Complexity of regular abstractions of one-counter languages

Mohamed Faouzi Atig
Uppsala University, Sweden
mohamed_faouzi.atig@it.uu.se

Dmitry Chistikov
MPI-SWS, Germany
dch@mpi-sws.org

Piotr Hofman *
LSV, CNRS & ENS Cachan, Université Paris-Saclay, France
piotr.hofman@lsv.ens-cachan.fr

K Narayan Kumar
Chennai Mathematical Institute, India
kumar@cmi.ac.in

Prakash Saivasan
Chennai Mathematical Institute, India
saivasan@cmi.ac.in

Georg Zetzsche †
LSV, CNRS & ENS Cachan, Université Paris-Saclay, France
zetzsche@lsv.fr

Abstract

We study the computational and descriptional complexity of the following transformation: Given a one-counter automaton (OCA) $\mathcal{A}$, construct a nondeterministic finite automaton (NFA) $\mathcal{B}$ that recognizes an abstraction of the language $\mathcal{L}(\mathcal{A})$: its (1) downward closure, (2) upward closure, or (3) Parikh image. For the Parikh image over a fixed alphabet and for the upward and downward closures, we find polynomial-time algorithms that compute such an NFA. For the Parikh image with the alphabet as part of the input, we find a quasi-polynomial time algorithm and prove a completeness result: we construct a sequence of OCA that admits a polynomial-time algorithm iff there is one for all OCA. For all three abstractions, it was previously unknown if appropriate NFA of sub-exponential size exist.

1. Introduction

The family of **one-counter languages** is an intermediate class between context-free and regular languages: it is strictly less expressive than the former and strictly more expressive than the latter. For example, the language $\{a^m b^m \mid m \geq 0\}$ is one-counter, but not regular, and the set of palindromes over the alphabet $\{a, b\}$ is context-free, but not one-counter. From the verification perspective, the corresponding class of automata, *one-counter automata* (OCA), can model some infinite-state phenomena with its ability to keep track of a non-negative integer counter, see, e.g., [10], [29, Section 5.1], and [3, Section 5.2].

* Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS

† Supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD).

Reasoning about OCA, however, is hardly an easy task. For example, checking whether two OCA accept some word in common is undecidable even in the deterministic case; for nondeterministic OCA, even language universality, as well as language equivalence, is undecidable. For deterministic OCA, equivalence is NL-complete; the proof of the membership in NL took 40 years [9, 35].

This lack of tractability suggests the study of finite-state abstractions for OCA. Such a transition is a recurrent theme in formal methods: features of programs beyond finite state are modeled with infinite-state systems (such as pushdown automata, counter systems, Petri nets, etc.), and then finite-state abstractions of these systems come as an important tool for analysis (see, e.g., [4–7, 15, 31, 34]). In our work, we focus on the following three **regular abstractions**, each capturing a specific feature of a language $\mathcal{L} \subseteq \Sigma^*$: • The *downward closure* of $\mathcal{L}$, denoted $\mathcal{L}{\downarrow}$, is the set of all subwords (subsequences) of all words $w \in \mathcal{L}$, i.e., the set of all words that can be obtained from words in $\mathcal{L}$ by removing some letters. The downward closure is always a superset of the original language, $\mathcal{L} \subseteq \mathcal{L}{\downarrow}$, and, moreover, a regular one, no matter what $\mathcal{L}$ is, by Higman's lemma [22].

• The *upward closure* of $\mathcal{L}$, denoted $\mathcal{L}{\uparrow}$, is the set of all superwords (supersequences) of all words $w \in \mathcal{L}$, i.e., the set of all words that can be obtained from words in $\mathcal{L}$ by inserting some letters. Similarly to $\mathcal{L}{\downarrow}$, the language $\mathcal{L}{\uparrow}$ satisfies $\mathcal{L} \subseteq \mathcal{L}{\uparrow}$ and is always regular.

• The *Parikh image* of $\mathcal{L}$, denoted $\psi(\mathcal{L})$, is the set of all vectors $v \in \mathbb{N}^{|\Sigma|}$, that count the number of occurrences of letters of Σ in words from $\mathcal{L}$. That is, suppose $\Sigma = \{a_1, \ldots, a_k\}$, then every word $w \in \mathcal{L}$ corresponds to a vector $\psi(w) = (v_1, \ldots, v_k)$ such that v_i is the number of occurrences of a_i in w. The set $\psi(\mathcal{L})$ is always a regular subset of $\mathbb{N}^{|\Sigma|}$ if $\mathcal{L}$ is context-free, by the Parikh theorem [33]. It has long been known that all three abstractions can be *effectively computed* for context-free languages (CFL), by the results of van Leeuwen [36] and Parikh [33]. Algorithms performing these tasks, as well as finite automata recognizing these abstractions, are now widely used as building blocks in the language-theoretic approach to verification. Specifically, computing upward and downward closures occurs as an ingredient in the analysis of systems communicating via shared memory, see, e.g., [4, 6, 7, 31]. As the recent paper [28] shows, for parameterized networks of such systems the decidability hinges on the ability to compute downward closures. The Parikh image as an abstraction in the verification of infinite-state systems has been used extensively; see, e.g., [1, 2, 5, 13, 15, 17, 20, 25, 34]. For pushdown systems, it is possible to construct a linear-sized existential Presburger formula that captures the Parikh image [37], which leads, for a variety of problems (see, e.g., [1, 13, 20, 25]), to algorithms that rely on deciding satisfiability for such formulas (which is in NP). Finite automata for Parikh images are used as intermediate representations, for example, in the analysis of multi-threaded programs [5, 15, 34] and in recent work on so-called availability languages [2].

Extending the scope of these three abstractions from CFL to other classes of languages has been a natural topic of interest. Effective constructions for the downward closure have been developed for Petri nets [19] and stacked counter automata [39]. The paper [38] gives a sufficient condition for a class of languages to have effective downward closures; this condition has since been applied to higher-order pushdown automata [21]. The effective regularity of the Parikh image is known for linear indexed languages [12], phase-bounded and scope-bounded multi-stack visibly pushdown languages [26, 27], and availability languages [2]. However, there are also negative results: for example, it is not possible to effectively compute the downward closure of languages recognized by lossy channels automata—this is a corollary of the fact that, for the set of reachable configurations of a lossy channel system, boundedness is undecidable [32].

Our contribution

We study the construction of nondeterministic finite automata (NFA) for $\mathcal{L}\downarrow$, $\mathcal{L}\uparrow$, and $\psi(\mathcal{L})$, if $\mathcal{L}$ is given as an OCA $\mathcal{A}$ with n states: $\mathcal{L} = \mathcal{L}(\mathcal{A})$. It turns out that for one-counter languages—a proper subclass of CFL—all three abstractions can be computed much more efficiently than for the entire class of CFL.

Upward and downward closures: We show, for OCA, how to construct NFA accepting $\mathcal{L}\uparrow$ and $\mathcal{L}\downarrow$ in polynomial time (Theorems 2 and 7). The construction for $\mathcal{L}\uparrow$ is straightforward, but the one for $\mathcal{L}\downarrow$ is involved and uses pumping-like techniques from automata theory.

These results are in contrast with the exponential lower bounds known for both closures in the case of CFL [36]: Several constructions for $\mathcal{L}\uparrow$ and $\mathcal{L}\downarrow$ have been proposed in the literature (see, e.g., [8, 11, 18, 36]), and the best in terms of the size of NFA are exponential, due to van Leeuwen [36] and Bachmeier, Luttenberger, and Schlund [8], respectively.

Parikh image: For OCA, the problem of constructing NFA for the Parikh image turns out to be quite tricky. While we were unable to solve the problem completely, we make significant progress towards its solution:

• For any fixed alphabet Σ we provide a complete solution: We find a polynomial-time algorithm that computes an NFA for $\psi(\mathcal{L}(\mathcal{A}))$ that has size $O(|\mathcal{A}|^{\mathrm{poly}(|\Sigma|)})$ (Theorem 8). Two key ingredients of this construction are a sophisticated version of a pumping lemma (Lemma 14; cf. a standard pumping lemma for one-counter languages, due to Latteux [30]) and the classic Carathéodory theorem for cones in a multi-dimensional space.

• We provide a quasi-polynomial solution to this problem in the general case: We find an algorithm that constructs a suitable NFA of size $O(|\Sigma|\cdot|\mathcal{A}|^{O(\log(|\mathcal{A}|))})$ (Theorem 22). This construction has two steps, both of which are of interest: in the first step we show, using a combination of local and global transformations on runs (Lemmas 20 and 21), that we may focus our attention on runs with at most polynomially many reversals. In the second step, which also works for pushdown automata, we turn the bound on reversals, using an argument with a flavour of Strahler numbers [16], into a logarithmic bound on the stack size of a pushdown system (Lemma 23).

• We prove a lower-bound type result (Theorem 24): We find a sequence of OCA $(\mathcal{H}_n)_{n\geq 1}$, where n denotes the number of states, over alphabets of growing size, that admits a polynomial-time algorithm for computing an NFA for the Parikh image if and only if there is such an algorithm for all OCA. Thus, the problem of transforming an arbitrary OCA $\mathcal{A}$ into an NFA for $\psi(\mathcal{L}(\mathcal{A}))$ is reduced to performing this transformation on $\mathcal{H}_n$, which enables us to call $\mathcal{H}_n$ *complete*. This result also has a counterpart referring to just the existence of NFA of polynomial size.

For the Parikh image of CFL, a number of constructions can be found in the literature as well; we refer the reader to the paper by Esparza et al. [14] for a survey and state-of-the-art results: exponential upper and lower bounds of the form $2^{\Theta(n)}$ on the size of NFA for $\psi(\mathcal{L})$.

Applications

Our results show that for OCA, unlike for pushdown systems, NFA representations of downward and upward closures and Parikh image (for fixed alphabet size) have efficient polynomial constructions. This suggests a possible way around standard NP procedures that handle existential Presburger formulas. This insight also leads to significant gains when abstractions are used in a nested manner, as illustrated by the following examples.

Consider a *network of pushdown systems* communicating via a shared memory. The reachability problem is undecidable in this setting. In [7] a restriction called stage-boundedness, generalizing context-boundedness, is explored. During a stage, the memory can be written to only by one system. Reachability along runs with at most k stages is decidable when all but one pushdown in the network are counters. The procedure in [7] uses NFA that accept upward and downward closures of one-counter languages; the polynomial-time algorithms developed in the present paper bring the complexity from NEXP down to NP for any network with a fixed number of components.

Availability expressions [23] extend regular expressions by an additional counting operator to express quantitative properties of behaviours. It uses a feature called *occurrence constraint* to impose a set of linear constraints on the number of occurrences of alphabet symbols in sub-expressions. As the paper [2] shows, the emptiness problem for availability expressions is decidable, and the algorithm involves nested calls to Parikh-image computation for OCA. Our quasi-polynomial time algorithm for the Parikh image brings the complexity from non-elementary down to 2EXP.

2. Preliminaries

2.1 One-counter automata

A *one-counter automaton* (*OCA*) $\mathcal{A}$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q a a finite set of states, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of final states. Σ is a finite alphabet and $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \{-1, 0, +1, \mathrm{z}\} \times Q$ is a set of transitions. Transitions $(p_1, a, s, p_2) \in \delta$ are classified as *incrementing*

$(s = +1)$, *decrementing* $(s = -1)$, *internal* $(s = 0)$, or *tests for zero* $(s = \mathrm{z})$. The *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, is its number of states, $|Q|$.

A *configuration* of an OCA is a pair that consists of a state and a (non-negative) counter value, i.e., $(q, n) \in Q \times \mathbb{N}$. A pair $(p_1, c_1) \in Q \times \mathbb{Z}$ may evolve to a pair $(p_2, c_2) \in Q \times \mathbb{Z}$ via a transition $t = (p_1, a, s, p_2) \in \delta$ iff either $s \in \{-1, 0, +1\}$ and $c_1 + s = c_2$, or $s = \mathrm{z}$ and $c_1 = c_2 = 0$. We denote this by $(p_1, c_1) \xrightarrow{t} (p_2, c_2)$.

Consider a sequence of the form $\pi = (p_0, c_0)$, t_1, (p_1, c_1), t_2, $\ldots$, t_m, (p_m, c_m) where $(p_i, c_i) \in Q \times \mathbb{Z}$ for $0 \le i \le m$ and, whenever $i > 0$, it also holds that $t_i \in \delta$ and $(p_{i-1}, c_{i-1}) \xrightarrow{t_i} (p_i, c_i)$. We say that π *induces* a word $w = a_1 a_2 \ldots a_m \in \Sigma^*$ where $a_i \in \Sigma \cup \{\varepsilon\}$ and $t_i = (p_{i-1}, a_i, s, p_i)$; we also say that the word w can be *read* or *observed* along the sequence π. We call the sequence π:

- a *quasi-run*, denoted $\pi = (p_0, c_0) \overset{w}{\Rightarrow}_{\mathcal{A}} (p_m, c_m)$, if none of t_i is a test for zero;
- a *run*, denoted $\pi = (p_0, c_0) \xrightarrow{w}_{\mathcal{A}} (p_m, c_m)$, if all $(p_i, c_i) \in Q \times \mathbb{N}$.

We abuse notation and write $\overset{w}{\Rightarrow}$ (resp. $\xrightarrow{w}$) to mean $\overset{w}{\Rightarrow}$ (resp. $\xrightarrow{w}_{\mathcal{A}}$) when it is clear from context. For $m = 0$, we also use this notation with $w = \varepsilon$. In addition, for any quasi-run π as above, the sequence of transitions $t_1, \ldots, t_m$ is called a *walk* from the state p_0 to the state p_m.

