universal iff the OCN does not accept some word w. To show that such w exists, we distinguish between two cases: either w is "relatively short", in which case we use a guess-and-check

---

[1] The proof in [24, Theorem 6.1] in fact shows NP-completeness of the problem of whether two regular expressions over $\{0\}$ define different languages. Hardness is shown by reduction from Boolean satisfiability to non-universality of expressions using prime-cycles, and it is straightforward to rephrase it in terms of DFAs.

approach to find it, or it is long, in which case we deduce its existence by analyzing some cyclic behaviour of the OCN. The details of both the guess-and-check elements and the cyclic behaviour depend on the encoding of the weights and the variant of universality.

4.1 Universality

We start by describing a procedure to decide the ordinary universality problem for OCN over singleton alphabets – with fixed initial configuration and no bounds on the counter.

Consider a cycle $\gamma = s_1, s_2, \ldots, s_k$ (with $s_1 = s_k$). Recall that *effect*(γ) is the sum of weights along γ and *depth*(γ) is the inverse of the lowest effect along the prefixes of γ. We call $1 \leq d \leq k$ a *nadir* of γ if it is the index of a prefix that attains the depth of γ. That is, *effect*$(s_1, \ldots, s_d) = -$*depth*(γ). We say that γ is *positive* if *effect*(γ) is positive (and similarly for negative, non-negative, zero, etc.). We call γ *good* if it a simple, non-negative cycle, and *depth*$(\gamma) = 0$.

▶ **Observation 3.** *If γ is non-negative and it has a nadir d, then the* shifted cycle $\gamma^{\leftarrow d} \stackrel{def}{=} s_d\, s_{d+1}, \cdots, s_k, s_2, \cdots, s_d$ *is good. Similarly, if γ is negative, then effect$(\gamma^{\leftarrow d}) = -$depth$(\gamma^{\leftarrow d})$.*

For a state $r \in Q$ and an initial configuration s_0, c_0, let $\mathcal{L}^r(s_0, c_0) \subseteq \mathcal{L}(s_0, c_0)$ be the language of words accepted by a run that visits r.

The first tool we use in studying the universality problem is a canonical form for accepting runs, akin to *linear path schemes* of [19, 7].

▶ **Definition 4** (Linear Forms). *A path π is in* linear form *if there exist simple cycles $\gamma_1, \ldots, \gamma_k$ and paths $\tau_0, \ldots, \tau_k$ such that $\pi = \tau_0\gamma_1^{e_1}\tau_1 \cdots \tau_{k-1}\gamma_k^{e_k}\tau_k$ for some numbers $e_1, \ldots, e_k \in \mathbb{N}$, and such that every non-negative cycle γ_i, is taken from a nadir, and so is executable with any counter value.*

We call e_i the exponent *of γ_i, and we refer to $\tau_0\gamma_1\tau_1 \ldots \gamma_k\tau_k$ as the* underlying path *of π. The* length *of the linear form is the length of the underlying path.*

A linear form is described by the components above, where the exponents are given in binary. In the following, we show that every path can be transformed to a path in linear form with a small description size.

▶ **Lemma 5.** *Let π be an executable path of length n from (p, c) to (q, c'). Then there exists an executable path π' of length n in linear form whose length is at most $2|Q|^2$, from (p, c) to (q, c'') with $c'' \geq c'$.*

Proof Sketch: π' is obtained from π in two steps, namely rearranging simple cycles, and then choosing a small set of "representative" simple cycles to replace others. The crux of the proof is the first step, where instead of simply moving a cycle, we also shift it so that it is taken from its nadir. Then, for every set of simple cycles of the same length and on the same state, we take the one with maximal effect as a representative. ◀

We now turn to identify states that have a special significance in analyzing universality.

▶ **Definition 6.** *Let* Pump $\subseteq Q$ *be the set of states that admit good cycles. For each such state r fix a shortest good cycle γ_r.*

Intuitively, a state r is in Pump if it has a cycle that can be taken with any counter value, any number of times. That is, it can be used to "pump" the length of the word. Another important property is that if a path never visits a state in Pump then *all* its simple cycles must be negative. Indeed, any non-negative cycle must contain a non-negative simple cycle and any state at a nadir of such cycle must be in Pump.

If however, a state in Pump occurs along an accepting run, we can accept the same word using a run in a short linear form, as we now show.

▶ **Lemma 7.** *There exists a bound $B_1 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that, for every $n \in \mathbb{N}$, if n is accepted by a run that visits a state $r \in \text{Pump}$, then n has an accepting run of the form $\eta_1\gamma_r^t\eta_2$ for paths η_1, η_2 of length at most B_1.*

Proof Sketch: Using Lemma 5, we split an accepting run on n that visits r to the form π_1, r, π_2 where π_1 and π_2 are in linear form. Then, we successively shorten π_1 and π_2 by eliminating simple cycles along them, and instead pumping the non-negative cycle γ_r. Some careful accounting is needed so that the length of the path is maintained, and so that it remains executable. ◀

We now characterize the regular language $\mathcal{L}^r(s_0, c_0)$ using a DFA of bounded size.

▶ **Lemma 8.** *There exists a bound $B_2 \in \mathsf{poly}(\|\delta\| \cdot |Q|)$ such that, for every $r \in \text{Pump}$, there exists a DFA that accepts $\mathcal{L}^r(s_0, c_0)$ and is of size at most B_2.*

Define $\mathcal{P} \stackrel{def}{=} \bigcup_{r \in \text{Pump}} \mathcal{L}^r(s_0, c_0)$. Notice that $\mathcal{P} \subseteq \mathcal{L}(s_0, c_0)$ and that $\mathcal{L}(s_0, c_0) \setminus \mathcal{P}$ must be finite. Indeed, if $w \in \mathcal{L}(s_0, c_0) \setminus \mathcal{P}$ then it can only be accepted by runs with *only* negative cycles, of which there are finitely many. In particular, if $\mathbb{N} \setminus \mathcal{P}$ is infinite, then $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$.

Using the bounds from Lemma 8, we have the following.

▶ **Lemma 9.** *There exists $B_3 \in \mathsf{poly}(\|\delta\|, |Q|)$ such that $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$ if, and only if, there exists $n \in \mathbb{N}$ such that either $n < B_2$ and $n \notin \mathcal{L}(s_0, c_0)$, or $B_3^{|Q|} \leq n \leq 2B_3^{|Q|}$ and $n \notin \mathcal{P}$.*

Lemma 9 suggests the following algorithmic scheme for deciding non-universality: non-deterministically either (1) guess $n < B_3$, and check that $n \notin \mathcal{L}(s_0, c_0)$, or (2) guess $B_3^{|Q|} \leq n \leq 2B_3^{|Q|}$ and check that $n \notin \mathcal{L}^r(s_0, c_0)$ for all $r \in \text{Pump}$, which implies that $n \notin \mathcal{P}$.

Note that even if the transitions are encoded in unary, n still needs to be guessed in binary for part (2) (and also for part (1) if the encoding is binary). The complexity of the checks involved in both parts of the algorithm depend on the encoding of the transitions, and are handled separately in the following.

Unary Encoding. If the transitions are encoded in unary, then B_3 is polynomial in the size of the OCN. Consequently, we can check for $n < B_3$ whether $n \in \mathcal{L}(s_0, c_0)$ by simulating the OCN for n steps, while keeping track of the maximal run to each state. Indeed, due to the monotonicity of executability of OCN paths it suffices to remember, for each state s, the maximal possible counter-value c so that (s, c) is reachable via the current prefix, which must be a number $\leq c_0 + n \cdot \|\delta\|$ or $-\infty$ (to represent that no configuration (s, c) can be reached).

Next, in order to check whether $n \notin \mathcal{L}^r(s_0, c_0)$ for all $r \in \text{Pump}$ for $B_3^{|Q|} \leq n \leq 2B_3^{|Q|}$ written in binary, we notice that since B_3 is polynomial in the description of the OCN, then the size of each DFA for $\mathcal{L}^r(s_0, c_0)$ constructed as per Lemma 8 is polynomial in the OCN. Since the proof in Lemma 8 is constructive, we can obtain an explicit representation of these DFAs. Finally, given a DFA (or indeed, and NFA) over a singleton alphabet and n written in binary, we can check whether n is accepted in time $O(\log n)$ by repeated squaring of the transition matrix for the DFA [24]. We conclude with the following.

▶ **Theorem 10.** *The universality problem for singleton-alphabet one-counter nets with transitions encoded in unary is in* coNP*, and is thus* coNP*-complete.*

Binary Encoding. When the transitions are encoded in binary, B_3 is potentially exponential in the encoding of the OCN. Thus, naively adapting the methods taken in the unary case (with basic optimization) will lead to a PSPACE algorithm for universality (using Savitch's Theorem). As we now show, by taking a different approach, we can obtain an upper bound of $\mathsf{coNP}^{\mathsf{NP}}$, placing the problem in the second level of the polynomial hierarchy.

