

Parametrized Universality Problems for One-Counter Nets

Shaul Almagor 


Technion, Israel

Udi Boker

Interdisciplinary Center (IDC) Herzliya, Israel

Piotr Hofman 

University of Warsaw, Poland

Patrick Totzke 

University of Liverpool, United Kingdom

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?

Although the ordinary universality problem is decidable (and Ackermann-complete) and these parameterized variants 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

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.34

Related Version A full version of the paper is [5], available at <https://arxiv.org/abs/2005.03435>.

Funding *Shaul 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.

Piotr Hofman: Supported by the NCN grant 2017/27/B/ST6/02093.

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



© Shaul Almagor, Udi Boker, Piotr Hofman, Patrick Totzke;
licensed under Creative Commons License CC-BY

31st International Conference on Concurrency Theory (CONCUR 2020).

Editors: Igor Konnov and Laura Kovács; Article No. 34; pp. 34:1–34:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 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.

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 second 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.

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 [15], 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 [12, 4], reducing the halting problem of two-counter machines to our setting.

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 [18, 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 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

■ **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 ; [11] NL-hard	NL Theorem 34	NC^2 Theorem 34	NL Theorem 36	NC^2 Theorem 36
Non-deterministic	coNP-comp. Theorem 10	Ackermann [15]	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 [11]	NC^2 Theorem 34	NC^2 Theorem 34	coNP ^{NP} Theorem 22	PSPACE Theorem 36
Non-deterministic	coNP ^{NP} Theorem 12	Ackermann [15]	coNP-comp. Theorem 15	Undecidable Theorem 1	coNP ^{NP} Theorem 22	Undecidable Theorem 2

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 [24], 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 [14]. 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 [16] and Ackermann-complete for the special case of OCNs [15]. 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 [19]. 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 reachab-

ility set) which is EXPSPACE-complete [21, 13] in general and PSPACE-complete for systems with fixed dimensions [22]. 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 [17].

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 [12, 4].

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 $\text{effect}(t) \stackrel{\text{def}}{=} e$ 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 *effect*(π) $\stackrel{\text{def}}{=} \sum_{i=1}^k e_i$ 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') ,

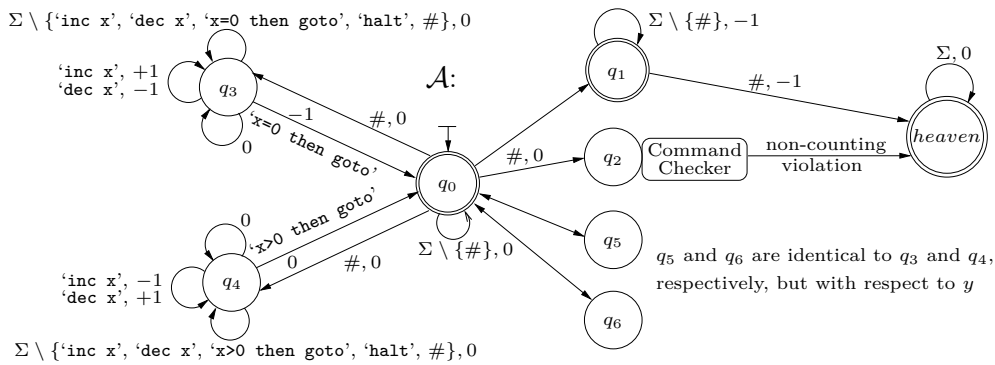
$$(s, c) \xrightarrow{a} (s', c') \iff (s, a, d, s') \in \delta \quad \text{and} \quad c' = c + d.$$

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 $1 \leq i \leq k$. Naturally, a run uniquely describes a path in the underlying finite OCN. Conversely, for every such path and initial counter value $c_0 \in \mathbb{N}$, 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 $c_i \leq b$ for all $0 \leq i \leq k$.

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* $\mathcal{L}^{\leq b}(s, c)$ 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 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) .

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



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

locations of the machine. There are five possible forms of commands: $\text{inc}(c)$, $\text{dec}(c)$, $\text{goto } l_i$, halt , if $c=0$ $\text{goto } l_i$ else $\text{goto } l_j$, where $c \in \{x, y\}$ is a counter and $1 \leq i, j \leq n$ are locations. The counters are initially set to 0. Since we can always check whether $c = 0$ before a $\text{dec}(c)$ command, we assume that the machine never reaches $\text{dec}(c)$ 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 undecidable halting problem of two-counter machines (2CM) [20].

The idea underlying both reductions 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 whether the input word, which encodes a prefix of the run of the 2CM, 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 2CM now implies that its length corresponds to a sufficient initial bound for this simulating OCN to be universal. Conversely, if the run of the machine does not halt then for every bound n , there exists a non-cheating, and non-terminating prefix of length n . Repeating this prefix n times witnesses non-universality for the simulating OCN with initial counter n .