We will *concatenate* runs, quasi-runs, and walks, using the notation $\pi_1 \cdot \pi_2$ and sometimes dropping the dot. If π_2 is a walk and π_1 is a run, then $\pi_1 \cdot \pi_2$ will also denote a run. In this and other cases, we will often assume that the counter values in π_2 are picked or adjusted automatically to match the last configuration of π_1.

The number m is the *length* of π, denoted $|\pi|$; for a walk, its length is equal to the length of the sequence. All concepts and attributes naturally carry over from runs to walks and vice versa. Quasi-runs are not used until further sections; the semantics of OCA is defined just using runs.

A run $(p_0, c_0) \xrightarrow{w} (p_m, c_m)$ is called *accepting* in $\mathcal{A}$ if $(p_0, c_0) = (q_0, 0)$ where q_0 is the initial state of $\mathcal{A}$ and p_m is a final state of $\mathcal{A}$, i.e., $p_m \in F$. In such a case the word w is *accepted* by $\mathcal{A}$; the set of all accepted words is called the *language* of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$.

2.2 Regular abstractions

A *nondeterministic finite automaton with ε-transitions* (*NFA*) is a one-counter automaton where all transitions are tests for zero. Languages of the form $\mathcal{L}(\mathcal{N})$, where $\mathcal{N}$ is an NFA, are *regular*. If $\mathcal{A}$ is an OCA, then $\mathcal{L}(\mathcal{A})$ —a one-counter language— is not necessarily regular. In what follows, we consider three *regular abstractions* of (one-counter) languages: downward closures, upward closures, and Parikh-equivalent regular languages.

Let $w, w' \in \Sigma^*$. We say that a word w is a *subword* of a word w' if $w = a_1 \ldots a_n$ and there are $x_i \in \Sigma^*$, $1 \le i \le n{+}1$, such that $w' = x_1 a_1 x_2 a_2 \ldots x_n a_n x_{n+1}$. We write $w \preceq w'$ to indicate this. For any language $\mathcal{L} \subseteq \Sigma^*$, the *upward* and *downward closures* of $\mathcal{L}$ are the languages

$$\mathcal{L}{\uparrow} = \{w' \mid \exists w \in \mathcal{L}.\ w \preceq w'\} \quad \text{and}$$
$$\mathcal{L}{\downarrow} = \{w \mid \exists w' \in \mathcal{L}.\ w \preceq w'\}, \ \text{respectively.}$$

Any $w \in \Sigma^*$ defines a function $\psi(w) \colon \Sigma \to \mathbb{N}$, called the *Parikh image* of w (i.e., $\psi(w) \in \mathbb{N}^\Sigma$ for all $w \in \Sigma^*$). The value $\psi(w)(a)$ is the number of occurrences of a in w. The *Parikh image* of a language $\mathcal{L}$ is the following subset of $\mathbb{N}^\Sigma$:

$$\psi(\mathcal{L}) = \{\psi(w) \mid w \in \mathcal{L}\}.$$

In the sequel, we usually identify $\mathbb{N}^\Sigma$ and $\mathbb{N}^{|\Sigma|}$.

It follows from Higman's lemma [22] that, for any $\mathcal{L} \subseteq \Sigma^*$, the languages $\mathcal{L}{\uparrow}$ and $\mathcal{L}{\downarrow}$ are regular; since they abstract away some specifics of $\mathcal{L}$, they are regular abstractions of $\mathcal{L}$. For Parikh images, the situation is different: for example, unary languages $\mathcal{L} \subseteq \{a\}^*$ are essentially unaffected by the Parikh mapping ψ, but it is easy to find unary languages that are not even decidable, let alone regular. However, Parikh's theorem [33] states that if $\mathcal{L} \subseteq \Sigma^*$ is a context-free language, then there exists a regular language $\mathcal{R} \subseteq \Sigma^*$ that is *Parikh-equivalent* to $\mathcal{L}$, i.e., such that $\psi(\mathcal{L}) = \psi(\mathcal{R})$. Hence, such languages $\mathcal{R}$ are also regular abstractions of $\mathcal{L}$; since all one-counter languages are context-free, every OCA $\mathcal{A}$ has at least one regular language that is Parikh-equivalent to $\mathcal{L}(\mathcal{A})$.

2.3 Convention on OCA

To simplify the presentation, everywhere below we focus our attention on a sublcass of OCA that we call *simple one-counter automata* (*simple OCA*). A simple OCA is defined analogously to OCA and is different in the following aspects: (1) there are no zero tests, (2) there is a unique final state, $F = \{q_{\mathsf{final}}\}$, (3) only runs that start from the configuration $(q_{\mathsf{init}}, 0)$ and end at the configuration $(q_{\mathsf{final}}, 0)$ are considered *accepting*. The language of a simple OCA $\mathcal{A}$, also denoted $\mathcal{L}(\mathcal{A})$, is the set of words induced by accepting runs. We now show that this restriction is without loss of generality.

For an OCA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ and any $p, q \in Q$, define a simple OCA $\mathcal{A}^{p,q} \stackrel{\text{def}}{=} (Q, \Sigma, p, \delta^+, \{q\})$ where $\delta^+ \subseteq \delta$ is the set of all transitions in δ that are not tests for zero.

For any simple OCA $\mathcal{A}$, define a sequence of *approximants* $\mathcal{L}_{(n)}(\mathcal{A})$, $n \ge 0$: the language $\mathcal{L}_{(n)}(\mathcal{A})$ is the set of all words observed along runs of $\mathcal{A}$ from (q_0, n) to (q_{final}, n).

Lemma 1. *Let $K \in \mathbb{N}$ and $\Diamond \in \{\uparrow, \downarrow, \psi\}$ be an abstraction. Assume that there is a polynomial $g_\Diamond$ such that for any OCA $\mathcal{A}$ the following holds: for every $\mathcal{A}^{p,q}$ there is an NFA $\mathcal{B}_{p,q,\Diamond}$ such that $\Diamond\mathcal{L}(\mathcal{A}^{p,q}) \subseteq \mathcal{L}(\mathcal{B}_{p,q,\Diamond}) \subseteq \Diamond(\mathcal{L}_{(K)}(\mathcal{A}^{p,q}))$ and $|\mathcal{B}_{p,q,\Diamond}| \le g_\Diamond(|\mathcal{A}|)$. Then there is a polynomial f such that for any $\mathcal{A}$ there is an NFA $\mathcal{B}^\Diamond$ of size at most $f(|\mathcal{A}|, K)$ with $\mathcal{L}(\mathcal{B}^\Diamond) = \Diamond(\mathcal{L}(\mathcal{A}))$.*

Proof (sketch). We use the following two ideas: (i) to compute the abstraction of $\{w\}$ where $w = w_1 \cdot w_2$, it suffices to concatenate abstractions of $\{w_1\}$ and $\{w_2\}$; (ii) for any $K \in \mathbb{N}$, every run of $\mathcal{A}$ can be described as interleaving of runs below K and above K. The NFA $\mathcal{B}^\Diamond$ is constructed as follows: first encode counter values below K using states, and then insert NFA $\mathcal{B}_{p,q,\Diamond}$ in between (p, n) and (q, n). □

Restriction to simple OCA is now a consequence of Lemma 1 for $K = 0$.

3. Upward and Downward Closures

The standard argument used to bound the length of accepting runs of PDAs (or OCAs) can be adapted easily to show

Theorem 2. *There is a polynomial-time algorithm that takes as input an OCA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ and computes an NFA with $O(|\mathcal{A}|^3)$ states accepting $\mathcal{L}(\mathcal{A}){\uparrow}$.*

Next we show a polynomial time procedure that constructs an NFA accepting the downward closure of the language of any simple OCA. For pushdown automata the construction involves a necessary exponential blow-up. We sketch some observations that lead to our polynomial time construction.

Let $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ be a simple OCA and let $K = |Q|$. Consider any run ρ of $\mathcal{A}$ from a configuration (p, i) to a configuration (q, j). If the value of the counter increases (resp. decreases) by at least K in ρ then, it contains a segment that can be *pumped* (or iterated) to increase (resp. decrease) the value of the counter. Quite clearly, the word read along this iterated run will be a superword of word read along ρ. The following lemmas formalize this.

Lemma 3. *Let $(p,i) \xrightarrow{x} (q,j)$ with $j - i > K$. Then, there is an integer $k > 0$ such that for each $N \geq 0$ there is a run $(p,i) \xrightarrow{w=y_1.(y_2)^{N+1}.y_3} (p', j + N.k)$ with $x = y_1y_2y_3$.*

Lemma 4. *Let $(q', j') \xrightarrow{z} (p', i')$ with $j' - i' > K$. Then, there is an integer $k' > 0$ such that for every $N \geq 0$ there is a run $(q', j' + N.k') \xrightarrow{w=y_1(y_2)^{N+1}y_3} (p', i')$ with $z = y_1y_2y_3$.*

A consequence of these somewhat innocous lemmas is the following interesting fact: we can turn a triple consisting of two runs, where the first one increases the counter by at least K and the second one decreases the counter by at least K, and a quasi-run that connects them, into a real run provided we are content to read a superword along the way.

Lemma 5. *Let $(p,i) \xrightarrow{x} (q,j) \overset{y}{\Rrightarrow} (q',j') \xrightarrow{z} (p',i')$, with $j - i > K$ and $j' - i' > K$. Then, there is a run $(p,i) \xrightarrow{w} (p',i')$ such that $xyz \preceq w$.*

Interesting as this may be, this lemma still relies on the counter value being recorded exactly in all the three segments in its antecedent and we weaken this next.

Lemma 6. *Let $(p,i) \xrightarrow{x} (q,j)$,$(q,j) \xrightarrow{z} (p',i')$, with $j - i > K$ and $j' - i' > K$. Let there be a walk from q to q that reads y. Then, there is a run $(p,i) \xrightarrow{w} (p',i')$ such that $xyz \preceq w$.*

Proof. Let the given walk result in the quasi-run $(q, j) \overset{y}{\Rrightarrow} (q, j+d)$ (where d is the net effect of the walk on the counter, which may be positive or negative). Iterating this quasi-run m times yields a quasi-run $(q, j) \overset{y^m}{\Longrightarrow} (q, j + m.d)$, for any $m \geq 0$. Next, we use Lemma 3 to find a $k > 0$ such that for each $N > 0$ we have a run $(p,i) \xrightarrow{x_N} (q, j + N.k)$ with $x \preceq x_N$. Similarly, we use Lemma 4 to find a $k' > 0$ such that for each $N' > 0$ we have a run $(q, j + N'.k') \xrightarrow{y_{N'}} (p', i')$ with $y \preceq y_{N'}$.

Now, we pick m and N to be multiples of k' in such a way that $N.k + m.d > 0$. This can always be done since k is positive. Thus, $N.k + m.d = N'.k'$ with $N' > 0$. Now we try and combine the (quasi) runs $(p,i) \xrightarrow{x_N} (q, j + N.k)$, $(q, j + N.k) \overset{y^m}{\Longrightarrow} (q, j + N.k + m.d)$ and $(q, j + N'.k') \xrightarrow{y_{N'}} (p', i')$ to form a run. We are almost there, as $j + N.k + m.d = j + N'.k'$. However, it is not guaranteed that this combined quasi-run is actually a run as the value of the counter may turn negative in the segment $(q, j + N.k) \overset{y^m}{\Longrightarrow} (q, j + N.k + m.d)$. Let $-N''$ be the smallest value attained by the counter in this segment. Then by replacing N by $N + N''.k'$ and N' by $N' + N''.k$ we can manufacture a triple which actually yields a run (since the counter values are ≥ 0), completing the proof. $\square$

With this lemma in place we can now explain how to relax the usage of counters. Let us focus on runs that are interesting, that is, those in which the counter value exceeds K at some point. Any such run may be broken into 3 stages: the first stage where counter value starts at 0 and remains strictly below $K+1$, a second stage where it starts and ends at $K + 1$ and a last stage where the value begins at K and remains below K and ends at 0 (the 3 stages are connected by two transitions, an increment and a decrement). Suppose, we write the given accepting run as $(p, 0) \xrightarrow{w_1} (q, c) \xrightarrow{w_2} (r, 0)$ where (q, c) is a configuration in the second stage. If $a \in \Sigma$ is a letter that may be read in some transition on some walk from q to q. Then, w_1aw_2 is in $\mathcal{L}(\mathcal{A})\downarrow$. This is a direct consequence of the above lemma. It means that in the configurations in the middle stage we may freely read certain letters without bothering to update the counters. This turns out to be a crucial step in our construction. To turn this relaxation idea into a construction, the following seems a natural.

We make an equivalent, but expanded version of $\mathcal{A}$. This version has 3 copies of the state space: The first copy is used as long as the value of the counter stays below $K+1$ and on attaining this value the second copy is entered. The second copy simulates $\mathcal{A}$ exactly but nondeterministically chooses to enter third copy whenever the counter value is moves from $K + 1$ to K. The third copy simulates $\mathcal{A}$ but does not permit the counter value to exceed K. For every letter a and state q with a walk from q to q along which a is read on some transition, we add a self-loop transition to the state corresponding to q in the second copy that does not affect the counter and reads the letter a. This idea has two deficiencies: first, it is not clear how to define the transition from the second copy to the third copy, as that requires knowing that value of the counter is $K + 1$, and second, this is still an OCA (since the second copy simply faithfully simulates $\mathcal{A}$) and not an NFA.