In order to obtain this bound, we essentially show that given n encoded in binary, checking whether n is accepted by the OCN can be done in NP. This is based on the linear form of Lemma 5.

▶ **Lemma 11.** *Let $\pi = \tau_0\gamma_1^{e_1}\tau_1 \cdots \tau_{k-1}\gamma_k^{e_k}\tau_k$ be a run in linear form, then we can check whether π is executable from counter value c in time polynomial in the description of π.*

Lemma 11 shows that, given n in binary, we can check whether $n \in \mathcal{L}(s_0, c_0)$ in NP. Indeed, we guess the structure of an accepting run in linear form (including the exponents of the cycles), and check in polynomial time whether this run is executable, and whether it is accepting.

In order to complete our algorithmic scheme for universality, it remains to show how we can check in NP, given n in binary, whether $n \notin \mathcal{L}^r(s_0, c_0)$ for every r. In contrast to the case of unary encoding, this is fairly simple.

Given r, we can construct an OCN $\mathcal{A}^r$ such that $\mathcal{L}_{\mathcal{A}^r}(s_0, c_0) = \mathcal{L}^r_{\mathcal{A}}(s_0, c_0)$ by taking two copies of $\mathcal{A}$, and allowing a transition to the second copy only once r is reached. The accepting states are then those of the second copy. Thus, checking whether $n \notin \mathcal{L}^r(s_0, c_0)$ amounts to checking whether $n \notin \mathcal{L}_{\mathcal{A}^r}(s_0, c_0)$. We can now complete the algorithmic scheme.

▶ **Theorem 12.** *The universality problem for singleton-alphabet one-counter nets with transitions encoded in binary is in* $\mathsf{coNP}^{\mathsf{NP}}$.

4.2 Initial-Value Universality

The characterization of universality given in Lemma 9 can be simplified in the case of initial-value universality, in the sense that the freedom in choosing an initial value allows us to work with the underlying automaton of the OCN, disregarding the transition effects. This also allows us to obtain the same complexity results under unary and binary encodings.

Recall that Pump is the set of states that admit good cycles (see Definition 6). Let $\mathcal{N}$ be the underlying NFA of $\mathcal{A}$. For a state $r \in \text{Pump}$, define $\mathcal{L}^r_{\mathcal{N}}(s_0)$ to be the set of words accepted by $\mathcal{N}$ via a run that visits r. Overloading the notation of Section 4.1, we define $\mathcal{P} \stackrel{def}{=} \bigcup_{r\in\text{Pump}} \mathcal{L}^r_{\mathcal{N}}(s_0)$.

▶ **Lemma 13.** *There exists c_0 such that $\mathcal{L}_{\mathcal{A}}(s_0, c_0) = \mathbb{N}$ iff $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$ and $\mathbb{N} \setminus \mathcal{P}$ is finite.*

Following similar arguments to those in Lemmas 7 and 8, and using the fact that we work with the underlying NFA, we can show the following.

▶ **Lemma 14.** *There exists a bound $B_4 \in \mathsf{poly}(|Q|)$ such that, for every $r \in$ Pump there exists a DFA that accepts $\mathcal{L}^r(s_0)$ and which is of size at most B_4.*

We can now solve the initial-value universality problem.

▶ **Theorem 15.** *The initial-value universality problem for one-counter nets (in unary or binary encoding) is* coNP*-complete.*

Proof. First, observe that the problem is coNP-hard by reduction from the universality problem for NFAs. We now turn to show the upper bound.

By Lemma 13, it is enough to decide whether $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$ and $\mathbb{N} \setminus \mathcal{P}$ is finite. Checking whether $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$, i.e., deciding the universality problem for NFA over a single-letter alphabet, can be done in coNP [24].

By Lemma 14, there exists a DFA $\mathcal{D}$ for $\mathbb{N} \setminus \mathcal{P}$ of size at most $M = B_4^{|Q|}$, by taking the intersection of the respective DFAs over every $r \in \text{Pump}$. Thus, $\mathbb{N} \setminus \mathcal{P}$ is infinite iff $\mathcal{D}$ accepts a word of length $M < n \leq 2M$ (as such a word induces infinitely many other words). Thus, we can decide in NP whether $\mathbb{N} \setminus \mathcal{P}$ is infinite, by guessing $M < n \leq 2M$, and checking that it is in $\mathcal{L}^r(s_0)$ for every $r \in \text{Pump}$ (using repeated squaring on the respective DFAs).

We conclude that both checking whether $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$ and whether $\mathbb{N} \setminus \mathcal{P}$ is finite can be done in coNP, and so the initial value universality problem is also in coNP. ◀

4.3 Bounded Universality

For bounded universality, the states in Pump are not restrictive enough: in order to keep the counter bounded, a state must admit a 0-effect cycle. However, these cycles need not be simple. Thus, we need to adjust our definitions somewhat. Fortunately, however, once the correct definitions are in place, most of the proofs carry out similarly to those of Section 4.1.

▶ **Definition 16.** *A state $q \in Q$ is* stable *if either:*
1. *it is at the nadir of a simple positive cycle, and admits a negative cycle, or*
2. *it is at the nadir of a simple zero cycle.*

We denote by Stable *the set of stable states.*

Identifying stable states can be done in polynomial time (see e.g. Lemma 24). The motivation behind this definition is to identify states that admit a zero-effect (not necessarily simple) cycle.

▶ **Lemma 17.** *There exists a bound $B_5 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that, every stable state q admits a zero cycle of length and depth at most B_5.*

By Lemma 17 we can fix, for each $q \in \text{Stable}$, some zero-cycle ζ_q with effect and depth bounded by B_5. Recall that $\mathcal{L}^r(s_0, c_0)$ is the set of words that are accepted with a path that passes through r. Let $\mathcal{S} \stackrel{def}{=} \bigcup_{r \in \text{Stable}} \mathcal{L}^r(s_0, c_0)$. We prove an analogue of Lemma 7.

▶ **Lemma 18.** *There exists a bound $B_6 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that every $n \in \mathcal{L}^r(s_0, c_0)$ has an accepting run of the form $\eta_1 \zeta_r^t \eta_2$ for paths η_1, η_2 of length at most B_6.*

Proof. The proof follows *mutatis-mutandis* that of Lemma 7, with one important difference: before replacing cycles with iterations of the zero cycle ζ_r, we replace a bounded number of cycles with the positive cycle on r, on which r is at a nadir,[2] so that the counter value goes above *depth*(ζ_r), enabling us to take ζ_r arbitrarily many times. Note that this lengthens the prefix η_1 at most polynomially in $(|Q| \cdot \|\delta\|)$. ◀

Lemma 18 implies that every word $n \in \mathcal{S}$ can be accepted by a run whose counter values are bounded because there must by an accepting run that, except for some bounded prefix and suffix, only iterates some zero-cycle ζ_r. More precisely, we have the following.

[2] That is, unless r is the nadir of a zero cycle, in which case the proof requires no changes.

▶ **Theorem 19.** *There exists $B_6 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that every word $n \in \mathcal{S}$ is accepted by a run whose counter value remains below $2B_6 + c_0$.*

In addition, Lemma 18 immediately gives us (with an identical proof) an analogue of Lemma 8.

▶ **Lemma 20.** *There exists a bound $B_7 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that, for every $r \in \text{Stable}$ there exists a DFA that accepts $\mathcal{L}^r(s_0, c_0)$ and is of size at most B_7.*

We can now characterize bounded universality in terms of $\mathcal{S}$, the set of stable states.

▶ **Lemma 21.** *$\mathcal{L}(s_0, c_0)$ is bounded-universal if, and only if, the underlying automaton $\mathcal{N}$ is universal ($\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$) and $\mathbb{N} \setminus \mathcal{S}$ is finite.*

Finally, checking whether $\mathbb{N} \setminus \mathcal{S}$ is finite can be done similarly to Section 4.1 (and the complexity depends on the transition encoding), by checking that a candidate word n of bounded length is not in $\mathcal{L}^r(s_0, c_0)$ for all stable states r. We conclude with the following.

▶ **Theorem 22.** *Bounded universality of one-counter nets is* coNP*-complete assuming unary encoding, and in* $\mathsf{coNP}^{\mathsf{NP}}$ *assuming binary encoding.*

5 Deterministic Systems

We turn to deterministic one-counter nets (DOCNs) for which the underlying finite automaton is a DFA. We assume without loss of generality that the graphs underlying the DOCNs are connected, i.e., that all states are reachable from the initial state.