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 holds: 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 $l = \text{'goto } i\text{'}$ and $a = \text{'inc } x\text{'}$, the transition from q goes to a forever accepting state (*heaven*), and if $a = \text{'goto } i\text{'}$, 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} : $\text{'inc } x\text{'}$, $\text{'inc } y\text{'}$, $\text{'dec } x\text{'}$, $\text{'dec } y\text{'}$, 'halt' , and for every line i of \mathcal{M} , the commands $\text{'goto } i\text{'}$, $\text{'x=0 then goto } i\text{'}$, $\text{'y=0 then goto } i\text{'}$, $\text{'x>0 then goto } i\text{'}$, and $\text{'y>0 then goto } i\text{'}$.
- 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 consecutive $\#$'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 $\text{'inc } x\text{'}$ with weight $+1$ and over $\text{'dec } x\text{'}$ 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 letters, 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 $\text{'inc } x\text{'}$ with weight -1 and over $\text{'dec } x\text{'}$ 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 letters, except for 'halt' and $\#$, it has a self loop with weight 0.
- 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 .

► **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 consecutive $\#$'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.

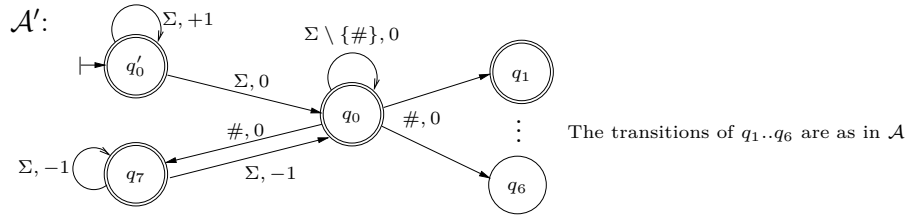
1. If it makes a non-counting cheat, q_0 will go to q_2 over $\#$, and will reach *heaven*. (This is also the case if it has additional letters different from $\#$ after the 'halt' letter.)
2. If it makes a positive cheat on x , q_0 will go to q_3 upon reading the next $\#$. When the cheat occurs, the 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



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

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 . ◀

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 . We refer the reader to the full version [5] for a detailed proof.

► **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 [23], a lower bound which trivially carries over to all problems considered here¹.

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 approach to find it, or it is long, in which case we deduce its existence by analyzing some

¹ The proof in [23, 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.

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 $\gamma = s_1, s_2, \dots, s_k$ (with $s_1 = s_k$). Recall that $effect(\gamma)$ is the sum of weights along γ and $depth(\gamma)$ 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, $effect(s_1, \dots, s_d) = -depth(\gamma)$. We say that γ is *positive* if $effect(\gamma)$ is positive (and similarly for negative, non-negative, zero, etc.). We call γ *good* if it is 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 $\gamma^{\leftarrow d} \stackrel{def}{=} s_d s_{d+1}, \dots, s_k, s_2, \dots, s_d$ 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 [18, 7].

► **Definition 4 (Linear Forms).** *A path π is in linear form if there exist simple cycles $\gamma_1, \dots, \gamma_k$ and paths τ_0, \dots, τ_k such that $\pi = \tau_0 \gamma_1^{e_1} \tau_1 \dots \tau_{k-1} \gamma_k^{e_k} \tau_k$ for some numbers $e_1, \dots, e_k \in \mathbb{N}$, 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 $\tau_0 \gamma_1 \tau_1 \dots \gamma_k \tau_k$ 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 $\text{Pump} \subseteq Q$ 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 $B_1 \in \text{poly}(|Q|, \|\delta\|)$ 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 $B_2 \in \text{poly}(\|\delta\| \cdot |Q|)$ 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 $\mathcal{P} \stackrel{\text{def}}{=} \bigcup_{r \in \text{Pump}} \mathcal{L}^r(s_0, c_0)$. 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 $B_3 \in \text{poly}(\|\delta\|, |Q|)$ such that $\mathcal{L}(s_0, c_0) \neq \mathbb{N}$ if, and only if, there exists $n \in \mathbb{N}$ such that either $n < B_3$ 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 [23]. 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 coNP^{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 $\mathcal{L}_{\mathcal{A}^r}(s_0, c_0) = \mathcal{L}_{\mathcal{A}}^r(s_0, c_0)$ 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 coNP^{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 $\mathcal{L}_{\mathcal{N}}^r(s_0)$ 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 $\mathcal{P} \stackrel{\text{def}}{=} \bigcup_{r \in \text{Pump}} \mathcal{L}_{\mathcal{N}}^r(s_0)$.

► **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 $B_4 \in \text{poly}(|Q|)$ such that, for every $r \in \text{Pump}$ 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 [23].