Suppose we bound the value of the counter by some value U in the second stage. Then we can overcome both of these defects and construct a finite automaton. By using a slight generalization of Lemma 6, which allows for the simultaneous insertion of a number of walks (or by applying the Lemma iteratively), we can show that any word accepted by such a finite automaton lies $\mathcal{L}(\mathcal{A})\downarrow$. However, there is no guarantee that such an automaton will accept every word in $\mathcal{L}(\mathcal{A})\downarrow$. The second crucial point is that we are able to show that if $U \geq K^2 + K + 1$ then every word in $\mathcal{L}(\mathcal{A})$ is accepted by this 3 stage NFA. We show that for each accepting run ρ in $\mathcal{A}$ there is an accepting run in the NFA reading the same word. The proof is by a double induction, first on the maximum value attained by the counter and then on the number of times this value is attained along the run. Clearly, segments of the run where the value of the counter does not exceed $K^2 + K + 1$ can be simulated as is. We then show that whenever the counter value exceeds this number, we can find suitable segments whose net effect on the counter is 0 and which can be simulated using the self-loop transitions added to stage 2 (which do not modify the counters), reducing the maximum value of the counter along the run. Formalizing this gives:

Theorem 7. *There is a polynomial-time algorithm that takes as input a simple OCA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ and computes an NFA with $O(|\mathcal{A}|^3)$ states accepting $\mathcal{L}(\mathcal{A})\downarrow$.*

4. Parikh image: Fixed alphabet

The result of this section is the following theorem.

Theorem 8. *For any fixed alphabet Σ there is a polynomial-time algorithm that, given as input a one-counter automaton over Σ with n states, computes a Parikh-equivalent NFA.*

Note that in Theorem 8 the size of Σ is fixed. The theorem implies, in particular, that any one-counter automaton over Σ with n states has a Parikh-equivalent NFA of size $\mathrm{poly}_\Sigma(n)$, where poly_Σ is a polynomial of degree bounded by $f(|\Sigma|)$ for some computable function f.

We now provide the intuition behind the proof of Theorem 8. Our key technical contribution is capturing the structure of the Parikh image of the language $\mathcal{L}(\mathcal{A})$.

Recall that a set $A \subseteq \mathbb{N}^{|\Sigma|}$ is called *linear* if it is of the form $\mathrm{Lin}(b; P) \stackrel{\text{def}}{=} \{b + \lambda_1 p_1 + \ldots + \lambda_r p_r \mid \lambda_1, \ldots, \lambda_r \in \mathbb{N}, p_1, \ldots, p_r \in P\}$ for some vector $b \in \mathbb{N}^{|\Sigma|}$ and some finite set $P \subseteq \mathbb{N}^{|\Sigma|}$; this vector b is called the *base* and vectors $p \in P$ *periods.* A set $S \subseteq \mathbb{N}^d$ is called *semilinear* if it is a finite union of linear sets, $S = \cup_{i \in I} \mathrm{Lin}(b_i; P_i)$. Semilinear sets were introduced in the 1960s and have since received a lot of attention in formal language theory and its applications to verification. They are precisely the sets definable in Presburger arithmetic, the first-order theory of natural numbers with addition. Intuitively, semilinear sets are a multi-dimensional analogue of ultimately periodic sets in $\mathbb{N}$. For our purposes, of most importance is the following way of stating the Parikh theorem [33]: the Parikh image of any *context-free* language is a semilinear set; in particular, so is the Parikh image of any one-counter language, $\psi(\mathcal{L}(\mathcal{A}))$.

Our proof of Theorem 8 captures the periodic structure of this set $\psi(\mathcal{L}(\mathcal{A}))$. More precisely, we prove polynomial upper bounds on the number of linear sets in the semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$ and on the magnitude of periods and base vectors. Since converting such a semilinear representation into a polynomial-size NFA is easy, these bounds (subsection 4.1) entail the existence of an appropriate NFA. After this, we show how to compute, in time polynomial in $|\mathcal{A}|$, this semilinear representation from $\mathcal{A}$ (subsection 4.2).

4.1 Semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$

We now explain where the periodic structure of the set $\psi(\mathcal{L}(\mathcal{A}))$ comes from. Consider an individual accepting run π and assume that one can factorize it as $\pi = \rho \cdot \sigma \cdot \tau$ so that for any $k \geq 0$ the run $\rho \cdot \sigma^k \cdot \tau$ is also accepting. Values $k > 0$ correspond to pumping the run "up", and the value $k = 0$ corresponds to "unpumping" the infix σ. If we apply this "unpumping" to π several times (each time taking a new appropriate factorization of shorter and shorter runs), then the remaining part eventually becomes small (short). Its Parikh image will be a base vector, and the Parikh images of different infixes σ will be period vectors of a linear set in the semilinear representation.

However, this strategy faces several obstacles. First, the overall reasoning should work on the level of the whole automaton, as opposed to individual runs; this means that we need to rely on a form of a pumping lemma to factorize long runs appropriately. The pumping lemma for one-counter languages involves, instead of individual infixes σ, their pairs (σ_1, σ_2), so that the entire run factorizes as $\pi = \rho \cdot \sigma_1 \cdot \upsilon \cdot \sigma_2 \cdot \tau$, and runs $\pi = \rho \cdot \sigma_1^k \cdot \upsilon \cdot \sigma_2^k \cdot \tau$ are accepting for all $k \geq 0$. We incorporate this into our argument, talking about *split runs* (Definition 9). Here and below, for any run ζ, $\mathsf{effect}(\zeta)$ denotes the *effect* of ζ on the counter, i.e., the difference between the final and initial counter value along ζ.

Definition 9 (split run)**.** A *split run* is a pair of runs (σ_1, σ_2) such that $\mathsf{effect}(\sigma_1) \geq 0$ and $\mathsf{effect}(\sigma_2) \leq 0$.

Second and most importantly, it is crucial for the periodic structure of the set that individual "pumpings" and "unpumpings" can be performed independently. That is, suppose we can insert a copy of a sub-run σ into π, as above; also suppose we can remove from π some other sub-run σ'. What we need to ensure is that, after removing σ' from π, the obtained run π' will have the property that we can still insert a σ in it, as in the original run π. In general, of course, this does not have to be the case: even for finite-state machines, removal of loops can lead to removal of individual control states, which can, in turn, prevent the insertion of other loops (in our case the automaton also has a counter that must always stay non-negative). To deal with this phenomenon, we introduce the concept of "availability" (Definition 11). Essentially, a "pumpable" part of the run—i.e., a "split walk"—defines a *direction* (Definition 10); we say that a direction is *available* at the run π if it is possible to insert its copy into π. Thus, when doing "unpumping", we need to make sure that the set of available directions does not change: we call such unpumpings *safe* (Definition 13). We show that long accepting runs can always be safely unpumped (Lemma 14), which will lead us (Lemma 15) to the semilinear representation that we sketched at the beginning of this subsection.

We now describe the formalism behind our arguments.

Definition 10 (direction)**.** A *direction* is a pair of walks α and β, denoted $d = \langle \alpha, \beta \rangle$, such that:

- α begins and ends in the same control state,
- β begins and ends in the same control state,
- $0 < |\alpha| + |\beta| < n(2n^2 + 3)(n^3) + 1$,
- $0 \leq \mathsf{effect}(\alpha) \leq n^3$,
- $\mathsf{effect}(\alpha) + \mathsf{effect}(\beta) = 0$,
- if $\mathsf{effect}(\alpha) = 0$, then either $|\alpha| = 0$ or $|\beta| = 0$.

The direction is *of the first kind* if $\mathsf{effect}(\alpha) = 0$, and *of the second kind* otherwise.

One can think of a direction as a pair of short loops with zero total effect on the counter. Pairs of words induced by these loops are sometimes known as iterative pairs. Directions of the first kind are essentially just individual loops; in a direction of the second kind, the first loop increases and the second loop decreases the counter value (this restriction, however, only concerns the total effects of α and β; i.e., proper prefixes of α can have negative effects and proper prefixes of β positive effects). The condition that $\mathsf{effect}(\alpha) \leq n^3$ is a pure technicality and is only put in to make some auxiliary statements in the proof more laconic; in contrast, the upper bound $|\alpha| + |\beta| < n(2n^2 + 3)(n^3) + 1$ is crucial (although the choice of larger polynomial is, of course, possible, at the expense of an increase in the obtained upper bound on the size of NFA).

Definition 11 (availability of directions)**.** Suppose π is an accepting run. A direction $d = \langle \alpha, \beta \rangle$ is *available* at π if there exists a factorization $\pi = \pi_1 \cdot \pi_2 \cdot \pi_3$ such that $\pi' = \pi_1 \cdot \alpha \pi_2 \beta \cdot \pi_3$ is also an accepting run. We write $\pi + d$ to refer to π'.

Note that for a particular run π there can be more than one factorization of π into π_1, π_2, π_3 such that $\pi_1 \cdot \alpha \pi_2 \beta \cdot \pi_3$ is an accepting run. In such cases the direction d can be introduced at different points inside π. We only use the notation $\pi + d$ to refer to a single run π' obtained in this way, without specifying a particular factorization of π.

Denote by $\mathrm{avail}(\pi)$ the set of all directions available at π.

Lemma 12 (monotonicity of availability). *If π is an accepting run of an OCA and d is a direction available at π, then* $\text{avail}(\pi) \subseteq \text{avail}(\pi + d)$.

Definition 13 (unpumping). A run π' *can be unpumped* if there exist a run π and a direction d such that $\pi' = \pi + d$. If additionally $\text{avail}(\pi') = \text{avail}(\pi)$, then we say that π' *can be safely unpumped*.

Note that $\text{avail}(\pi')$ is always a superset of $\text{avail}(\pi)$ by Lemma 12. The key part of our argument is the proof that, indeed, every long run can be unpumped in a safe way:

Lemma 14 (safe unpumping lemma). *Every accepting run π' of $\mathcal{A}$ of length greater than $n^2((2n^2+3)(n^3))^3$ can be safely unpumped.*

Proof (sketch). We consider two cases, depending on whether the *height* (largest counter value) of π' exceeds a certain polynomial in n. The strategy of the proof is the same for both cases (although the details are somewhat different). We first show that sufficiently large parts (runs or split runs) of π' can always be unpumped (as in standard pumping arguments). We notice that for such an unpumping to be *unsafe*, it is necessary that the part contain a configuration whose removal shrinks the set of available directions—a reason for non-safety; this *important* configuration cannot appear anywhere else in π'. We prove that the total number of important configurations is at most $\text{poly}(n)$. As a result, if we divide the run π' into sufficiently many sufficiently large parts, at least one of the parts will contain no important configurations and, therefore, can be unpumped safely. □

Lemma 14 ensures that we faithfully represent the semilinear structure of the Parikh image of the entire language when we take Parikh images of short runs as base vectors and Parikh images of available directions as period vectors in the semilinear representation:

Lemma 15. *For any OCA $\mathcal{A}$, it holds that*

$$\psi(\mathcal{L}(\mathcal{A})) = \bigcup_{|\pi| \le s(n)} \text{Lin}(\psi(\pi); \psi(\text{avail}(\pi))), \qquad (1)$$

where the union is taken over all accepting runs of $\mathcal{A}$ of length at most $s(n) = n^2((2n^2+3)(n^3))^3$.

Finally, to keep the representation of the Parikh image small, we rely on a Carathéodory-style argument ensuring that the number of the linear sets in the semilinear representation needs to grow only polynomially in the size of the original automaton, while the sets of period vectors is also kept small. For this (and only this) part of the argument, we need the alphabet size, $|\Sigma|$, to be fixed.

4.2 Computing the semilinear representation

Lemma 15 suggests the following algorithm for computing the semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$. Enumerate all potential Parikh images v of small accepting runs π of $\mathcal{A}$ and all potential Parikh images of directions. For every v and for every tuple of $r \le |\Sigma|$ vectors $v_1, \ldots, v_r$ that could be Parikh images of directions in $\mathcal{A}$, check if $\mathcal{A}$ indeed has an accepting run π and directions $d_1, \ldots, d_r$ available at π such that $\psi(\pi) = v$ and $\psi(d)_i = v_i$ for all i (Parikh images of runs and directions are defined as Parikh images of words induced by them). Whenever the answer is yes, take a linear set $\text{Lin}(v; \{v_1, \ldots, v_r\})$ into the semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$. Terminate when all tuples $(v, v_1, \ldots, v_r)$ have been considered for all $r \le |\Sigma|$.

We now explain why this algorithm works in polynomial time. Recall that the size of the alphabet, $|\Sigma|$, is fixed. Note that by Definition 10 the total length of runs α_i and β_i in a direction $d_i = \langle \alpha_i, \beta_i \rangle$ is at most polynomial in n; similarly, equation (1) in Lemma 15 only refers to accepting runs π of polynomial length. Therefore, all the components of all potential Parikh images v and $v_1, \ldots, v_r$ are upper-bounded by polynomials in n of fixed degree. The number of such vectors in $\mathbb{N}^{|\Sigma|}$ is polynomial, and so is the number of appropriate tuples $(v, v_1, \ldots, v_r)$, $r \le |\Sigma|$. It now remains to argue that each tuple can be processed in polynomial time.

Lemma 16. *For every Σ there is a polynomial-time algorithm that, given a simple OCA $\mathcal{A}$ over Σ and vectors $v, v_1, \ldots, v_r \in \mathbb{N}^\Sigma$, $0 \le r \le |\Sigma|$, with all numbers written in unary, decides if $\mathcal{A}$ has an accepting run π and directions $d_1, \ldots, d_r \in \text{avail}(\pi)$ with $\psi(\pi) = v$ and $\psi(d_i) = v_i$ for all i.*

Lemma 16 is based on the following building block:

Lemma 17. *For every Σ there is a polynomial-time algorithm that, given a simple OCA $\mathcal{A}$ over Σ, two configurations (q_1, c_1) and (q_2, c_2) and a vector $v \in \mathbb{N}^\Sigma$ with all numbers written in unary, decides if $\mathcal{A}$ has a run $\pi = (q_1, c_1) \rightarrow (q_2, c_2)$ with $\psi(\pi) = v$.*

The algorithm of Lemma 17 solves a version of the Parikh membership problem for OCA. It constructs a multidimensional table by dynamic programming: for *all* pairs of configurations (q_1', c_1'), (q_2', c_2') with bounded c_1', c_2' and all vectors $v' \in \mathbb{N}^\Sigma$ of appropriate size, it keeps the information whether $\mathcal{A}$ has a run $(q_1', c_1') \rightarrow (q_2', c_2')$ with Parikh image v'.