For such systems, (bounded) universality problems can be decided by checking a suitable combination of simple conditions on cycles and short words. In order to prevent tedious repetition, we list these conditions first and prove (in Appendix C) upper bounds for checking each of them (Lemma 24). We then show which combination allows to solve each decision problem (Lemma 25).

All mentioned upper bounds follow either easily from first principles, or from the result that the state reachability problem (a.k.a., coverability) for OCN is in NC [6, Theorem 15]. We will also use the following fact, which follows from [26] (see C).

▶ **Lemma 23.** *Given a set $S = \{\alpha_1, \alpha_2 \ldots \alpha_n\}$ of integers written in binary, the question whether the sum of all elements in S is non-negative is in* NC^2.

▶ **Lemma 24** (Basic Conditions)**.** *Consider the following conditions on a deterministic one-counter net $\mathcal{A} = (\Sigma, Q, s_0, \delta, F)$, initial value $c_0 \in \mathbb{N}$, and bound $b \in \mathbb{N}$.*

(C1) *The underlying automaton is universal.*

(C2) *Every word w of length $|w| \leq |Q|$ is in $\mathcal{L}(s_0, c_0)$*

(C3) *Every word w of length $|w| \leq |Q|$ is in $\mathcal{L}^{\leq b}(s_0, c_0)$*

(C4) *All simple cycles have non-negative effect.*

(C5) *All simple cycles have 0-effect.*

*Condition (C1) can be checked in non-deterministic logspace (*NL*), independently of the encoding of numbers. All other conditions can be verified in* NL *assuming unary encoding, and in* NC *(conditions (C4) and (C5) even in* NC^2*) assuming binary encoding.*

▶ **Lemma 25.** *Consider a deterministic one-counter net with initial state s_0.*

1. *For any $c_0 \in \mathbb{N}$, the language $\mathcal{L}(s_0, c_0)$ is universal if, and only if, all simple cycles are non-negative (C4), and all words shorter than the number of states are accepting (C2).*

2. *There exists an initial counter value* $c_0 \in \mathbb{N}$ *such that* $\mathcal{L}(s_0, c_0)$ *is universal if, and only if, all simple cycles are non-negative (C4), and the underlying automaton is universal (C1).*
3. *For any* $c_0 \in \mathbb{N}$*, there exists a bound* $b \in \mathbb{N}$ *such that the bounded language* $\mathcal{L}^{\leq b}(s_0, c_0)$ *is universal if, and only if, (C5) the effect of all simple cycles is* 0 *and (C3) all words shorter than the number of states are in* $\mathcal{L}^{\leq b'}(s_0, c_0)$ *for* $b' \stackrel{def}{=} |Q| \cdot \|\delta\|$*.*

The following is a direct consequence of Lemmas 24 and 25.

▶ **Theorem 26.** *The universality, initial-value universality, and bounded universality problems for deterministic one-counter nets are in* NL *assuming unary encoding, and in* NC *assuming binary encoding.*

For the special case of DOCN over single letter alphabets, it is possible to derive even better upper bounds, based on the particular shape of the underlying automaton.

Recall that a deterministic automaton over a singleton alphabet is in the shape of a lasso: it consists of an acyclic path that ends in a cycle.

▶ **Lemma 27.** *For any given deterministic one-counter net* $\mathcal{A} = (\Sigma, Q, s_0, \delta, F)$ *with* $|\Sigma| = 1$ *and* $c_0, b \in \mathbb{N}$*, one can verify in deterministic logspace (*L*) that (C1) the underlying DFA is universal. Moreover, conditions (C2), (C3), (C4), and (C5) as defined in Lemma 24 can be verified in* L *assuming unary encodings and in* NC^2 *assuming binary encodings.*

Using Lemma 27 and the characterisation of the three universality problems by Lemma 25, we get the desired complexity upper bounds.

▶ **Theorem 28.** *The universality, initial-value universality, and bounded universality problems of deterministic one-counter nets over a singleton alphabet are in* L *assuming unary encoding and in* NC^2 *assuming binary encoding.*

6 Unambiguous Systems

In line with the usual definition of unambiguous finite automata, we call an OCN with a given initial configuration *unambiguous* iff for every word in its language there exists exactly one accepting run. Since the language of an OCN depends in a monotone fashion on the initial counter value, there is also a related, but different, notion of unambiguity. We call an OCN (which has a fixed initial state s_0) *structurally unambiguous* if the unambiguity condition holds for every initial counter c_0. Notice that every OCN that has an unambiguous underlying automaton is necessarily structurally unambiguous. We will show (Lemma 32) that these conditions are in fact equivalent.

In [12], the complexity of the universality problem for unambiguous vector addition systems with states (VASSs) was studied. In particular, for unambiguous OCNs, it is shown that checking universality is in NC^2 and NL-hard, assuming unary encoded inputs, and in PSPACE and coNP-hard, assuming binary encoding. The special case of unambiguous OCN over a single letter alphabet is not considered there, nor are the initial-counter – and bounded universality problems. We discuss these problems in the remainder of this section.

We assume w.l.o.g, that for any given OCN, all states in the underlying automaton are reachable from the initial state, and that from every state it is possible to reach an accepting state. States that do not satisfy these properties can be removed in NL. Moreover, all algorithms we propose need to check universality for the underlying automaton, and hence rely on the following computability result (see [27] for a proof for general alphabet, and Appendix D for singleton alphabet).

▶ **Lemma 29.** *Universality of an unambiguous finite automaton over single letter alphabet is in* NL*, and over general alphabet is in* NC^2*.*

We will start by considering the universality problem for unambiguous OCNs over a single letter alphabet. Here, unambiguity implies a strong restriction on accepting runs: if a run is accepting then it contains at most one positive cycle (which may be iterated multiple times).

▶ **Lemma 30.** *Let $\pi = \pi_1\pi_2\pi_3$ be an accepting run where π_2 is a positive simple cycle. Then $\pi_3 = \pi_2^k\pi_4$ for some $k \in \mathbb{N}$ and acyclic path π_4.*

Proof. Assume towards contradiction that there is an accepting run $\pi = \pi_1\pi_2\pi_3\pi_4\pi_5$, where π_2 is a positive simple cycle and π_4 is a simple cycle. Based on this we show that the system cannot be unambiguous. Let $c = |Q| \cdot \|\delta\|$ and denote by $|\pi|$ the length of path π.

Since π_2 has a positive effect, it follows that $\pi' = \pi_1\pi_2^{|\pi_4|+c\cdot|\pi_2|}\pi_3\pi_4\pi_5$ is an accepting run. But there is a second run that reads the same word, namely $\pi'' = \pi_1\pi_2^{c\cdot|\pi_2|}\pi_3\pi_4^{|\pi_2|}\pi_5$. The second run is indeed a run as the increment along $\pi_2^{c\cdot|\pi_2|}$ is bigger than any possible negative effect of $\pi_4^{|\pi_2|}$. Moreover the lengths of both runs are the same as $\pi_2^{|\pi_4|} = \pi_4^{|\pi_2|}$. ◀

A consequence of Lemma 30 is that if along any accepting run the value of the counter exceeds $B_0 = |Q| \cdot \|\delta\|$ then it cannot drop to zero afterwards, as it would require at least one negative cycle to do so. One can therefore encode all counter values up to B_0 into the finite-state control and solve universality for the resulting UFA. Lemma 29 thus yields the following.

▶ **Theorem 31.** *The universality problem of unary encoded unambiguous one-counter nets over a singleton alphabet is in* NL*.*

We consider next the initial-value universality problem for unambiguous OCNs. Since whether an OCN is unambiguous depends on the initial counter value, the initial-value universality problem is only meaningful for structurally unambiguous systems, those which are unambiguous regardless of the initial counter. We first observe a simple fact about these definitions.

▶ **Lemma 32.** *An OCN is structurally unambiguous if and only if its underlying automaton is unambiguous.*

▶ **Lemma 33.** *Consider a structurally unambiguous OCN with initial state s_0. There exists an initial counter c_0 so that $\mathcal{L}(s_0, c_0) = \Sigma^*$ if, and only if, the underlying automaton is universal and has no negative cycles.*

The following is a direct consequence of Lemma 33 and the complexity bounds provided by Lemmas 24 and 29, for the cycle condition *(C4)*.

▶ **Theorem 34.** *The initial-value universality problem of structurally unambiguous one-counter nets is in* NC^2 *assuming binary encoding, and in* NL *assuming unary encoding and single-letter alphabets.*

Finally, we turn our attention to the bounded universality problem for unambiguous OCNs. This turns out to be quite easy, due to the following observation.