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 $B_5 \in \text{poly}(|Q|, \|\delta\|)$ 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 $\mathcal{S} \stackrel{\text{def}}{=} \bigcup_{r \in \text{Stable}} \mathcal{L}^r(s_0, c_0)$. We prove an analogue of Lemma 7.

► **Lemma 18.** *There exists a bound $B_6 \in \text{poly}(|Q|, \|\delta\|)$ 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 $\text{depth}(\zeta_r)$, 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 be 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 $B_6 \in \text{poly}(|Q|, \|\delta\|)$ 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 $B_7 \in \text{poly}(|Q|, \|\delta\|)$ 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 coNP^{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. Lemma 24 lists these conditions and upper complexity bounds for checking them. 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 [25].

► **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 .*

► **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 $b' \stackrel{\text{def}}{=} |Q| \cdot \|\delta\|$.

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 for 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 [11], 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 [26] for a proof for general alphabet, and the full paper 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 *ACM/IEEE Symposium on Logic in Computer Science (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 *International Conference on Concurrency Theory (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 *International Conference on Concurrency Theory (CONCUR)*, 2008.
- 4 S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2011.
- 5 Shaull Almagor, Udi Boker, Piotr Hofman, and Patrick Totzke. Parametrized universality problems for one-counter nets. *CoRR*, 2020. URL: <https://arxiv.org/abs/2005.03435>, arXiv:2005.03435.
- 6 Shaull Almagor, Nathann Cohen, Guillermo A. Pérez, Mahsa Shirmohammadi, and James Worrell. Coverability in 1-vass with disequality tests. In *International Conference on Concurrency Theory (CONCUR)*, 2020.
- 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 *ACM/IEEE Symposium on Logic in Computer Science (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 *International Symposium on Theoretical Aspects of Computer Science (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 Jančár. Bisimilarity of one-counter processes is pspace-complete. In *International Conference on Concurrency Theory (CONCUR)*, 2010.
- 10 Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2008. doi:10.1007/978-3-540-85778-5_4.
- 11 Wojciech Czerwinski, Diego Figueira, and Piotr Hofman. Universality Problem for Unambiguous VASS. In *International Conference on Concurrency Theory (CONCUR)*, 2020.
- 12 A. Degorre, L. Doyen, R. Gentilini, J.F. Raskin, and S. Torunczyk. Energy and mean-payoff games with imperfect information. In *Computer Science Logic (CSL)*, 2010.
- 13 Stéphane Demri. On selective unboundedness of VASS. *Journal of Computer and System Sciences*, 2013.
- 14 Piotr Hofman, Slawomir Lasota, Richard Mayr, and Patrick Totzke. Simulation problems over one-counter nets. *Logical Methods in Computer Science*, 2016. doi:10.2168/LMCS-12(1:6)2016.
- 15 Piotr Hofman and Patrick Totzke. Trace inclusion for one-counter nets revisited. *Theoretical Computer Science*, 2017. URL: <http://www.sciencedirect.com/science/article/pii/S0304397517303961>, doi:<https://doi.org/10.1016/j.tcs.2017.05.009>.
- 16 Petr Jancar, Javier Esparza, and Faron Moller. Petri Nets and Regular Processes. *Journal of Computer and System Sciences*, 1999.
- 17 S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In *Symposium on Theory of Computing (STOC)*. ACM, 1988.
- 18 Jérôme Leroux and Grégoire Sutre. On flatness for 2-dimensional vector addition systems with states. In *International Conference on Concurrency Theory (CONCUR)*. Springer Berlin Heidelberg, 2004.

- 19 Richard Mayr. Undecidability of weak bisimulation equivalence for 1-counter processes. In *International Colloquium on Automata, Languages and Programming (ICALP)*. Springer, 2003.
- 20 M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1 edition, 1967.
- 21 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 1978. URL: <http://www.sciencedirect.com/science/article/pii/0304397578900361>, doi:[https://doi.org/10.1016/0304-3975\(78\)90036-1](https://doi.org/10.1016/0304-3975(78)90036-1).
- 22 Louis E. Rosier and Hsu-Chun Yen. A multiparameter analysis of the boundedness problem for vector addition systems. In *International Symposium on Fundamentals of Computation Theory (FCT)*. Springer Berlin Heidelberg, 1985.
- 23 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.
- 24 Leslie G. Valiant. *Decision Procedures for Families of Deterministic Pushdown Automata*. PhD thesis, University of Warwick, 1973. URL: <http://wrap.warwick.ac.uk/34701/>.
- 25 Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag, 1999.
- 26 Tzeng Wen-Guey. On path equivalence of nondeterministic finite automata. *Information Processing Letters*, 1996. URL: <http://www.sciencedirect.com/science/article/pii/0020019096000397>, doi:[https://doi.org/10.1016/0020-0190\(96\)00039-7](https://doi.org/10.1016/0020-0190(96)00039-7).