This completes our description of how to compute, from an OCA $\mathcal{A}$, a semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$. Transforming this representation into an NFA is a simple exercise.

5. Parikh image: Unbounded alphabet

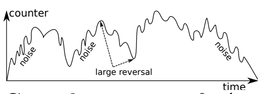
In this section we describe an algorithm to construct an NFA Parikh-equivalent to an OCA $\mathcal{A}$ without assumptions $|\Sigma|$. The NFA has at most $O(|\Sigma| K^{O(\log K)})$ states where $K = |\mathcal{A}|$, a significant improvement over $O(2^{\text{poly}(K, |\Sigma|)})$ for PDA.

We establish this result in two steps. In the first step, we show that we can focus our attention on computing Parikh-images of words recognized along *reversal bounded* runs. A reversal in a run occurs when the OCA switches to incrementing the counter after a non-empty sequence of decrements (and internal moves) or when it switches to decrementing the counter after a non-empty sequence of increments (and internal moves). For a number R, a run is R reversal bounded, if the number of reversals along the run is $\le R$. Let us use $\mathcal{L}_R(\mathcal{A})$ to denote the set of words accepted by $\mathcal{A}$ along runs with at most R reversals.

We construct a new polynomial size simple OCA from $\mathcal{A}$ and show that we can restrict our attention to runs with at most R reversals of this OCA, where R is a polynomial in K. In the second step, from any simple OCA $\mathcal{A}$ with K states and any integer R we construct an NFA of size $O(K^{O(log(R))})$ whose Parikh image is $\mathcal{L}_R(\mathcal{A})$. Combination of the two steps gives a $O(K^{O(logK)})$ construction.

5.1 Reversal bounding

We establish that, up to Parikh-image, it suffices to consider runs with $2K^2 + K$ reversals. We use two constructions: one that eliminates *large* reversals (think of a waveform) and another that eliminates *small* reversals (think of the noise on a noisy waveform). For the large reversals, the idea used is the following: we can reorder the transitions used along a run, hence preserving Parikh-image, to turn it into one with few large reversals (a noisy waveform with few reversals). The key idea used is to move each simple cycle at state q with a positive (resp. negative) effect on the counter to the first (resp. last) occurrence of the state along the run. To eliminate the smaller reversals (noise), the idea is to maintain the changes to the counter in the state and transfer it only when necessary to the counter to avoid unnecessary reversals.



Consider a run of $\mathcal{A}$ starting at a configuration (p, c) and ending at some configuration (q, d) such that the value of the counter e in any intermediate configuration satisfies $c - D \leq e \leq c + D$ (where D is some positive integer). We refer to such a run as an *D-band run.* Reversals along such a run are not important and we get rid of them by maintaining the (bounded) changes to the counter within the state.

We construct a simple OCA $\mathcal{A}[D]$ as follows: its states are $Q \cup Q_1 \cup Q_2$ where $Q_1 = Q \times [-D, D]$ and $Q_2 = [-D, D] \times Q$. All transitions of $\mathcal{A}$ are transitions of $\mathcal{A}[D]$ as well and thus using Q it can simulate any run of $\mathcal{A}$ faithfully. From any state $q \in Q$ the automaton may move nondeterministically to $(q, 0)$ in Q_1. The states in Q_1 are used to simulate D-band runs of $\mathcal{A}$ without altering the counter and by keeping track of the net change to the counter in the second component of the state. From a state (q, j) in Q_1, $\mathcal{A}[D]$ is allowed to nondeterministically move to (j, q) indicating that it will now transfer the (positive or negative) value j to the counter. After completing the transfer it reaches a state $(0, q)$ from where it can enter the state q via an internal move to continue the simulation of $\mathcal{A}$.

Observe that there are no reversals in the simulation and it involves only increments (if $d > c$) or only decrements (if $d < c$). Actually this automaton $\mathcal{A}[D]$ does even better. Concatenation of D-band runs is often not an D-band run but the idea of reversal free simulation extends to certain concatenations. We say that a run $(p_0, c_0) \xrightarrow{w} (p_n, c_n)$ is an increasing (resp. decreasing) *iterated D-band run* if it can be decomposed as

$$(p_0, c_0) \xrightarrow{w_1} (p_1, c_1) \xrightarrow{w_2} \ldots (p_{n-1}, c_{n-1}) \xrightarrow{w_n} (p_n, c_n)$$

where each $(p_i, c_i) \xrightarrow{w_{i+1}} (p_{i+1}, c_{i+1})$ is an D-band run and $c_i \leq c_{i+1}$ (resp. $c_i \geq c_{i+1}$). We say it is an iterated D-band run if it is an increasing or decreasing iterated D-band run.

Lemma 18. *Let $(p, c) \xrightarrow{w} (q, d)$ be an increasing (resp. decreasing) D-band run in $\mathcal{A}$. Then, there is a run $(p, c) \xrightarrow{w} (q, d)$ in $\mathcal{A}[D]$ along which the counter value is never decremented (resp. incremented).*

While clearly $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}[D])$, the converse is not in general true as along a run of $\mathcal{A}[D]$ the real value of the counter, i.e. the current value of the counter plus the offset available in the state, may be negative, leading to runs that are not simulations of runs of $\mathcal{A}$. The trick that helps us get around this is to relate runs of $\mathcal{A}[D]$ to $\mathcal{A}$ with a shift in counter values. The following lemma summarizes this shifted relationship:

Lemma 19. *Let $p, q \in Q$. If $(p, 0) \xrightarrow{w} (q, 0)$ is a run in $\mathcal{A}[D]$ then $(p, D) \xrightarrow{w} (q, D)$ is a run in $\mathcal{A}$.*

With these two lemmas we have enough information about $\mathcal{A}[D]$ and its relationship with $\mathcal{A}$. We need a bit more terminology to proceed. We say that a run of is an $D_{\leq}$ run (resp. $D_{\geq}$ run) if the value of the counter is bounded from above (resp. below) by D in every configuration encountered along the run. We say that a run of $\mathcal{A}$ is an $D_{>}$ run if it is of the form $(p, D) \xrightarrow{w} (q, D)$, it has at least 3 configurations and the value of the counter at every configuration other than the first and last is $> D$. Consider any run from a configuration $(p, 0)$ to $(q, 0)$ in $\mathcal{A}$. Once we identify the maximal $D_{>}$ subruns, what is left is a collection of $D_{\leq}$ subruns.

Let $\rho = (p, c) \xrightarrow{w} (q, d)$ be a run of $\mathcal{A}$ with $c, d \leq D$. If ρ is a $D_{\leq}$ run then its *D-decomposition* is ρ. Otherwise, its D-decomposition is given by a sequence of runs $\rho_0, \rho'_0, \rho_1, \rho'_1 \ldots \rho'_{n-1}, \rho_n$ with $\rho = \rho_0.\rho'_0.\rho_1.\rho'_1 \ldots .\rho'_{n-1}.\rho_n$, where each ρ_i is a $D_{\leq}$ run and each ρ'_i is a $D_{>}$ run for $0 \leq i \leq n$. Notice, that some of the ρ_i's may be trivial. Since the $D_{>}$ subruns are uniquely identified this definition is unambiguous. We refer to the ρ'_i's (resp. ρ_is) as the $D_{>}$ (resp. $D_{\leq}$) components of ρ.

Observe that the $D_{\leq}$ runs of $\mathcal{A}$ can be easily simulated by an NFA. Thus we may focus on transforming the $D_{>}$ runs, preserving just the Parikh-image, into a suitable form. For $D, M \in \mathbb{N}$, we say that a $D_{>}$ run ρ is a *(D, M)-good run* (think noisy waveform with few reversals) if there are runs $\sigma_1, \sigma_2 \ldots, \sigma_n, \sigma_{n+1}$ and iterated D-band runs $\rho_1, \rho_2, \ldots, \rho_n$ such that $\rho = \sigma_1 \rho_1 \sigma_2 \rho_2 \ldots \sigma_n \rho_n \sigma_{n+1}$ and $|\sigma_1| + \ldots + |\sigma_{n+1}| + 2.n \leq M$. Using Lemma 18 and that it is a $D_{>}$ run we show

Lemma 20. *Let $(p, D) \xrightarrow{w} (q, D)$ be an (D, M)-good run of $\mathcal{A}$. Then, there is a run $(p, 0) \xrightarrow{w} (q, 0)$ in $\mathcal{A}[D]$ with atmost M reversals.*

So far we have not used the fact that we can ignore the ordering of the letters read along a run (since we are only interested in the Parikh-image of $\mathcal{L}(\mathcal{A})$). We show that for any run ρ of $\mathcal{A}$ we may find another run ρ' of $\mathcal{A}$, that is equivalent up to Parikh-image, such that every $D_{>}$ component in the D-decomposition of ρ' is (D, M)-good, where M and D are polynomially related to K.

We fix $D = K$ in what follows. We take $M = 2K^2 + K$ for reasons that will become clear soon. We focus our attention on some $D_{>}$ component ξ of ρ which is not (D, M)-good. Let $X \subseteq Q$ be the set of states of Q that occur in at least two different configurations along ξ. For each of the states in X we identify the configuration along ξ where it occurs for the very first time and the configuration where it occurs for the last time. There are at most $2|X| (\leq 2K)$ such configurations and these decompose the run ξ into a concatenation of $2|X| + 1 (\leq 2K + 1)$ runs $\xi = \xi_1.\xi_2 \ldots \xi_m$ where $\xi_i, 1 < i < m$ is a segment connecting two such configurations. Now, suppose one of these ξ_i's has length K or more. Then it must contain a sub-run $(p, c) \rightarrow (p, d)$ with at most K moves, for some $p \in X$ (so, this is necessarily a K-band run). If $d - c \geq 0$ (resp. $d - c < 0$), then we *transfer* this subrun from its current position to the first occurrence (resp. last occurrence) of p in the run. This still leaves a valid run ξ' since ξ begins with a K as counter value and $|\xi_i| \leq K$. Moreover ξ and ξ' are equivalent upto Parikh-image.

If this ξ' continues to be a $K_{>}$ run then we again examine if it is (K, M)-good and otherwise, repeat the operation described above. As we proceed, we continue to accumulate a increasing iterated K-band run at the first occurrence of each state and decreasing iterated K-band run at the last occurrence of each state. We also ensure that in each iteration we only pick a segment that does NOT appear in these $2|X|$ iterated K-bands. Thus, these iterations will stop when either the segments outside the iterated K-bands are all of length $< K$ and we cannot find any suitable segment to transfer, or when the resulting run is no longer a $K_>$ run. In the first case, we must necessarily have a $(K, 2K^2 + K)$-good run. In the latter case, the resulting run decomposes as usual in $K_\leq$ and $K_>$ components, and we have that every $K_>$ component is strictly shorter than ξ, allowing us to use an inductive argument to prove the following:

Lemma 21. *Let $\rho = (p, 0) \xrightarrow{w} (q, 0)$ be any run in $\mathcal{A}$. Then, there is a run $\rho' = (p, 0) \xrightarrow{w'} (q, 0)$ of $\mathcal{A}$ with $\psi(w) = \psi(w')$ such that every $K_>$ component ξ in the canonical decomposition of ρ' is $(K, 2K^2 + K)$-good.*

Let $\mathcal{B}^{pq}$, $p, q \in Q$, be NFA Parikh-equivalent to $\mathcal{L}_{2K^2+K}(\mathcal{A}[K]^{p,q})$ where $A[K]^{pq}$ is $\mathcal{A}[K]$ with p as the only initial and q as the only final state. As a consequence of Lemmas 21,20 and 19, we can obtain an NFA $\mathcal{B}$ such that $\psi(\mathcal{L}(\mathcal{B})) = \psi(\mathcal{L}(\mathcal{A}))$. The number of states in the automaton $\mathcal{B}$ is $\sum_{p,q \in Q} |\mathcal{B}^{pq}| + K^2$. What remains to be settled is the size of the automata $\mathcal{B}^{pq}$. This problem is solved in the next subsection and the solution (see Lemma 23) implies that that the size of $\mathcal{B}^{pq}$ is bounded by $O(|\Sigma| K^{O(logK)})$. Thus we have

Theorem 22. *There is an algorithm, which given an OCA with K states and alphabet Σ, constructs a Parikh-equivalent NFA with $O(|\Sigma|.K^{O(\log K)})$ states.*

5.2 Parikh image under reversal bounds

Here we show that, for an OCA $\mathcal{A}$, with K states and whose alphabet is Σ, and any $R \in \mathbb{N}$, an NFA Parikh-equivalent to $\mathcal{L}_R(\mathcal{A})$ can be constructed with size $O(|\Sigma|.K^{O(logK)})$. As a matter of fact, this construction works even for pushdown systems and not just OCAs.

Let $\mathcal{A}$ be a simple OCA. It will be beneficial to think of the counter as a stack with a single letter alphabet, with pushes for increments and pops for decrements. Then, in any run from $(p, 0)$ to $(q, 0)$, we may relate an increment move uniquely with its *corresponding* decrement move, the pop that removes the value inserted by this push.