▶ **Lemma 35.** *If an unambiguous OCN is bounded universal then no accepting run contains a positive cycle.*

▶ **Theorem 36.** *The bounded universality problem of unambiguous one-counter nets with unary-encoded transition weights is in* NC^2*, and in* NL *if the alphabet has only one letter, and for binary-encoded transition weights it is in* PSPACE*.*

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Piotr Hofman, Richard Mayr, K. Narayan Kumar, and Patrick Totzke. Infinite-state energy games. In *LICS*. ACM, 2014. `doi:10.1145/2603088.2603100`.

2 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Richard Mayr Radu Ciobanu, and Patrick Totzke. Universal safety for timed Petri nets is PSPACE-complete. In *CONCUR*, 2018. URL: `http://dx.doi.org/10.4230/LIPIcs.CONCUR.2018.6`, `doi:10.4230/LIPIcs.CONCUR.2018.6`.

3 Parosh Aziz Abdulla, Pavel Krcal, and Wang Yi. R-automata. In *CONCUR*. Springer Berlin Heidelberg, 2008.

4 Alfred V Aho, John E Hopcroft, and Jeffrey D Ullman. *The design and analysis of computer algorithms*. Pearson, 1974.

5 S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *ATVA*, 2011.

6 Shaull Almagor, Nathann Cohen, Guillermo A. Pérez, Mahsa Shirmohammadi, and James Worrell. Coverability in 1-vass with disequality tests. *CoRR*, 2019. URL: `http://arxiv.org/abs/1902.06576`, `arXiv:1902.06576`.

7 Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *LICS*. IEEE Computer Society, 2015. `doi:10.1109/LICS.2015.14`.

8 Stanislav Böhm, Stefan Göller, Simon Halfon, and Piotr Hofman. On Büchi One-Counter Automata. In *STACS*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7019`, `doi:10.4230/LIPIcs.STACS.2017.14`.

9 Stanislav Böhm, Stefan Göller, and Petr Jančar. Bisimilarity of one-counter processes is pspace-complete. In *CONCUR*. Springer-Verlag, 2010.

10 Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. *Infinite Runs in Weighted Timed Automata with Energy Constraints*. 2008. `doi:10.1007/978-3-540-85778-5_4`.

11 Thomas Colcombet. Unambiguity in automata theory. In *DCFS*. Springer, 2015. `doi:10.1007/978-3-319-19225-3\_1`.

12 Wojciech Czerwiński, Diego Figueira, and Piotr Hofman. Universality Problem for Unambiguous VASS. 2020. URL: `https://hal.archives-ouvertes.fr/hal-02483495`.

13 A. Degorre, L. Doyen, R. Gentilini, J.F. Raskin, and S. Torunczyk. Energy and mean-payoff games with imperfect information. In *CSL*, 2010.

14 Stéphane Demri. On selective unboundedness of VASS. *J. Comput. Syst. Sci.*, 2013.

15 Piotr Hofman, Slawomir Lasota, Richard Mayr, and Patrick Totzke. Simulation problems over one-counter nets. *Log Meth Comput Sci*, 2016. `doi:10.2168/LMCS-12(1:6)2016`.

16 Piotr Hofman and Patrick Totzke. Trace inclusion for one-counter nets revisited. *Theor Comput Sci*, 2017. URL: `http://www.sciencedirect.com/science/article/pii/S0304397517303961`, `doi:https://doi.org/10.1016/j.tcs.2017.05.009`.

17 Petr Jancar, Javier Esparza, and Faron Moller. Petri Nets and Regular Processes. *J. Comput. Syst. Sci.*, 1999.

18 S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In *STOC*. ACM, 1988.

19 Jérôme Leroux and Grégoire Sutre. On flatness for 2-dimensional vector addition systems with states. In *CONCUR*. Springer Berlin Heidelberg, 2004.

20 Richard Mayr. Undecidability of weak bisimulation equivalence for 1-counter processes. In *ICALP*. Springer, 2003.

21 M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1 edition, 1967.

22 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor Comput Sci*, 1978. URL: `http://www.sciencedirect.com/science/article/pii/0304397578900361`, `doi:https://doi.org/10.1016/0304-3975(78)90036-1`.

23 Louis E. Rosier and Hsu-Chun Yen. A multiparameter analysis of the boundedness problem for vector addition systems. In *FCT*. Springer Berlin Heidelberg, 1985.

24 L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*. ACM, 1973. URL: `http://doi.acm.org/10.1145/800125.804029`, `doi:10.1145/800125.804029`.

25 Leslie G. Valiant. *Decision Procedures for Families of Deterministic Pushdown Automata*. PhD thesis, University of Warwick, 1973. URL: `http://wrap.warwick.ac.uk/34701/`.

26 Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag, 1999.

27 Tzeng Wen-Guey. On path equivalence of nondeterministic finite automata. *Inf Proc Let*, 1996. URL: `http://www.sciencedirect.com/science/article/pii/0020019096000397`, `doi:https://doi.org/10.1016/0020-0190(96)00039-7`.

A Proofs of Section 3

▶ **Theorem 1.** *The initial-value universality problem for one-counter nets is undecidable.*

Proof. We show that a given two-counter machine $\mathcal{M}$ halts if and only if the corresponding one-counter net $\mathcal{A}$, as constructed in Section 3.1, is initial-value universal.

$\Rightarrow$: *When $\mathcal{M}$ halts*, its (legal) run has some length $n-1$. We claim that $\mathcal{A}$ is universal with the initial value n.

Consider some word w over the alphabet of $\mathcal{A}$. We shall describe an accepting run ρ of $\mathcal{A}$ on w. Until the first occurrence of #, the run ρ is deterministically in q_0, which is accepting. We show that for every segment between two consequent #'s, as well as the segment after the last #, the run ρ may either reach *heaven* or reach q_0 with counter value at least n (and remains there until the next # or the end of the word), from which it follows that ρ is accepting.

If the segment is shorter than n, q_0 can choose to go to q_1 over #, and from there it will reach heaven. If the segment is longer than n, it cannot describe the legal run of $\mathcal{M}$. Then, it must cheat within up to n steps. We show that each of the 5 possible cheats fulfills the claim.

1. If it makes a non-counting cheat, q_0 will go to q_2 over #, and will reach *heaven*. (This is also the case if it has additional letters different from # after the '`halt`' letter.)
2. If it makes a positive cheat on x, q_0 will go to q_3 upon reading the next #. When the cheat occurs, the value of x is positive, while reading the letter '`x=0 then goto`'. Notice that the value of $\mathcal{A}$'s counter is accordingly bigger than its value when entering q_3 (and by the inductive assumption bigger than n). Then, q_3 goes to q_0 with weight -1, guaranteeing that $\mathcal{A}$'s counter value is at least n. Notice that the counter value cannot go below n at any point, since $\mathcal{M}$ cannot make the value of x negative without a counting cheat. (We equipped $\mathcal{M}$ with a counter check before every decrement.)
3. If it makes a negative cheat on x, q_0 will go to q_4. Then, when the cheat occurs, the value of x is 0, while there is the letter '`x>0 then goto`'. Notice that the value of $\mathcal{A}$'s counter is accordingly exactly its value when entering q_3 (and by the inductive assumption at least n). Then, q_4 goes to q_0 with weight 0, guaranteeing that $\mathcal{A}$'s counter value is at least n. Notice that the counter might go below n between getting to q_4 and returning to q_0. Yet, since the violation must occur within up to n steps, and the value of the counter when entering q_4 is at least n, we are guaranteed to be able to properly continue with the run, as the counter need not go below 0.

4-5. Analogously, if it makes a positive or negative cheat over y, the choice of q_0 will be q_5 or q_6, respectively.

$\Leftarrow$: *When $\mathcal{M}$ does not halt*, for every positive integer n, we build the word w_n and show that it is not accepted by $\mathcal{A}$ with an initial counter value n.

The word w_n consists of $n+1$ segments between #'s, where each segment is the prefix of length $n+1$ of the (legal) run of $\mathcal{M}$. Consider the possible runs of $\mathcal{A}$ on w_n. It cannot go from q_0 to q_1, because it will stop after n steps. It also cannot go to q_2, because there is no cheating. We show that if it goes to $q_3..q_6$, it must return to q_0 before the next #, while decreasing the value of $\mathcal{A}$'s counter, which can be done only n times until the run stops.

If it goes to q_3, it must return to q_0 upon some '`x=0 then goto`', as it cannot continue the run on #. Yet, as there is no cheating, it returns to q_0 when $x = 0$, which implies that $\mathcal{A}$'s counter has the same value as when entering q_3, and due to the -1 weight of the transition to q_0, it returns to q_0 while decreasing the value of $\mathcal{A}$'s counter by 1. An analogous argument follows if it goes to q_5.

If it goes to q_4, it must return to q_0 upon some '`x>0 then goto`', as it cannot continue the run on #. Yet, as there is no cheating, it returns to q_0 while the value of x is indeed strictly positive, which implies that the value of $\mathcal{A}$'s counter is smaller than the value it had when entering q_4, and therefore due to the 0-weight transition to q_0, it returns to q_0 with a smaller value of $\mathcal{A}$'s counter. An analogous argument follows if it goes to q_6. ◀

▶ **Theorem 2.** *The bounded universality problem for one-counter nets is undecidable.*

Proof. We show that a given two-counter machine $\mathcal{M}$ halts if and only if the corresponding one-counter net $\mathcal{A}'$, as constructed in Section 3.2, is bounded universal for an initial counter value 0.

$\Rightarrow$:*When $\mathcal{M}$ halts*, its (legal) run has some length $n-1$. We claim that $\mathcal{A}'$ is universal with the counter bound $2n$.

Consider some word w' over the alphabet of $\mathcal{A}'$. We shall describe an accepting run ρ' of $\mathcal{A}'$ on w'. In the first n steps, ρ' remains in q_0', increasing the counter to n. Then, it moves to q_0. In the rest of the run, ρ' continues as the accepting run ρ of $\mathcal{A}$ on the word w that is the suffix of w' from the $n+1$ position (as described in the proof of Theorem 1), except for the following changes: whenever it is in q_0 and the counter is bigger than n, it goes to q_7 on #. In q_7, it uses the self loop until the counter's value becomes n and then goes to q_0.

If the length of w' is up to n, then ρ' is obviously accepting, as it remains in the accepting states q_0' and q_0, and the counter need not exceed $2n$ nor go below 0.

If the length of w' is more than n, we prove that for every segment between two consequent #'s, as well as the segment after the last #, the run ρ' may either reach *heaven* or reach q_0 with counter value at least n, and proceed from q_0 to $q_1..q_6$ with counter value exactly n. This will immediately imply that ρ' is accepting.

The challenge is to show that the counter of $\mathcal{A}'$ never needs to exceed $2n$. (It does not go below 0, since we go from q_0 to $q_1..q_6$ with a counter value of at least n (in this case exactly n), which satisfies the assumptions in the proof of Theorem 1.)

Now, in states q_1, q_2, q_4, q_6, and q_7 there is no problem, as the counter never gets above its value when entering these states. Yet, in states q_3 and q_5 there is a potential problem, since $\mathcal{A}'$'s counter increases when $\mathcal{M}$'s counters increase. However, since the (legal) run of $\mathcal{M}$ is of length $n-1$, a violation must occur within up to n steps. Hence, getting to states q_3 and q_5 with counter value of exactly n, the run ρ' may return to q_0 over the first violation, and thus need not increase the counter's value to more than $2n$. Observe that when returning to q_0

the counter's value might be bigger than n, in which case ρ' will later decrease it to exactly n by going to q_7.

$\Leftarrow$: *When $\mathcal{M}$ does not halt*, for every positive integer n, we build the word w'_n and show that it is not accepted by $\mathcal{A}'$ for an initial counter value 0 and a bound n on the counter.

The word w'_n consists of $n+2$ segments between #'s, where each segment is the prefix of length n of the (legal) run of $\mathcal{M}$. Consider the possible runs of $\mathcal{A}'$ on w'_n. In q'_0 it can stay up to n steps, entering q_0 with a counter value of up to n. Then it should accept from q_0 the suffix of w'_n, which contains $n+1$ segments as described above. However, as shown in the proof of Theorem 1, using all states except for q_7, it must decrease the counter value in each segment, and so is the case if using q_7. Hence, the run must stop after at most n segments and cannot be accepting. ◀

B Proofs of Section 4

▶ **Lemma 5.** *Let π be an executable path of length n from (p, c) to (q, c'). Then there exists an executable path π' of length n in linear form whose length is at most $2|Q|^2$, from (p, c) to (q, c'') with $c'' \geq c'$.*

Proof. Let $n \in \mathcal{L}(s_0, c_0)$, and let $\pi = s_0, s_1, \ldots, s_n$ be an accepting run of the OCN on n. For each state q visited by π, let $\boldsymbol{f}(q)$ and $\boldsymbol{\ell}(q)$ denote the first and last indices where q occurs in π, respectively. Let Marks $\stackrel{def}{=} \{\boldsymbol{f}(q), \boldsymbol{\ell}(q) : q \text{ occurs in } \pi\}$ be the set of all markings in π. Observe that $|\text{Marks}| \leq 2|Q|$.

We reshape π into linear form in two phases. In the first phase, we move cycles around such that in the obtained path, any infix between two marked positions consists of a simple path, and a collection of simple cycles. In the second phase, we replace most of the simple cycles, such that any infix between two marked positions consists of a relatively short path, and a single repeating cycle (which completes the linear form). Crucially, in both phases we must take care that the path remains executable. The crux of the proof is that instead of simply shifting cycles, we also change their starting point, such that they always start from a nadir, thus making them executable with any counter value.

For the first phase, consider an interval $[i, i+|Q|]$ in π that does not intersect Marks (if no such interval exists, we proceed to the second phase). Since this interval has $|Q|+1$ states, it contains some simple cycle $\gamma = x_1, x_2, \ldots, x_k$. Let d be a nadir of γ, and observe that necessarily $\boldsymbol{f}(x_d) < i$ and $\boldsymbol{\ell}(x_d) > i + |Q|$, since the interval $[i, i+|Q|]$ does not contain any marks.

We now split into two cases.

- If *effect*$(\gamma) \geq 0$, we modify π by removing the cycle γ from the interval $[i, i+|Q|]$, and instead adding the shifted cycle $\gamma^{\leftarrow d}$ at index $\boldsymbol{f}(x_d)$.
  Observe that the modified path is still executable, since by Observation 3 the cycle $\gamma^{\leftarrow d}$ is good, and can be executed with any counter value, and following its execution, the remaining path either has higher counters (up to where γ occurred) or the same values as in π (after where γ occurred).
- If *effect*$(\gamma) < 0$, we modify π by removing the cycle γ from the interval $[i, i+|Q|]$, and instead adding the shifted cycle $\gamma^{\leftarrow d}$ at index $\boldsymbol{\ell}(x_d)$.
  Observe that the modified path is still executable. Indeed, by Observation 3 *effect*$(\gamma^{\leftarrow d}) = -$*depth*$(\gamma^{\leftarrow d})$, and so $\gamma^{\leftarrow d}$ can be executed as long as the counter is at least *effect*$(\gamma^{\leftarrow d})$. Moreover, removing this negative cycle results in a run in which, all counter-values from the index of removal are increased by $-$*effect*(γ). In particular, at index $\boldsymbol{\ell}(x_d)$ it is at

least $0 + \mathit{effect}(\gamma^{\leftarrow d})$, so $\gamma^{\leftarrow d}$ can be executed. Notice that moving a negative cycle like this results in a path that is executable an has the same effect as π.

This completes the first phase. We remark that conceptually, this cycle modification takes place in a single “shot” for all cycles, so that the indices in Marks do not change after every cycle is moved, but are rather the same for all cycles being moved (otherwise intervals may “expand”, and Marks becomes ill-defined).

We now proceed to the second phase. Let π' be the path obtained after the first phase. We refer to any cycle that was moved in π as a *dangling cycle*. Thus, π' consists of at most $2|Q|$ intervals[3] that contain no non-dangling cycles, and at most $2|Q|$ indices on which there are dangling cycles (namely the indices in Marks). Furthermore, the dangling cycles always start at their respective nadirs.

We now proceed to eliminate most dangling cycles at each state. Consider some mark $\boldsymbol{f}(q)$ or $\boldsymbol{\ell}(q)$ in Marks. For each $1 \leq t \leq |Q|$, consider all simple cycles of length t where q is a nadir, and let $\mu_{q,t}$ be such a cycle of maximal effect. We now replace every dangling cycle of length t in $\boldsymbol{f}(q)$ with $\mu_{q,t}$. Clearly the effect of the cycles does not decrease, so the path remains executable. Furthermore, we maintain the length of the paths, so the path still represents a run on n.