Now, consider a *one reversal run* ρ of $\mathcal{A}$ from say $(p, 0)$ to $(q, 0)$ involving two phases, a first phase ρ_1 with no decrement moves and a second phase ρ_2 with no increment moves. Such a run can be simulated, up to equivalent Parikh image (i.e. upto reordering of the letters read along the run) by an NFA as follows: simultaneously simulate the first phase (ρ_1) from the source and the second phase, in reverse order (ρ_2^{rev}), from the target. (The simulation of ρ_2^{rev} uses the transitions in the *opposite* direction, moving from the target of the transition to the source of the transition). The simulation matches increment moves of ρ_1 against decrement moves in ρ_2^{rev} (more precisely, matching the ith increment ρ_1 with the ith decrement in ρ_2^{rev}) while carrying out moves that do not alter the counters independently in both directions. The simulation terminates (or potentially terminates) when a common state, signifying the boundary between ρ_1 and ρ_2 is reached from both ends.

The state space of such an NFA will need pairs of states from Q, to maintain the current state reached by the forward and backward simulations. Since, only one letter of the input can be read in each move, we will also need two moves to simulate a matched increment and decrement and will need states of the form $Q \times Q \times \Sigma$ for the intermediate state that lies between the two moves.

Unfortunately, such a naive simulation would not work if the run had more *reversals*. For then the ith increment in the simulation from the left need not necessarily correspond to the ith decrement in the reverse simulation from the right. In this case, the run ρ can be written as follows:

$$
\begin{aligned}
(p, 0)\rho_1(p_1, c) &\xrightarrow{\tau_1} (p_1', c+1)\rho_3(p_2', c+1) \\
&\xrightarrow{\tau_2} (p_2, c)\rho_4(q_1, c)\rho_5(q, 0)
\end{aligned}
$$

where, the increment τ_1 corresponds to the decrement τ_2 and all the increments in ρ_1 are exactly matched by decrements in ρ_5. Notice that the increments in the run ρ_3 are exactly matched by the decrements in that run and similarly for ρ_4. Thus, to simulate such a well-matched run from p to q, after simulating ρ_1 and ρ_5^{rev} simultaneously matching corresponding increments and decrements, and reaching the state p_1 on the left and q_1 on the right, we can choose to now simulate matching runs from p_1 to p_2 and from p_2 to q_1 (for some p_2). Our idea is to choose one of these pairs and simulate it first, storing the other in a stack. We call such pairs *obligations*. The simulation of the chosen obligation may produce further such obligations which are also stored in the stack. The simulation of an obligation succeeds when the state reached from the left and right simulations are identical, and at this point we we may choose to close this simulation and pick up the next obligation from the stack or continue simulating the current pair further. The entire simulation terminates when no obligations are left. Thus, to go from a single reversal case to the general case, we have introduced a stack into which states of the NFA used for the single reversal case are stored. This can be formalized to show that the resulting PDA is Parikh-equivalent to $\mathcal{A}$.

But a little more analysis shows that there is a simulating run where the height of the stack is bounded by $log(R)$ where R is the number of reversals in the original run. Thus, to simulate all runs of $\mathcal{A}$ with at most R reversals, we may bound the stack height of the PDA by $log(R)$.

We show that if the stack height is h then we can choose to simulate only runs with at most $2^{log(R)-h}$ reversals for the obligation on hand. Once we show this, notice that when $h = log(R)$ we only need to simulate runs with 1 reversal which can be done without any further obligations being generated. Thus, the overall height of the stack is bounded by $log(R)$. Now, we explain why the claim made above holds. Clearly it holds initially when $h = 0$. Inductively, whenever we split an obligation, we choose the obligation with fewer reversals to simulate first, pushing the other obligation onto the stack. Notice that this obligation with fewer reversals is guaranteed to contain at most half the number of reversals of the current obligation (which is being split). Thus, whenever the stack height increases by 1, the number of reversals to be explored in the current obligation falls at least by half as required. On the other hand, an obligation (p, q) that lies in the stack at position h from the bottom, was placed there while executing (earlier) an obligation (p', q') that only required 2^{k-h+1} reversals. Since the obligation (p, q) contributes only a part of the obligation (p', q'), its number of reversals is also bounded by 2^{k-h+1}. And when (p, q) is removed from the stack for simulation, the stack height is $h - 1$. Thus, the invariant is maintained. Once we have this bound on the stack, for a given R, we can simulate it by an exponentially large NFA. This yields the following lemma:

Lemma 23. *There is a procedure that takes a simple OCA $\mathcal{A}$ with K states and whose alphabet is Σ, and a number*

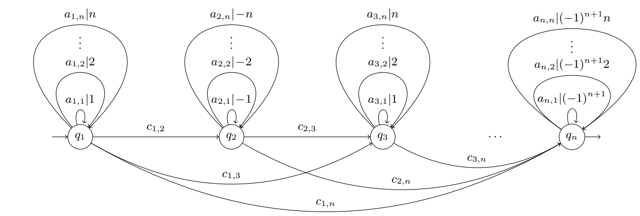


Figure 1. The one-counter automaton $\mathcal{H}_n$

$R \in \mathbb{N}$ and returns an NFA Parikh-equivalent to $\mathcal{L}_R(\mathcal{A})$ of size $O(|\Sigma|.(RK)^{O(log(R))})$.

5.3 Completeness result

In this section, we present a simple sequence of OCA that is complete with respect to small Parikh-equivalent NFAs. This means, if the OCA in this sequence have polynomial-size Parikh-equivalent NFAs, then all OCA have polynomial-size Parikh-equivalent NFAs.

It will be convenient to slightly extend the definition of OCA. An *extended OCA* is defined as an OCA, but in its transition (p, a, s, q), the entry s can assume any integer (in addition to z). Of course here, the number of states is not an appropriate measure of size. Therefore, the *size* of a transition $t = (p, a, s, q)$ of $\mathcal{A}$ is $|t| = \max(0, |s| - 1)$ if $s \in \mathbb{Z}$ and 0 if $s = z$. If $\mathcal{A}$ has n states, then we define its *size* is $|\mathcal{A}| = n + \sum_{t\in\delta} |t|$. Given an extended OCA of size n, one can clearly construct an equivalent OCA with n states. Furthermore, if one consideres an (ordinary) OCA as an extended OCA, then its size is the number of states.

The complete sequence $(\mathcal{H}_n)_{n\geq 1}$ of automata consists of extended OCA and is illustrated in Figure 1. The automaton $\mathcal{H}_n$ has n states, $q_1, \dots, q_n$. On each q_i and for each $k \in [1, n]$, there is a loop reading $a_{i,k}$ and adding $(-1)^{i+1} \cdot k$ to the counter. Moreover, for $i, j \in [1, n]$ with $i < j$, there is a transition reading $c_{i,j}$ that does not use the counter. For each $k \in [1, n]$, $\mathcal{H}_n$ has n transitions of size $k-1$. Since it has n states, this results in a size of $n+\sum_{k=1}^{n}(k-1) = \frac{1}{2}n(n+1)$.

The result of this section is the following.

Theorem 24. *There are polynomials p and q such that the following holds.*

1. *If for each n, there is a Parikh-equivalent NFA for $\mathcal{H}_n$ with $h(n)$ states, then for every OCA of size n, there is a Parikh-equivalent NFA with at most $q(h(p(n)))$ states.*
2. *If there is an algorithm that computes a Parikh-equivalent NFA for $\mathcal{H}_n$ in time $O(h(n))$, then one can compute a Parikh-equivalent NFA for arbitrary OCA in time $O(q(h(p(n))))$.*

Explicitly, we only prove the first statement and keep the analogous statements in terms of time complexity implicit. Our proof consists of three steps (Lemmas 25, 27, and 28). Intuitively, each of them is one algorithmic step one has to carry out when constructing a Parikh-equivalent NFA for a given OCA.

For the first step in our proof, we need some terminology. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be an extended OCA. Recall that a word $(p_1, a_1, s_1, p_1') \cdots (p_n, a_n, s_n, p_n')$ over δ is called a *walk* if $p_i' = p_{i+1}$ for every $i \in [1, n-1]$. The walk u is called a p_1-*cycle* (or just *cycle*) if $p_n' = p_1$. If, in addition, $i \neq j$ implies $p_i \neq p_j$, then u is called *simple*. A cycle as above is called *proper* if there is some $i \in [1, n]$ with $p_i \neq p_1$. We say that $\mathcal{A}$ is *acyclic* if it has no proper cycles, i.e. if all cycles consist solely of loops. Equivalently, an OCA is acyclic if there is a partial order $\leq$ on the set of states such that if a transition leads from a state p to q, then $p \leq q$.

A transition (p, a, s, q) is called *positive* (*negative*) if $s > 0$ ($s < 0$). We say that a walk *contains k reversals* if it has a scattered subword of length $k + 1$ in which positive and negative transitions alternate. An (extended) OCA is called *(r-)reversal-bounded* if none of its walks contains $r + 1$ reversals. Observe that an acyclic (extended) OCA is reversal-bounded if and only if on each state, there are either no positive transitions or no negative transitions. We call such automata *RBA* (*reversal-bounded acyclic automata*).

Recall that we have seen in section 5.1 that constructing Parikh-equivalent NFAs essentially reduces to the case of reversal-bounded simple OCA. The first construction here takes a reversal-bounded automaton and decomposes it into an RBA and a regular substitution. This means, if we can find Parikh-equivalent OCA for RBAs, we can do so for arbitrary OCA: Given an NFA for the RBA, we can replace every letter by the finite automaton specified by the substitution. Here, the *size* of the substitution σ is the maximal number of states of an automaton specified for a language $\sigma(a)$, $a \in \Sigma$.

Lemma 25. *Given an r-reversal-bounded simple OCA $\mathcal{A}$ of size n, one can construct an RBA $\mathcal{B}$ of size $6n^5(r + 1)$ and a regular substitution σ of size $n(n + 1)$ such that $\psi(\sigma(\mathcal{L}(\mathcal{B}))) = \psi(\mathcal{L}(\mathcal{A}))$.*

We prove this lemma by showing that runs of reversal-bounded simple OCA can be 'flattened': Each run can be turned into one with Parikh-equivalent input that consists of a skeleton of polynomial length in which simple cycles are inserted flat, i.e. without nesting them. The RBA $\mathcal{B}$ simulates the skeleton and has self-loop which are replaced by σ with a regular language that simulates simple cycles.

In the next construction of our proof (Lemma 27), we employ a combinatorial fact. A *Dyck sequence* is a sequence $x_1, \dots, x_n \in \mathbb{Z}$ such that $\sum_{i=1}^{k} x_i \geq 0$ for every $k \in [1, n]$. We call a subset $I \subseteq [1, n]$ *removable* if removing all x_i, $i \in I$, from the sequence yields again a Dyck sequence and $\sum_{i\in I} x_i = \sum_{i=1}^{n} x_i$. We call the sequence *$r$-reversal-bounded* if there are at most r alternations between positive numbers and negative numbers.

Lemma 26. *Let $N \geq 0$ and $x_1, \dots, x_n$ be an r-reversal-bounded Dyck sequence with $x_i \in [-N, N]$ for each $i \in [1, n]$ such that $\sum_{i=1} x_i \in [0, N]$. Then it has a removable subset $I \subseteq [1, n]$ with $|I| \leq 2r(2N^2 + N)$.*

We now want to make sure that the self-loops on our states are the only transitions that use the counter. Note that this is a feature of $\mathcal{H}_n$. An RBA is said to be *loop-counting* if its loops are the only transitions that use the counter, i.e. all other transitions (p, a, s, q) have $s = 0$.

Lemma 27. *Given an RBA $\mathcal{A}$, one can construct a loop-counting RBA $\mathcal{B}$ of polynomial size such that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}_{(K)}(\mathcal{A})$ for a polynomially bounded K.*

Here, the idea is to add a counter that tracks the counter actions of non-loop transitions. However, in order to show that the resulting automaton can still simulate all runs while respecting its own counter (i.e. it has to reach zero in the end and cannot drop belo zero), we use Lemma 26. It allows us to 'switch' parts of the run so as not to use $\mathcal{B}$'s counter, but the internal one in the state. Note that according to

Lemma 1, it suffices to construct Parikh-equivalent NFAs that fulfill the approximation relation in the lemma here.

We are now ready to reduce to the NFA $\mathcal{H}_n$.

Lemma 28. *Given a loop-counting RBA $\mathcal{A}$ of size n, one can construct a regular substitution σ of size at most 2 such that $\psi(\sigma(\mathcal{L}(\mathcal{H}_{2n+2}))) = \psi(\mathcal{L}(\mathcal{A}))$.*

Here, roughly speaking, we embed the partial order on the set of states into the one in $\mathcal{H}_{2n+2}$. Then, the substitution σ replaces each symbol in $\mathcal{H}_{2n+2}$ by the outputs of the correponding transition in $\mathcal{A}$. Here, we have to deal with the fact that in $\mathcal{A}$, there might be loops that do not read input, but those do not exist in $\mathcal{H}_{2n+2}$. However, we can replace the symbols on non-loops by regular languages so as to produce the output of their neighboring loops.

References

[1] P. A. Abdulla, M. F. Atig, and J. Cederberg. Analysis of message passing programs using SMT-solvers. In *ATVA'13*, volume 8172 of *LNCS*, pages 272–286, 2013.

[2] P. A. Abdulla, M. F. Atig, R. Meyer, and M. S. Salehi. What's decidable about availability languages? In *FSTTCS'15*, volume 45 of *LIPIcs*, pages 192–205, 2015.

[3] R. Alur and P. Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *POPL'11*, pages 599–610, 2011.

[4] M. F. Atig, A. Bouajjani, and T. Touili. On the reachability analysis of acyclic networks of pushdown systems. In *CONCUR'08*, volume 5201 of *LNCS*, pages 356–371, 2008.