Finally, within each mark, we can bunch the cycles by length, so that all cycles of the same length are executed consecutively. Thus, the obtained path consists of at most $2|Q|$ simple paths and $2|Q| \cdot |Q| = 2|Q|^2$ simple cycles, which is a linear form as required. ◀

▶ **Lemma 7.** *There exists a bound $B_1 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that, for every $n \in \mathbb{N}$, if n is accepted by a run that visits a state $r \in$ Pump, then n has an accepting run of the form $\eta_1\gamma_r^t\eta_2$ for paths η_1, η_2 of length at most B_1.*

Proof. Let γ_r be a shortest good cycle on r, and let ρ be a an accepting run that passes through r. We write $\rho = \pi_1, r, \pi_2$, where π_r is a prefix of the run before it visits r and π_2 is the suffix after visiting r (note that r may occur in π_2). Furthermore, by Lemma 5 we can assume π_1 and π_2 are in linear form of length at most $2|Q|^2$. Thus, we can write $\pi_1 = \tau_0\gamma_1^{e_1}\tau_1 \cdots \tau_{k-1}{\gamma_k}^{e_k}\tau_k$ with $k \leq 2|Q|^2$, and similarly for π_2.

We now start by replacing negative cycles in π_1 and in π_2 by repetitions of γ_r (the good cycle on r). This is done as follows. For every subset of cycles whose combined length equals $m|\gamma_r|$ for some $m \in \mathbb{N}$, we remove those cycles and replace them by m iterations of the good cycle γ_r. Since we only remove negative cycles, and since γ_r has non-negative effect and depth 0, the run remains executable. Recall that the γ_i cycles are simple, and are therefore of length at most $|Q|$. Thus, after removing cycles in this manner, we are left with at most $|\gamma_r| - 1 \leq |Q|$ negative cycles of every length.

We now aim to remove non-negative cycles in the same fashion. This, however, requires some caution, as some cycles might have effect greater than that of γ_r, or appear before the run visits state r for the first time, and therefore replacing them with γ_r may cause the path to become non-executable. Recall that by Definition 4 (and indeed, by the construction in the proof of Lemma 5) all the non-negative γ_i cycles start from their nadir, and therefore have depth 0. In addition, after removing the negative cycles as done above, the path length (excluding the non-negative cycles) is at most $2|Q|^2 + |Q|^2 = 3|Q|^2$ in each of π_1 and π_2. Thus, the maximal depth possible along the entire path is $6|Q|^2\|\delta\|$. Thus, as long as a (strictly) positive cycle (or a combination thereof) is taken enough times to maintain

[3] The first and last indices of π must be marked and so there are in fact at most $2|Q| - 1$ intervals.

the counter above $6|Q|^2\|\delta\|$, the path remains executable. We can now proceed to replace non-negative cycles with γ_r in the same manner done for negative cycles, while maintaining executability.

We thus end up with a modified run of the form $\eta_1\gamma_r^t\eta_2$ where η_1 and η_2 are of length $\mathsf{poly}(|Q|, \|\delta\|)$, which implies the claim. ◀

▶ **Lemma 8.** *There exists a bound $B_2 \in \mathsf{poly}(\|\delta\| \cdot |Q|)$ such that, for every $r \in \text{Pump}$, there exists a DFA that accepts $\mathcal{L}^r(s_0, c_0)$ and is of size at most B_2.*

Proof. From Lemma 7 it follows that there exists a bound $B_1 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that every word accepted with a run that goes through r is of the form $x + y|\gamma_r|$ where $x, y \in \mathbb{N}$ and $x \leq B_0$. Thus, we can construct a DFA of size $B_2 \stackrel{def}{=} B_1 + |\gamma_r|$ whose form is an initial prefix of length B_1, followed by a cycle of length $|\gamma_r|$, and whose accepting states correspond to all the x above, with corresponding accepting states on the cycle. ◀

▶ **Lemma 9.** *There exists $B_3 \in \mathsf{poly}(\|\delta\|, |Q|)$ such that $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$ if, and only if, there exists $n \in \mathbb{N}$ such that either $n < B_2$ and $n \notin \mathcal{L}(s_0, c_0)$, or $B_3^{|Q|} \leq n \leq 2B_3^{|Q|}$ and $n \notin \mathcal{P}$.*

Proof. Let B_2 be as per Lemma 8, and define $B_3 \stackrel{def}{=} B_2^{|\text{Pump}|} \leq B_2^{|Q|}$. Observe that by taking the product of the DFAs obtained in Lemma 8, we can construct a DFA $\mathcal{D}$ of size at most B_3 for $\mathbb{N} \setminus \mathcal{P}$. Then, $\mathbb{N} \setminus \mathcal{P}$ is infinite iff there exists a word of length $B_3 \leq n \leq 2B_3$ that is accepted by $\mathcal{D}$ (as such a word is necessarily accepted by a run that contains a cycle in $\mathcal{D}$).

Towards the claim, if $\mathbb{N} \setminus \mathcal{P}$ is infinite, then $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$, and clearly if there exists $n < B_2$ such that $n \notin \mathcal{L}(s_0, c_0)$ then again, $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$.

Conversely, assume $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$. We claim that either there exists $n < B_2$ with $n \notin \mathcal{L}(s_0, c_0)$, or $\mathbb{N} \setminus \mathcal{P}$ is infinite. Indeed, observe that since $\mathcal{D}$ is obtained as the product of singleton-alphabet DFAs, then it has a "lasso" shape: a finite prefix of states, followed by a cycle. Moreover, the size of the prefix is at most B_2, namely the maximal size of the prefix in each of the DFAs in the product. Thus, if there exists $n < B_2$ with $n \notin \mathcal{L}(s_0, c_0)$ then we are done, and otherwise there is some $n > B_2$ with $n \notin \mathcal{L}(s_0, c_0)$, and in particular $n \notin \mathcal{P}$, so $\mathcal{D}$ accepts some word along its cycle, and so accepts infinitely many words, and in particular some word $B_3 \leq n \leq 2B_3$. ◀

▶ **Lemma 11.** *Let $\pi = \tau_0\gamma_1^{e_1}\tau_1 \cdots \tau_{k-1}\gamma_k^{e_k}\tau_k$ be a run in linear form, then we can check whether π is executable from counter value c in time polynomial in the description of π.*

Proof. Checking that the transitions follow those of the OCN can be done in polynomial time, since we only need to check the underlying path, regardless of the exponents. In order to check that the counter value remains non-negative, we observe that for any cycle γ_i, if $\mathit{effect}(\gamma_i) \geq 0$, then γ_i is taken from a nadir (by Definition 4), and hence can be taken with any counter value. If that is the case, then we can compute directly $\mathit{effect}(\gamma_i^{e_i}) = e_i \cdot \mathit{effect}(\gamma_i)$. Otherwise, if $\mathit{effect}(\gamma_i) < 0$, then in order to check if $\gamma_i^{e_i}$ is executable from counter value c, it suffices to check that $(e_i - 1) \cdot \mathit{effect}(\gamma_i) - \mathit{depth}(\gamma_i) \leq c$. Indeed, for negative cycles, the last iteration is the "hardest". Again, we can now compute $\mathit{effect}(\gamma_i^{e_i}) = e_i \cdot \mathit{effect}(\gamma_i)$.

Thus, we can keep track of the counter value along the underlying path, and update it directly for every cycle. This takes polynomial time overall. ◀

▶ **Lemma 13.** *There exists c_0 such that $\mathcal{L}_{\mathcal{A}}(s_0, c_0) = \mathbb{N}$ iff $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$ and $\mathbb{N} \setminus \mathcal{P}$ is finite.*

Proof. For the first direction, assume $\mathcal{L}_{\mathcal{A}}(s_0, c_0) = \mathbb{N}$ for some c_0. Clearly $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$ as otherwise some word is not accepted in the underlying NFA, let alone the OCN. Assume by way of contradiction that $\mathbb{N} \setminus \mathcal{P}$ is infinite, and recall that in every accepting run on a word $n \in \mathbb{N} \setminus \mathcal{P}$, all cycles must be negative. Thus, for long enough words, the counter value, starting at c_0, must become negative, which is a contradiction.

Conversely, if $\mathbb{N} \setminus \mathcal{P}$ is finite and $\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$, we can take an initial counter value large enough so that all words not in $\mathcal{P}$ have accepting runs. Then, similarly to Lemma 7, we can show that every word in $\mathcal{P}$ has an accepting run of the form $\tau_1 \gamma_r^t \tau_2$ with τ_1 and τ_2 of length $\mathsf{poly}(|Q|)$ and where γ_r is the canonical good cycle from state $r \in \text{Pump}$ with maximal effect. Notice here that the bound on the lengths of paths τ_1 and τ_2 is polynomial only in the number of states and not, as in Lemma 7, also in $\|\delta\|$. This is because we can safely remove any combination of simple cycles in these sub-paths without preserving the executability of the resulting path in the net. A large enough counter value ensures that the prefix and suffix are executable, so all words in $\mathcal{P}$ are accepted as well. ◀

▶ **Lemma 17.** *There exists a bound $B_5 \in \mathsf{poly}(|Q|, \|\delta\|)$ such that, every stable state q admits a zero cycle of length and depth at most B_5.*

Proof. If q is at the nadir of a simple zero cycle, then $|Q|$ bounds its length and we are done.