[5] M. F. Atig, A. Bouajjani, and T. Touili. Analyzing asynchronous programs with preemption. In *FSTTCS'08*, volume 2 of *LIPIcs*, pages 37–48, 2008.

[6] M. F. Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *Logical Methods in Computer Science*, 7(4), 2011.

[7] M. F. Atig, A. Bouajjani, K. N. Kumar, and P. Saivasan. On bounded reachability analysis of shared memory systems. In *FSTTCS'14*, volume 29 of *LIPIcs*, pages 611–623, 2014.

[8] G. Bachmeier, M. Luttenberger, and M. Schlund. Finite automata for the sub- and superword closure of CFLs: Descriptional and computational complexity. In *LATA'15*, volume 8977 of *LNCS*, pages 473–485, 2015.

[9] S. Böhm, S. Göller, and P. Jancar. Equivalence of deterministic one-counter automata is NL-complete. In *STOC'13*, pages 131–140, 2013. .

[10] C. Chitic and D. Rosu. On validation of XML streams using finite state machines. In *WebDB'04*, pages 85–90, 2004.

[11] B. Courcelle. On constructing obstruction sets of words. *Bulletin of the EATCS*, 44:178–186, 1991.

[12] J. Duske and R. Parchmann. Linear indexed languages. *Theor. Comput. Sci.*, 32:47–60, 1984. .

[13] J. Esparza and P. Ganty. Complexity of pattern-based verification for multithreaded programs. In *POPL'11*, pages 499–510, 2011.

[14] J. Esparza, P. Ganty, S. Kiefer, and M. Luttenberger. Parikh's theorem: A simple and direct automaton construction. *Inf. Process. Lett.*, 111(12):614–619, 2011.

[15] J. Esparza, P. Ganty, and T. Poch. Pattern-based verification for multithreaded programs. *ACM Trans. Program. Lang. Syst.*, 36(3):9:1–9:29, 2014.

[16] J. Esparza, M. Luttenberger, and M. Schlund. A brief history of Strahler numbers. In *LATA'14*, volume 8370 of *Lecture Notes in Computer Science*, pages 1–13, 2014. .

[17] P. Ganty and R. Majumdar. Algorithmic verification of asynchronous programs. *ACM Trans. Program. Lang. Syst.*, 34(1):6:1–6:48, May 2012. ISSN 0164-0925.

[18] H. Gruber, M. Holzer, and M. Kutrib. More on the size of higman-haines sets: Effective constructions. In *MCU'07*, volume 4664 of *LNCS*, pages 193–204, 2007.

[19] P. Habermehl, R. Meyer, and H. Wimmel. The downward-closure of Petri net languages. In *ICALP'10*, volume 6199 of *LNCS*, pages 466–477, 2010.

[20] M. Hague and A. W. Lin. Synchronisation- and reversal-bounded analysis of multithreaded programs with counters. In *CAV'12*, volume 7358 of *LNCS*, pages 260–276, 2012.

[21] M. Hague, J. Kochems, and C. L. Ong. Unboundedness and downward closures of higher-order pushdown automata. In *POPL'16*, pages 151–163, 2016.

[22] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7), 1952.

[23] J. Hoenicke, R. Meyer, and E. Olderog. Kleene, rabin, and scott are available. In *CONCUR'10*, pages 462–477, 2010.

[24] D. T. Huynh. The complexity of equivalence problems for commutative grammars. *Information and Control*, 66(1/2): 103–121, 1985. .

[25] E. Kopczynski and A. W. To. Parikh images of grammars: Complexity and applications. In *LICS'10*, pages 80–89, 2010. .

[26] S. La Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *LICS'07*, pages 161–170, 2007. .

[27] S. La Torre, M. Napoli, and G. Parlato. Scope-bounded pushdown languages. In *DLT'14*, volume 8633 of *Lecture Notes in Computer Science*, pages 116–128, 2014. .

[28] S. La Torre, A. Muscholl, and I. Walukiewicz. Safety of parametrized asynchronous shared-memory systems is almost always decidable. In *CONCUR'15*, volume 42 of *LIPIcs*, pages 72–84, 2015.

[29] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for *AC*-like equational theories with homomorphisms. In *RTA'05*, pages 308–322, 2005.

[30] M. Latteux. Langages à un compteur. *J. Comput. Syst. Sci.*, 26(1):14–33, 1983.

[31] Z. Long, G. Calin, R. Majumdar, and R. Meyer. Language-theoretic abstraction refinement. In *FASE'12*, volume 7212 of *LNCS*, pages 362–376, 2012.

[32] R. Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 297(1-3):337–354, 2003.

[33] R. J. Parikh. On context-free languages. *J. ACM*, 13(4): 570–581, Oct. 1966.

[34] K. Sen and M. Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *CAV'14*, volume 4144 of *LNCS*, pages 300–314, 2006.

[35] L. G. Valiant and M. Paterson. Deterministic one-counter automata. *J. Comput. Syst. Sci.*, 10(3):340–350, 1975. .

[36] J. van Leeuwen. Effective constructions in well-partially-ordered free monoids. *Discrete Mathematics*, 21(3):237–252, 1978.

[37] K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational horn clauses. In *CADE-20*, volume 3632 of *LNCS*, pages 337–352, 2005.

[38] G. Zetzsche. An approach to computing downward closures. In *ICALP'15*, volume 9135 of *LNCS*, pages 440–451, 2015.

[39] G. Zetzsche. Computing downward closures for stacked counter automata. In *STACS'15*, volume 30 of *LIPIcs*, pages 743–756, 2015.

A. Equivalence between problems of characterizations of the three regular abstractions for OCA and simple OCA

Due to a following Lemma 30 without loosing of generality we can restrict our attention to considering problem of computing the regular abstractions of a subclass OCA, namely *simple OCA* defined analogously to OCA but different in the following aspects: (1) there are no zero tests, (2) there is a unique final state, $F = \{q_{\mathsf{final}}\}$, (3) only runs that start from the configuration $(q_{\mathsf{init}}, 0)$ and end at the configuration $(q_{\mathsf{final}}, 0)$ are considered *accepting*. The language of a simple OCA $\mathcal{A}$, also denoted $\mathcal{L}(\mathcal{A})$, is the set of words induced by accepting runs.

Definition 29. Let $\mathcal{A}$ be a simple OCA. We define a sequence of $\mathcal{L}_{(n)}(\mathcal{A})$ called approximants defined as language of words observed along runs of $\mathcal{A}$ from a configuration (q_0, n) to a configuration (q_{final}, n).

Let OCA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, for any $p, q \in Q$ we define $\mathcal{A}^{p,q} \stackrel{\text{def}}{=} (Q, \Sigma, p, \delta^+, \{q\})$ where δ^+ is the subset of all transitions in δ which are not test for zero transitions.

Lemma 30. *Let $K \in \mathbb{N}$ and $\diamondsuit \in \{\uparrow, \downarrow, \psi\}$ be an abstraction. Assume that there is a polynomial $g_\diamondsuit$ such that for any OCA $\mathcal{A}$ following holds: for every $\mathcal{A}^{p,q}$ there is an NFA $\mathcal{B}_{p,q,\diamondsuit}$ such that $\diamondsuit\mathcal{L}(\mathcal{A}^{p,q}) \subseteq \mathcal{L}(\mathcal{B}_{p,q,\diamondsuit}) \subseteq \diamondsuit(\mathcal{L}_{(K)}(\mathcal{A}^{p,q}))$ and $|\mathcal{B}_{p,q,\diamondsuit}| \le g_\diamondsuit(|\mathcal{A}|)$. Then there is a polynomial f such that for any $\mathcal{A}$ there is an NFA $\mathcal{B}^\diamondsuit$ of size at most $f(|\mathcal{A}|, K)$ with $\mathcal{L}(\mathcal{B}^\diamondsuit) = \diamondsuit(\mathcal{L}(\mathcal{A}))$.*

Remark 31. Restricting ourself to simple OCA, mentioned in the beginning of the section, is a consequence of Lemma 30 for $K = 0$. Indeed, if we can build polynomially bounded NFA for an abstraction of any given NFA, then we can do it for any simple OCA as well. On the other hand, if we can do it, for any simple OCA than due to Lemma 30 we can construct a polynomially bounded NFA that is abstraction of any given OCA.

In order to prove Lemma properly start from definitions.

Definition 32. Let $\mathcal{L}, \mathcal{L}' \subseteq \Sigma^*$. By a *concatenation of languages* denoted $\mathcal{L} \cdot \mathcal{L}'$ we mean a set of all words that can be obtained by concatenation of a word from $\mathcal{L}$ and a word $\mathcal{L}'$ i.e. $\{w \cdot w' | w \in \mathcal{L}, w' \in \mathcal{L}'\}$.

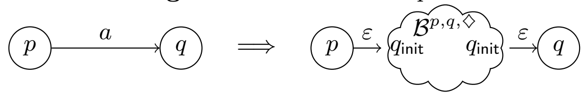
Let Π be a finite alphabet. A *substitution* is a function $\phi\colon \Pi \to 2^{\Sigma^*}$. Suppose $\mathcal{L} \subseteq \Pi^*$ is a language. For $w = w_1 \cdots w_n$ with $w_1, \ldots, w_n \in \Pi$, the set $\phi(w)$ contains all words $x_1 \cdots x_n$ where $x_i \in \phi(w_i)$. Lifting this notation to languages, we get $\phi(\mathcal{L}) = \bigcup_{w \in \mathcal{L}} \phi(w)$.

Lemma 33. *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be an OCA and suppose we are given, for each $\mathcal{A}^{p,q}$, an OCA $\mathcal{B}_{p,q}$ with $\mathcal{L}(\mathcal{A}^{p,q}) \subseteq \mathcal{L}(\mathcal{B}_{p,q}) \subseteq \mathcal{L}_{(n)}(\mathcal{A}^{p,q})$. Then there is an NFA $\mathcal{B}$ over an alphabet Π and substitution $\phi : \Pi \longrightarrow 2^{\Sigma^*}$ such that $\mathcal{L}(\mathcal{A}) = \phi(\mathcal{L}(\mathcal{B}))$ and that for every letter $a \in \Pi$ hold $\phi(a) \in \Sigma$ or $\phi(a)$ equals $\mathcal{L}(\mathcal{B}_{p,q})$ for some $p, q \in Q$.*

Proof. First, for any $n \in \mathbb{N}$ holds that, an accepting run π of $\mathcal{A}$ can be decomposed into $\pi = \pi_1\pi_1'\pi_2\pi_2' \ldots \pi_{n-1}'\pi_n$ where each π_i is a run in OCA such that counter values along π_i stay below n, and where each run π_i' induces a word from a language $\mathcal{L}_{(n)}(\mathcal{A}^{\mathsf{init.state}(\pi_i'),\mathsf{final.state}(\pi_i')})$.

Thus we define $\mathcal{B}$ as follows: Π is a Σ plus the set of triples (n, p, q) where $p, q \in Q$, the set of states is equal $Q \times \{0 \ldots n\}$, initial state is $(q_0, 0)$ and final states are

Figure 2. Substitution operation.



$F \times \{0\}$. The last thing is set of transitions; for $p, q \in Q$ it contains transition $((p, n), (n, p, q), 0, (q, n))$ and for any $i, j \in \{0 \ldots n\}$ the transition $((p, i)a, 0, (q, j))$ if and only if there is a move $(p, i) \xrightarrow{a} (q, j)$ in $\mathcal{A}$.

The ϕ is simply induced by its definition on elements of Π: for every $a \in \Pi$ we have that if $a \in \Sigma$ then $\phi(a) = \{a\}$ else if $a = (n, p, q)$ and $\phi(a) = \mathcal{L}_{(n)}(\mathcal{A}^{p,q})$.

It is easy check that $\mathcal{L}(\mathcal{A}) = \phi(\mathcal{L}(\mathcal{B}))$. $\square$

Next, we show how use $\mathcal{B}$ if we are interested in F-abstractions of $\mathcal{L}(\mathcal{A})$. We start from a following simple observations; for any language $\mathcal{L} = \mathcal{L}_1 \cdot \mathcal{L}_2$ we have that:

- $\mathcal{L}\downarrow = \mathcal{L}_1\downarrow \cdot \mathcal{L}_2\downarrow$
- $\mathcal{L}\uparrow = \mathcal{L}_1\uparrow \cdot \mathcal{L}_2\uparrow$
- $\psi(\mathcal{L}) = \psi(\mathcal{L}_1) \cdot \psi(\mathcal{L}_2)$

The natural consequences is a following lemma.

Lemma 34. *Let $\mathcal{B}$ be an automaton like one defined in Lemma 33 for a given OCA $\mathcal{A}$. Now let $\phi_\downarrow, \phi_\uparrow, \phi_\psi$ are defined as follows: If $a \in \Sigma$ then $\phi_\downarrow(a) = \{a\}\downarrow$, $\phi_\uparrow(a) = \{a\}\uparrow$, $\phi_\psi(a) = \{a\}$, on the other hand if a is of the form (n, p, q) i.e. $a \in \Pi \setminus \Sigma$ then $\phi_\downarrow((n, p, q)) = \mathcal{L}_{(n)}(\mathcal{A}^{p,q})\downarrow$, $\phi_\uparrow((n, p, q)) = \mathcal{L}_{(n)}(\mathcal{A}^{p,q})\uparrow$, $\phi_\psi((n, p, q)) = \psi(\mathcal{L}_{(n)}(\mathcal{A}^{p,q}))$.*

Then the following equalities hold

- $\mathcal{L}(\mathcal{A})\downarrow = \phi_\downarrow(\mathcal{L}(\mathcal{B}))$,
- $\mathcal{L}(\mathcal{A})\uparrow = \phi_\uparrow(\mathcal{L}(\mathcal{B}))$,
- $\psi(\mathcal{L}(\mathcal{A})) = \phi_\psi(\mathcal{L}(\mathcal{B}))$.

Proof. Lemma contains three claims which have very similar proofs, thus we present only one of them, for downward closure.

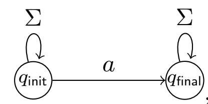
The first inclusion. Let $w' \in \mathcal{L}(\mathcal{A})\downarrow$ thus there is $w \in \mathcal{L}(\mathcal{A})$ such that $w' \preceq w$. Due to Lemma 33 we know that there is a word $u = u_1 \ldots u_k$, where $u_i \in \Pi$, such that $w \in \phi(u)$ (ϕ is defined in the proof of Lemma 33). We claim that $w' \in \phi_\downarrow(u)$. Indeed, observe that $\phi_\downarrow(u) = \phi_\downarrow(u_1) \cdots \phi_\downarrow(u_k) = \phi(u_1)\downarrow \cdots \phi(u_k)\downarrow$; further recall that if $\mathcal{L} = \mathcal{L}_1 \cdot \mathcal{L}_2$ implies $\mathcal{L}\downarrow = \mathcal{L}_1\downarrow \cdot \mathcal{L}_2\downarrow$ so we conclude that $\phi_\downarrow(u) = (\phi(u_1) \cdots \phi(u_k))\downarrow = \phi(u)\downarrow \supseteq w\downarrow$. As $w' \in w\downarrow$, this gives us what we need, so $w' \in \phi_\downarrow(u)$.

In the opposite direction. Let $w' \in \phi_\downarrow(\mathcal{L}(\mathcal{B}))$. Then $w' \in \phi_\downarrow(u)$ for some $u \in \mathcal{L}(\mathcal{B})$. Using similar calculation like previously we can show that $w' \in \phi(u)\downarrow$ but this mean that $w' \in w\downarrow$ for some $w \in \phi(u)$. Now as we know that $\phi(u) \in \mathcal{L}(\mathcal{A})$ we have that $w' \in \mathcal{L}(\mathcal{A})\downarrow$ which ends the proof. $\square$

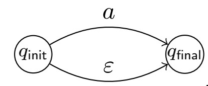
Finally we can prove the Lemma 30.

Proof. We start from Lemma 34. We take automaton B and choose suitable substitution $\phi_\diamondsuit$, where $\diamondsuit \in \{\uparrow, \downarrow, \psi\}$, depending on the abstraction that we are interested in. The idea is to substitute every transition in B, say $\tau = (p, a, 0, q)$, by a suitable designed automaton $\mathcal{B}^{p,q,\diamondsuit} = (Q_{p,q}, \Sigma, q_{\mathsf{init}}, \delta_{p,q}, \{q_{\mathsf{final}}\})$ that accepts the language $\phi_\diamondsuit(a)$.

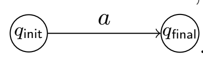
If $a \in \Pi \setminus \Sigma$ then automaton is $\mathcal{B}^{p,q}$, else if $a \in \Sigma$ then automaton that we glue is one of following:

for $\phi_{\uparrow}$ 

,

for $\phi_{\downarrow}$ 

,

for ϕ_{ψ} 

.

It is obvious that this construction provides a required NFA. The last question is about its size. According to Lemma 33 we have that $|B| = |\mathcal{A}| \cdot K$. So to estimate the size of the final construction we need to add at most $|\Sigma| \cdot 2|\mathcal{B}| + (|\mathcal{A}|)^2 \cdot g(|\mathcal{A}|)$ states, which proves the polynomial bound on the size of $\mathcal{B}^{\diamondsuit}$. $\square$

Remark 35. It is easy observation that above construction can be preformed in polynomial time in $K, |\mathcal{A}|$ and time needed to construct all automata $\mathcal{B}^{p,q,\diamondsuit}$ where $\diamondsuit \in \{\uparrow, \downarrow, \psi\}$.

B. Upward and Downward Closures

We begin by showing that for any OCA $\mathcal{A}$ we can effectively construct, in P, a NFA that accepts $\mathcal{L}(\mathcal{A})\uparrow$. This easy construction follows the argument traditionally used to bound the length of the shortest accepting run in a pushdown automaton. We give the details as the ideas used here recur elsewhere. We use the following notation in what follows: for a run ρ and an integer D we write $\rho[D]$ to refer to the quasi-run ρ' obtained from ρ by replacing the counter value v by $v + D$ in every configuration along the run.

Lemma 36. *Let $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ be a OCA and let w be a word accepted by $\mathcal{A}$. Then there is a word $y \preceq w$ in $\mathcal{L}(\mathcal{A})$ such that y is accepted by a run where the value of the counter never exceeds $|Q|^2 + 1$.*

Proof. We show that for any accepting run ρ reading a word w, there is an accepting run ρ', reading a word $y \preceq w$, in which the maximum value of the counter does not exceed $|Q|^2+1$. We prove this by double induction on the maximum value of the counter and the number of times this value is attained during the run ρ.

If the maximum value is below $|Q|^2 + 1$ there is nothing to prove. Otherwise let the maximum value $m > |Q|^2+1$ be attained c times along ρ. We break the run up as a concatenation of subruns $\rho = \rho_0\rho_1\rho_2\rho_3 \ldots \rho_m\rho'_{m-1} \ldots \rho'_2\rho'_1\rho'_0$ where

1. $\rho_0\rho_1\rho_2 \ldots \rho_m$ is the shortest prefix of ρ after which the counter reaches the value m.
2. $\rho_0\rho_1\rho_2 \ldots \rho_i$ is the longest prefix of $\rho_0\rho_1\rho_2 \ldots \rho_m$ after which the counter value is i, $1 \le i \le m-1$.
3. $\rho_0\rho_1\rho_2 \ldots \rho_m.\rho'_{m-1} \ldots \rho'_i$, is the shortest prefix of ρ with $\rho_0\rho_1\rho_2 \ldots \rho_m$ as a prefix and after which the counter value is i, $0 \le i \le m-1$.

Let the configuration reached after the prefix $\rho_0 \ldots \rho_i$ be (p_i, i), for $1 \le i \le m$. Similarly let the configuration reached after the prefix $\rho_0\rho_1\rho_2 \ldots \rho_m.\rho'_{m-1} \ldots \rho'_i$ be (q_i, i), for $0 \le i \le m-1$.

Now we make two observations: firstly, the value of the counter never falls below i during the segment of the run $\rho_{i+1} \ldots \rho'_i$ — this is by the definition of the ρ_is and ρ'_is. Secondly, there are $i < j$ such that $p_i = p_j$ and $q_i = q_j$ — this is because $m \ge |Q|^2 + 1$. Together this means that we may shorten the run by deleting the sequence of transitions corresponding to the segment $\rho_{i+1} \ldots \rho_j$ leading from (p_i, i) to (p_j, j) and the sequence corresponding to the segment $\rho'_{j-1} \ldots \rho'_i$ from (q_j, j) to (q_i, i) and still obtain a valid run of the system. That is, $\rho_0\rho_1 \ldots \rho_i\rho_{j+1}[-d]\rho_{j+2}[-d] \ldots \rho'_j[-d]\rho'_{i-1} \ldots \rho'_0$ is a valid run, where $d = j - i$. Clearly the word accepted by such a run is a subword of w, and further this run has at least one fewer occurrence of the maximal counter value m. The Lemma follows by an application of the induction hypothesis to this run and using the transitivity of the subword relation. $\square$

The set of words in $\mathcal{L}(\mathcal{A})$ accepted along runs where the value of the counter does not exceed $|Q|^2 + 1$ is accepted by an NFA with $|Q|.(|Q|^2+1)$ states (it keeps the counter values as part of the state). Combining this with the standard construction for upward closure for NFAs we get

Theorem 2. *There is a polynomial-time algorithm that takes as input an OCA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ and computes an NFA with $O(|\mathcal{A}|^3)$ states accepting $\mathcal{L}(\mathcal{A})\uparrow$.*

The construction can be extended to general OCA without any change in the complexity

Next we show a polynomial time procedure that constructs an NFA accepting the downward closure of the language of any simple OCA. For pushdown automata the construction involves a necessary exponential blow-up. We sketch some observations that lead to our polynomial time construction.

Let $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ be a simple OCA and let $K = |Q|$. Consider any run ρ of $\mathcal{A}$ from a configuration (p, i) to a configuration (q, j). If the value of the counter increases (resp. decreases) by at least K in ρ then, it contains a segment that can be *pumped* (or iterated) to increase (resp. decrease) the value of the counter. If the increase in the value of the counter in this iterable segment is k then by choosing an appropriate number of iterations we may increase the value of the counter at the end of the run by any multiple of this k. Quite clearly, the word read along this iterated run will be a superword of word read along ρ. The following lemmas,whose proof is a simplified version of that of Lemma 36, formalize this.

Lemma 37. *Let $(p, i) \xrightarrow{x} (q, j)$ with $j - i > K$. Then, there is an integer $k > 0$ such that for each $N \ge 0$ there is a run $(p, i) \xrightarrow{w = y_1.(y_2)^{N+1}.y_3} (p', j + N.k)$ with $x = y_1y_2y_3$.*

Proof. Consider the run $(p, i) \xrightarrow{x} (q, j)$ and break it up as

$$(p, i) = (p_i, i) \xrightarrow{x_1} (p_{i+1}, i+1) \xrightarrow{x_2} (p_{i+2}, i+2) \ldots$$
$$\ldots \xrightarrow{x_j} (p_j, j) \xrightarrow{x'} (q, j)$$

where the run $(p_i, i) \xrightarrow{x_1} \ldots \xrightarrow{x_r} (p_r, r)$ is the shortest prefix after which the value of the counter attains the value r. Since $j - i > K$ it follows that there are r, r' with $i \le r < r' \le j$ such that $p_r = p_{r'}$. Clearly one may iterate the segment of the run from (p_r, r) to (p_r, r') any number of times, say $N \ge 0$, to get a run $(p, i) \xrightarrow{w} (q, j + (r' - r)N)$. where $w = x_1 \ldots x_r(x_{r+1} \ldots x_{r'})^{N+1}x_{r+1} \ldots x_k$. Setting $k = r' - r$ yields the lemma. $\square$

An analogous argument shows that if the value of the counter decreases by at least K in ρ then we may iterate a suitable segment to reduce the value of the counter by any multiple of k' (where the k' is the net decrease in the value of the counter along this segment) while reading a superword. This is formalized as

Lemma 38. *Let $(q', j') \xrightarrow{z} (p', i')$ with $j' - i' > K$. Then, there is an integer $k' > 0$ such that for every $N \geq 0$ there is a run $(q', j' + N.k') \xrightarrow{w = y_1 (y_2)^{N+1} y_3} (p', i')$ with $z = y_1 y_2 y_3$.*

Proof. We break the run into segments as:

$$(q', j') = (q_{j'}, j') \xrightarrow{z_{j'-1}} (q_{j'-1}, j'-1) \xrightarrow{z_{j'-2}} (q_{j'-2}, j'-2) \ldots$$
$$\ldots \xrightarrow{z_{i'}} (q_{i'}, i') \xrightarrow{z'} (p', i')$$

where $(q_{j'}, j') \xrightarrow{z_{j'-1}} (q_{j'-1}, j'-1) \xrightarrow{z_{j'-2}} (q_{j'-2}, j'-2) \ldots \xrightarrow{z_t} (q_t, t)$ is the shortest prefix after which the value of counter is t. Since $j' - i' > K$ it follows that there are $t, t', j' \geq t > t' \geq i'$ such that $q_t = q_{t'}$. Then, starting at any configuration $(q_t, t + (t - t')N)$, $N \in \mathbb{N}$ we may iterate the transitions in the run $(q_t, t) \rightarrow (q_{t'}, t')$ an additional N times. This yields a run $(q_t, t + (t-t')N) \xrightarrow{z''} (q_{t'}, t')$ where $z'' = (z_{t-1} \ldots z_{t'})^{N+1}$. Observe that $z_{t-1} \ldots z'_t$ is a subword of z''. Finally, notice that this also means that $(q', j' + N.(t - t')) \xrightarrow{z_1 \ldots z_t} (q_t, t + N.(t - t')) \xrightarrow{z''} (q_{t'}, t') \xrightarrow{z_{t'+1} \cdots z_i} (p', i')$. Taking $k' = (t - t')$ completes the proof. ☐

A consequence of these somewhat innocous lemmas is the following interesting fact: we can turn a triple consisting of two runs, where the first one increases the counter by at least K and the second one decreases the counter by at least K, and a quasi-run that connects them, into a real run provided we are content to read a superword along the way.

Lemma 39. *Let $(p, i) \xrightarrow{x} (q, j) \stackrel{y}{\Rightarrow} (q', j') \xrightarrow{z} (p', i')$, with $j - i > K$ and $j' - i' > K$. Then, there is a run $(p, i) \xrightarrow{w} (p', i')$ such that $xyz \preceq w$.*

Proof. Let the lowest value of counter in the entire run be m. If $m \geq 0$ then the given quasi-run is by itself a run and hence there is is nothing to prove. Let us assume that m is negative.