Otherwise, since q admits a negative cycle, then there is a state $x \in Q$ that admits a simple negative cycle γ such that x and q are reachable from each other. Let τ_1 and τ_2 be simple paths from q to x and from x to q, respectively. Let $s = \mathit{effect}(\tau_1 \tau_2) + 1$, then $\chi = \tau_1 \gamma^s \tau_2$ is a negative cycle of length at most $3|Q| \cdot \|\delta\|$.

Let η be a simple positive cycle that has a nadir at q. Then q admits the zero cycle $\zeta_q = \eta^{-\mathit{effect}(\chi)} \cdot \chi^{\mathit{effect}(\eta)}$ and $B_5 \stackrel{def}{=} |Q| \cdot (|Q| \cdot \|\delta\|) + (3|Q| \cdot \|\delta\|) \cdot |Q|$ satisfies the claim. ◀

▶ **Lemma 21.** *$\mathcal{L}(s_0, c_0)$ is bounded-universal if, and only if, the underlying automaton $\mathcal{N}$ is universal ($\mathcal{L}_{\mathcal{N}}(s_0) = \mathbb{N}$) and $\mathbb{N} \setminus \mathcal{S}$ is finite.*

Proof. By Theorem 19, there exists a bound B_7 such that all words in $\mathcal{S}$ are accepted with paths whose counter values remains below B_7. Hence, if there are only finitely many words that are outside $\mathcal{S}$, and $\mathcal{L}_{\mathcal{N}}(c_0) = \mathbb{N}$, then the counter values among the runs on the remaining finite set of words are clearly bounded. Hence, $\mathcal{L}(s_0, c_0)$ is bounded-universal.

Conversely, assume $\mathbb{N} \setminus \mathcal{S}$ is infinite, we show that $\mathcal{L}(s_0, c_0)$ is not bounded-universal. First, if $\mathcal{L}_{\mathcal{N}}(s_0) \neq \mathbb{N}$ the OCN cannot be universal, and in particular it is not bounded-universal. Observe that by Definition 16, words outside $\mathcal{S}$ can be accepted only with paths on which the number of alternations between positive and negative cycles is at most $|Q|$, and that do not contain zero cycles. Since only finitely many words can be accepted using a bounded number of positive cycles, it follows that if $\mathbb{N} \setminus \mathcal{S}$ is infinite, then for every $M \in \mathbb{N}$ there exists a word that is only accepted by runs that have a positive cycle taken at least M times, and hence have effect at least M. It follows that $\mathcal{L}(s_0, c_0)$ is not bounded-universal. ◀

▶ **Theorem 12.** *The universality problem for singleton-alphabet one-counter nets with transitions encoded in binary is in $\mathsf{coNP}^{\mathsf{NP}}$.*

Proof. Following our algorithmic scheme, an $\mathsf{NP}^{\mathsf{NP}}$ algorithm for non-universality proceeds as follows. non-deterministically either (1) guess $n < B_3$, and check (using an NP oracle as per Lemma 11) that $n \notin \mathcal{L}(s_0, c_0)$, or (2) guess $B_3^{|Q|} \leq n \leq 2B_3^{|Q|}$ and check that $n \notin \mathcal{L}_{\mathcal{A}^r}(s_0, c_0)$ for all $r \in \text{Pump}$, using $|Q|$ calls to an NP oracle as per Lemma 11. ◀

C Proofs of Section 5

▶ **Lemma 23.** *Given a set $S = \{\alpha_1, \alpha_2 \dots \alpha_n\}$ of integers written in binary, the question whether the sum of all elements in S is non-negative is in NC^2.*

Proof. Addition of two integers written in binary can be done in AC^0 [26], and therefore in NC^1. As the summation of n numbers can be done in $\log n$ iterations (whereby each iteration reduces the number of elements by a factor of 2 by adding up α_{2i} and α_{2i+1}, for every index i up to half the number of elements), and each iteration is in NC^1 (by performing in parallel all of these additions), we get that the overall problem is in NC^2. ◀

▶ **Lemma 24** (Basic Conditions)**.** *Consider the following conditions on a deterministic one-counter net $\mathcal{A} = (\Sigma, Q, s_0, \delta, F)$, initial value $c_0 \in \mathbb{N}$, and bound $b \in \mathbb{N}$.*
(C1) *The underlying automaton is universal.*
(C2) *Every word w of length $|w| \leq |Q|$ is in $\mathcal{L}(s_0, c_0)$*
(C3) *Every word w of length $|w| \leq |Q|$ is in $\mathcal{L}^{\leq b}(s_0, c_0)$*
(C4) *All simple cycles have non-negative effect.*
(C5) *All simple cycles have 0-effect.*
Condition (C1) can be checked in non-deterministic logspace (NL), independently of the encoding of numbers. All other conditions can be verified in NL assuming unary encoding, and in NC (conditions (C4) and (C5) even in NC^2) assuming binary encoding.

Proof. Unary encoding. All conditions can be shown to be in NL using the theorems of Savitch (reachability in finite directed graphs is in NL) and Immerman–Szelepcsényi ($\mathsf{NL} = \mathsf{coNL}$). Indeed, *(C1)* holds iff no non-accepting state is reachable in the underlying automaton.For the remaining conditions, just notice that the assumption that inputs are given in unary means that all relevant numbers are bounded polynomially in the input. For instance, to show that *(C4)* does not hold, one simply guesses the offending simple cycle and stepwise computes its effect in binary representation.

Binary encoding. Let's first consider condition *(C2)*. This fails iff there is a short word whose run in $\mathcal{A}$ either ends in a non-accepting state or reduces the counter below zero. The first case is again a simple reachability condition in the underlying DFA. The second case reduces to a coverability problem as follows.

For $k \in \mathbb{N}$, let $\mathcal{A} \times k \stackrel{def}{=} (Q \times \{0, 1, \dots, k\}, \Sigma, \delta', F', s'_0)$ be the OCN that results from $\mathcal{A}$ by adding a step-counter up to k into the states. That is, $\delta' \stackrel{def}{=} \{((p, i), \alpha, e, (q, i+1)) : (p, \alpha, e, q) \in \delta, i \leq k\}$, $F' \stackrel{def}{=} F \times \{0 \dots k\}$, and $s'_0 \stackrel{def}{=} (s_0, 0)$. Further, let $\mathcal{B}$ denote the OCN $\mathcal{A} \times |Q|$, in which all transition effects are inverted. Notice that for every word w of length $|w| \leq |Q|$, the effect of its induced run in $\mathcal{A}$ (and $\mathcal{B}$) is between $-B$ and B, for $B \stackrel{def}{=} |Q| \cdot \|\delta\|$. Such a word cannot be accepted by $\mathcal{A}$ from (s_0, c_0) iff the run it induces in $\mathcal{B}$ starting from (s'_0, B) leads to some configuration $((q, |w|), (B + c_0 + 1))$. This reachability question about $\mathcal{B}$ can be answered in NC [6, Lemma 1 and Theorem 15], and since $\mathcal{A}$ and $\mathcal{B}$ are of polynomially the same size, also in NC with respect to $\mathcal{A}$.

An NC upper bound for condition *(C3)* is completely analogous and differs only in that an additional reachability check should be taken, in which the weights in $\mathcal{B}$ are not inverted and the target configuration is $((q, |w|), (B + b - c_0 + 1))$.

Conditions *(C4)* and *(C5)* on the effect of simple cycles can be verified in NC by a similar reduction to coverability. For example, to check if a simple cycle with negative effect exists it suffices to check that it is possible in $\mathcal{B}$ to start in a configuration $((q, 0), B)$ and cover a configuration $((q, k), (B + 1))$ for some $0 < k < |Q|$.

We can do slightly better than that and check these conditions in NC^2, as follows. Let $Q = \{p_1, p_2, \ldots, p_{|Q|}\}$, and for every $0 < k < |Q|$, let M_k denote the $|Q| \times |Q|$ matrix of elements in $\mathbb{Z} \cup \infty$, where the entry for i, j equals the minimal effect of a path of length k from state p_i to p_j. Then, M_k can be computed in NC^2 using standard repeated-squaring in the min-plus semiring [4]

To check condition *(C4)*, that all simple cycles have non-negative effect, we just need to check (in parallel) that all entries in the main diagonal of all the M_k matrices are non-negative. The same procedure, applied to an OCN that is derived from $\mathcal{A}$ by inverting all transition weights, allows to check for the presence of positive simple cycles, and hence for an NC^2 algorithm to check condition *(C5)*. ◀

▶ **Lemma 25.** *Consider a deterministic one-counter net with initial state s_0.*

1. *For any $c_0 \in \mathbb{N}$, the language $\mathcal{L}(s_0, c_0)$ is universal if, and only if, all simple cycles are non-negative (C4), and all words shorter than the number of states are accepting (C2).*
2. *There exists an initial counter value $c_0 \in \mathbb{N}$ such that $\mathcal{L}(s_0, c_0)$ is universal if, and only if, all simple cycles are non-negative (C4), and the underlying automaton is universal (C1).*
3. *For any $c_0 \in \mathbb{N}$, there exists a bound $b \in \mathbb{N}$ such that the bounded language $\mathcal{L}^{\leq b}(s_0, c_0)$ is universal if, and only if, (C5) the effect of all simple cycles is 0 and (C3) all words shorter than the number of states are in $\mathcal{L}^{\leq b'}(s_0, c_0)$ for $b' \stackrel{def}{=} |Q| \cdot \|\delta\|$.*

Proof. **1.** (Normal Universality): Clearly both conditions are necessary for the system to be universal. To see why they are sufficient for universality, assume that *(C4)* holds and consider shortest word $w \notin \mathcal{L}(s_0, c_0)$. Then the run on w cannot contain any non-negative cycle because this would contradict the minimality assumption. Since we assume *(C4)*, that all cycles are non-negative, the run on w must have no cycles. Thus, $|w| \leq |Q|$ which is impossible due to *(C2)*.

2. (Initial-Value Universality): If both conditions hold then any cycle on any run must have non-negative effect. So if one picks $c_0 \stackrel{def}{=} |Q| \cdot \|\delta\|$ then the counter cannot become negative on any run and the language $\mathcal{L}(s_0, c_0)$ equals that of the underlying automaton, namely Σ^* by condition *(C1)*.
Conversely, since $\mathcal{L}(s_0, c_0)$ is always included in the language of the underlying automaton, condition (C1) is clearly necessary. If (C4) fails then, because the system is deterministic, for every number c_0 there must be a word $w(c_0) \in \Sigma^*$ whose run has an effect strictly below $-c_0$. Then $w \notin \mathcal{L}(s_0, c_0)$. Therefore both conditions are necessary.

3. (Bounded Universality): Trivially, both conditions are necessary. For the opposite direction, assume that the conditions hold. We contradict the assumption that $\mathcal{L}^{\leq b'}(s_0, c_0) \neq \Sigma^*$. If that was the case, we can pick a shortest word w not in that language. The run of this word cannot contain a cycle, because by condition (C5) all cycles have zero effect on the counter and therefore the presence of a cycle on the run would contradict the assumed minimality of $|w|$. This implies that w is no longer than the number of states, and by condition (C3) it must be in $\mathcal{L}^{\leq b'}(s_0, c_0)$. Contradiction. ◀

▶ **Lemma 27.** *For any given deterministic one-counter net $\mathcal{A} = (\Sigma, Q, s_0, \delta, F)$ with $|\Sigma| = 1$ and $c_0, b \in \mathbb{N}$, one can verify in deterministic logspace (L) that (C1) the underlying DFA is universal. Moreover, conditions (C2), (C3), (C4), and (C5) as defined in Lemma 24 can be verified in L assuming unary encodings and in NC^2 assuming binary encodings.*

Proof. Condition *(C1)* is equivalent to checking that all states are accepting ($Q = F$). For the other conditions, notice that if all numbers are encoded in unary then one only needs to compute numbers bounded polynomially in $|Q|$ and $\|\delta\|$. This can be done in deterministic logspace by representing them in binary. If numbers are already encoded in binary then the NC^2 bounds follow from Lemma 23. ◀

D Proofs of Section 6

▶ **Lemma 29.** *Universality of an unambiguous finite automaton over single letter alphabet is in* NL*, and over general alphabet is in* NC^2*.*

Proof. The lemma was proven in [27], for the general alphabet. For the single letter alphabet we have that if the language is not universal then the shortest not accepted word is bounded by $|Q|$ [11] (Lemma 2). Thus to verify universality, we need to test if for every $0 \leq i \leq |Q|$ there is an accepting run of length i, which can be tested in NL. ◀

▶ **Theorem 31.** *The universality problem of unary encoded unambiguous one-counter nets over a singleton alphabet is in* NL*.*

Proof. By Lemma 30 it is possible to construct an unambiguous finite automaton (UFA) of polynomial size, which is universal if and only if the net is universal. This can be done by bounding the counter from above by B_0, remembering its value in the states, and switching to a copy of the underlying automaton once the counter is observed to exceed this bound. It is easy to see that every run in the net induces a run in the automaton and vice-versa. The number of states of this new finite automaton is $|Q| \cdot (1 + B_0) + |Q|$. Since the constructed UFA is still over a single letter alphabet, we can check if it is universal NL by Lemma 29. ◀

▶ **Lemma 32.** *An OCN is structurally unambiguous if and only if its underlying automaton is unambiguous.*

Proof. If the underlying automaton is unambiguous then the net is as well, as every run of the net is also a run of the automaton.

In the opposite direction, suppose that the underlying automaton is not unambiguous, then there is a word w read by two accepting runs π_1 and π_2. If we start with the counter value bigger than $(|\pi_1| + |\pi_2|) \cdot \|\delta\|$ then the both runs in the underlying automaton will describe two different accepting runs in the OCN. ◀

▶ **Lemma 33.** *Consider a structurally unambiguous OCN with initial state s_0. There exists an initial counter c_0 so that $\mathcal{L}(s_0, c_0) = \Sigma^*$ if, and only if, the underlying automaton is universal and has no negative cycles.*

Proof. *"If"*. If all cycles have non-negative effect then an initial value of $c_0 \stackrel{def}{=} B_0$ suffices to ensure that no run can drop the counter below zero. Consequently, the system behaves just like its underlying automaton, which is universal by assumption.

"Only if". The language $\mathcal{L}(s_0)$ of the underlying automaton clearly includes $\mathcal{L}(s_0, c)$ for any value $c \in \mathbb{N}$. By assumption that there is c_0 with $\mathcal{L}(s_0, c_0) = \Sigma^*$, the underlying automaton must be universal.

It remains to show that it cannot contain any (reachable) simple cycles with negative effect. Towards a contradiction, suppose that $\pi_1\pi_2\pi_3$ is an accepting run from a configuration (s_0, c_0) and that $\mathit{effect}(\pi_2) < 0$. Then there is must exist $k \in \mathbb{N}$ such that $\pi_1\pi_2^k\pi_3$ is not a run from the configuration (s_0, c_0), as the counter runs out. By assumption, that the language of the net with initial configuration (s_0, c_0) is universal, there must be another run π_4 on the same word, and which is accepting. But now both runs, π_4 and $\pi_1\pi_2^k\pi_3$, are accepting from the configuration $(s_0, c_0 + \|\delta\| \cdot |\pi_2| \cdot k)$ as the effect of π_2^k is larger than $\|\delta\| \cdot |\pi_2| \cdot k$. This means that the net is not structurally unambiguous, which contradicts our assumptions. ◀

▶ **Lemma 35.** *If an unambiguous OCN is bounded universal then no accepting run contains a positive cycle.*

Proof. Suppose otherwise, then for any bound k there will be an accepting run which is going through configurations with counter value bigger than k, and from unambiguity, there is no other run that stays below the bound. ◀

▶ **Theorem 36.** *The bounded universality problem of unambiguous one-counter nets with unary-encoded transition weights is in* NC^2*, and in* NL *if the alphabet has only one letter, and for binary-encoded transition weights it is in* PSPACE*.*

Proof. Unary encoded transitions: By Lemma 35, if the OCN is bounded universal then every accepting run will only visit counter values below $B_1 \stackrel{def}{=} c_0 + B_0 = c_0 + |Q| \cdot \|\delta\|$. This means that the OCN is bounded universal if, and only if, $\mathcal{L}^{\leq B_1}(s_0, c_0) = \Sigma^*$. This can be verified by checking universality for the UFA that results by remembering all bounded counter values in the finite state space. The claim now follows by Lemma 29.

Binary encoded transitions: By Lemma 35, if the OCN is bounded universal then every accepting run will only visit counter values below $B_1 \stackrel{def}{=} c_0 + B_0 = c_0 + |Q| \cdot \|\delta\|$. This means that the OCN is bounded universal if, and only if, $\mathcal{L}^{\leq B_1}(s_0, c_0) = \Sigma^*$. This can be verified by checking universality for the UFA that results by remembering all bounded counter values in the finite state space. The claim now follows by Lemma 29 and the following fact NC$=$ $PolyLog$ applied to the UFA which is of exponential size. ◀