First we use Lemma 3, to get a k and an x' for any $N > 1$ and a run $(p, i) \xrightarrow{x'} (q, j + k.N)$ with $x \preceq x'$. We can then extend this to a run $(p, i) \xrightarrow{x'} (q, j + k.N) \stackrel{y}{\Rightarrow} (q', j' + k.N)$, by simply choosing N so that $k.N > m$. Then, we have that the value of the counter is ≥ 0 in every configuration of this quasi-run. Thus $(p, i) \xrightarrow{x'} (q, j + k.N) \xrightarrow{y} (q', j' + k.N)$ for any such N. Now, we apply Lemma 4 to the run $(q', j') \xrightarrow{z} (p', i')$ to obtain the k'. We now set our N to be a value divisible by k', say $k'.I$. Thus, $(p, i) \xrightarrow{x'} (q, j + k.k'.I) \xrightarrow{y} (q', j' + k.k'.I)$ and now we may again use Lemma 4 to conclude that $(q', j' + k.k'.I) \xrightarrow{z''} (p', i')$ with $x \preceq x'$ and $z \preceq z''$. This completes the proof. ☐

Interesting as this may be, this lemma still relies on the counter value being recorded exactly in all the three segments in its antecedent and this is not sufficient. In the next step, we weaken this requirement (while imposing the condition that $q = q'$ and $j = j'$) by releasing the (quasi) middle segment from this obligation.

Lemma 40. *Let $(p, i) \xrightarrow{x} (q, j)$, $(q, j) \xrightarrow{z} (p', i')$, with $j - i > K$ and $j' - i' > K$. Let there be a walk from q to q that reads y. Then, there is a run $(p, i) \xrightarrow{w} (p', i')$ such that $xyz \preceq w$.*

Proof. Let the given walk result in the quasi-run $(q, j) \stackrel{y}{\Rightarrow} (q, j + d)$ (where d is the net effect of the walk on the counter, which may be positive or negative). Iterating this quasi-run m times yields a quasi-run $(q, j) \stackrel{y^m}{\Longrightarrow} (q, j + m.d)$, for any $m \geq 0$. Next, we use Lemma 3 to find a $k > 0$ such that for each $N > 0$ we have a run $(p, i) \xrightarrow{x_N} (q, j + N.k)$ with $x \preceq x_N$. Similarly, we use Lemma 4 to find a $k' > 0$ such that for each $N' > 0$ we have a run $(q, j + N'.k') \xrightarrow{y_{N'}} (p', i')$ with $y \preceq y_{N'}$.

Now, we pick m and N to be multiples of k' in such a way that $N.k + m.d > 0$. This can always be done since k is positive. Thus, $N.k + m.d = N'.k'$ with $N' > 0$. Now we try and combine the (quasi) runs $(p, i) \xrightarrow{x_N} (q, j + N.k)$, $(q, j + N.k) \stackrel{y^m}{\Longrightarrow} (q, j + N.k + m.d)$ and $(q, j + N'.k') \xrightarrow{y_{N'}} (p', i')$ to form a run. We are almost there, as $j + N.k + m.d = j + N'.k'$. However, it is not guaranteed that this combined quasi-run is actually a run as the value of the counter may turn negative in the segment $(q, j + N.k) \stackrel{y^m}{\Longrightarrow} (q, j + N.k + m.d)$. Let $-N''$ be the smallest value attained by the counter in this segment. Then by replacing N by $N + N''.k'$ and N' by $N' + N''.k$ we can manufacture a triple which actually yields a run (since the counter values are ≥ 0), completing the proof. ☐

With this lemma in place we can now explain how to relax the usage of counters. Let us focus on runs that are interesting, that is, those in which the counter value exceeds K at some point. Any such run may be broken into 3 stages: the first stage where counter value starts at 0 and remains strictly below $K + 1$, a second stage where it starts and ends at $K + 1$ and a last stage where the value begins at K and remains below K and ends at 0 (the 3 stages are connected by two transitions, an increment and a decrement). Suppose, we write the given accepting run as $(p, 0) \xrightarrow{w_1} (q, c) \xrightarrow{w_2} (r, 0)$ where (q, c) is a configuration in the second stage. If $a \in \Sigma$ is a letter that may be read in some transition on some walk from q to q. Then, $w_1 a w_2$ is in $\mathcal{L}(\mathcal{A}) \downarrow$. This is a direct consequence of the above lemma. It means that in the configurations in the middle stage we may freely read certain letters without bothering to update the counters. This turns out to be a crucial step in our construction. To turn this relaxation idea into a construction, the following seems a natural.

We make an equivalent, but expanded version of $\mathcal{A}$. This version has 3 copies of the state space: The first copy is used as long as the value of the counter stays below $K + 1$ and on attaining this value the second copy is entered. The second copy simulates $\mathcal{A}$ exactly but nondeterministically chooses to enter third copy whenever the counter value is moves from $K + 1$ to K. The third copy simulates $\mathcal{A}$ but does not permit the counter value to exceed K. For every letter a and state q with a walk from q to q along which a is read on some transition, we add a self-loop transition to the state corresponding to q in the second copy that does not affect the counter and reads the letter a. This idea has two deficiencies: first, it is not clear how to define the transition from the second copy to the third copy, as that requires knowing that value of the counter is $K + 1$, and second, this is still an OCA (since the second copy simply faithfully simulates $\mathcal{A}$) and not an NFA.

Suppose we bound the value of the counter by some value U in the second stage. Then we can overcome both of these defects and construct a finite automaton as follows: The state space of the resulting NFA has stages of the form (q, i, j) where $j \in \{1, 2, 3\}$ denotes the stage to which this copy of q belongs. The value i is the value of the counter as maintained within the state of the NFA. The transitions interconnecting the stages go from a state of the form $(q, K, 1)$ to one of the form $(q', K+1, 2)$ (while simulating a transition involving an increment) and from a stage of the form $(q, K+1, 2)$ to one of the form $(q', K, 3)$ (while simulating a decrement). The value of i is bounded by K if $j \in \{1, 3\}$ while it is bounded by U if $j = 2$. (States of the form $(q, i, 2)$ also have self-loop transitions described above.) By using a slight generalization of Lemma 6, which allows for the simultaneous insertion of a number of walks (or by applying the Lemma iteratively), we can show that any word accepted by such a finite automaton lies $\mathcal{L}(\mathcal{A})\!\downarrow$. However, there is no guarantee that such an automaton will accept every word in $\mathcal{L}(\mathcal{A})\!\downarrow$. The second crucial point is that we are able to show that if $U \geq K^2+K+1$ then every word in $\mathcal{L}(\mathcal{A})$ is accepted by this 3 stage NFA. We show that for each accepting run ρ in $\mathcal{A}$ there is an accepting run in the NFA reading the same word. The proof is by a double induction, first on the maximum value attained by the counter and then on the number of times this value is attained along the run. Clearly, segments of the run where the value of the counter does not exceed $K^2 + K + 1$ can be simulated as is. We then show that whenever the counter value exceeds this number, we can find suitable segments whose net effect on the counter is 0 and which can be simulated using the self-loop transitions added to stage 2 (which do not modify the counters), reducing the maximum value of the counter along the run.

We now present the formal details. We begin by describing the NFA $\mathcal{A}_U$ where $U \geq K + 1$.

$$\mathcal{A}_U = (Q_1 \cup Q_2 \cup Q_3, \Sigma, \Delta, i_U, F_U\})$$

where $Q_1 = Q \times \{0 \dots K\} \times \{1\}$, $Q_2 = Q \times \{0 \dots U\} \times \{2\}$ and $Q_3 = Q \times \{0 \dots K\} \times \{3\}$. We let $i_U = (s, 0, 1)$ and $F_U = \{(f, 0, 1), (f, 0, 3) \mid f \in F\}$. The transition relation is the union of the relations Δ_1, Δ_2 and Δ_3 defined as follows:

Transitions in Δ_1:

1. $(q, n, 1) \xrightarrow{a} (q', n, 1)$ for all $n \in \{0 \dots K\}$ whenever $(q, a, i, q') \in \delta$. Simulate an internal move.
2. $(q, n, 1) \xrightarrow{a} (q', n-1, 1)$ for all $n \in \{1 \dots K\}$ whenever $(q, a, -1, q') \in \delta$. Simulate a decrement.
3. $(q, n, 1) \xrightarrow{a} (q', n+1, 1)$ for all $n \in \{0 \dots K-1\}$ whenever $(q, a, +1, q') \in \delta$. Simulate an increment.
4. $(q, K, 1) \xrightarrow{a} (q', K+1, 2)$ whenever $(q, a, +1, q') \in \delta$. Simulate an increment and shift to second phase.

Transitions in Δ_2:

1. $(q, n, 2) \xrightarrow{a} (q', n, 2)$ for all $n \in \{0 \dots U\}$ when $(q, a, i, q') \in \delta$. Simulate an internal move.
2. $(q, n, 2) \xrightarrow{a} (q', n-1, 2)$ for all $n \in \{1 \dots U\}$ whenever $(q, a, -1, q') \in \delta$. Simulate a decrement.
3. $(q, K+1, 2) \xrightarrow{a} (q', K, 3)$ whenever $(q, a, -1, q') \in \delta$. Simulate a decrement and shift to third phase.
4. $(q, n, 2) \xrightarrow{a} (q', n+1, 2)$ for all $n \in \{0 \dots U-1\}$ whenever $(q, a, +1, q') \in \delta$. Simulate an increment move.
5. $(q, n, 2) \xrightarrow{a} (q, n, 2)$ whenever there is a walk from q to q on some word w and, $a \preceq w$. Freely simulate loops.

Transitions in Δ_3:

1. $(q, n, 3) \xrightarrow{a} (q', n, 3)$ for all $n \in \{0 \dots K\}$ whenever $(q, a, i, q') \in \delta$. Simulate an internal move.
2. $(q, n, 3) \xrightarrow{a} (q', n-1, 3)$ for all $n \in \{1 \dots K\}$ whenever $(q, a, -1, q') \in \delta$. Simulate a decrement.
3. $(q, n, 3) \xrightarrow{a} (q', n+1, 3)$ for all $n \in \{0 \dots K-1\}$ whenever $(q, a, +1, q') \in \delta$. Simulate an increment move.

The following Lemma, which is easy to prove, states that the first and third phases simulate faithfully any run where the value of the counter is bounded by K.

Lemma 41. *1. If $(q, i, l) \xrightarrow{w} (q', j, l)$ in $\mathcal{A}_U$ then $(q, i) \xrightarrow{w} (q', j)$ in $\mathcal{A}$, for $l \in \{1, 3\}$.*
2. If $(q, i) \xrightarrow{w} (q', j)$ in $\mathcal{A}$ through a run where the value of the counter is $\leq K$ in all the configurations along the run then $(q, i, l) \xrightarrow{w} (q', i, l)$ for $l \in \{1, 3\}$.

The next Lemma extends this to runs involving the second phase as well. All moves other than those simulating unconstrained walks can be simulated by $\mathcal{A}$. The second phase of $\mathcal{A}_U$ can also simulate any run where the counter is bounded by U. Again the easy proof is omitted.

Lemma 42. *1. If $(q, i, l) \xrightarrow{w} (q', j, l')$ is a run of $\mathcal{A}_U$ in which no transition from Δ_2 of type 5 is used then $(q, i) \xrightarrow{w} (q', j)$ is a run of $\mathcal{A}$.*
2. If $\rho = (q_0, i_0) \xrightarrow{a_1} (q_1, i_1) \xrightarrow{a_2} \dots \xrightarrow{a_m} (q_m, i_m)$ is a run in $\mathcal{A}$ in which the value of the counter never exceeds U then $\rho' = (q_0, i_0, 2) \xrightarrow{a_1} (q_1, i_1, 2) \xrightarrow{a_2} \dots \xrightarrow{a_m} (q_m, i_m, 2)$ is a run in $\mathcal{A}_U$.

Now, we are in a take the first step towards generalizing Lemma 6 to prove that $\mathcal{L}(\mathcal{A}_U) \subseteq \mathcal{L}(\mathcal{A})\!\downarrow$.

Lemma 43. *Let $(q, i, 2) \xrightarrow{w} (q', j, 2)$ be a run in $\mathcal{A}_U$. Then, there is an $N \in \mathbb{N}$, words $x_0, y_0, x_1, y_1, \dots, x_N$, and integers $n_0, n_1, \dots, n_{N-1}$ such that, in $\mathcal{A}$ we have:*

1. $w \preceq x_0 y_0 x_1 y_1 \dots x_N$.
2. $(q, i) \xrightarrow{x_0 y_0 \dots x_N} (q', j')$ where $j' = j + n_0 + n_1 \dots + n_{N-1}$.
3. $(q, i) \xrightarrow{x_0 y_0^{m_0} x_1 y_1^{m_1} \dots x_N} (q', j'')$ where $j'' = j + m_0.n_0 + m_1.n_1 \dots + m_{N-1}.n_{N-1}$, for any $1 \leq m_0, m_1, \dots m_{N-1}$.

Note that 2 is just a special case of 3 when $m_r = 1$ for all r.

Proof. The run $(q, i, 2) \xrightarrow{w} (q', j, 2)$ in $\mathcal{A}_U$ uses only transitions of the types $1, 2, 4$ and 5 in Δ_2. Let N be the number of transitions of type 5 used in the run. We then break up the run as follows:

$$(q, i, 2) \xrightarrow{x_0} (p_0, i_0, 2) \xrightarrow{a_0} (p_0, i_0, 2) \xrightarrow{x_1} (p_1, i_1, 2) \dots$$
$$\dots (p_{N-1}, i_{N-1}, 2) \xrightarrow{a_{N-1}} (p_{N-1}, i_{N-1}, 2) \xrightarrow{x_N} (q', j, 2)$$

where the transitions, indicated above, on a_i's are the N moves using transitions of type 5 in the run. Let $(p_r, i_r) \xRightarrow{y_r} (p_r, i'_r)$ be a quasi-run with $a_r \preceq y_r$ and let $n_r = i'_r - i_r$. Clearly $w \preceq x_0 y_0 x_1 y_1 \dots x_N$.

It is quite easy to show by induction on r, $0 \leq r < N$, by replacing moves of types $1, 2$ and 4 by the corresponding moves in $\mathcal{A}$ and moves of type 5 by the iterations of the quasi-runs identified above that:

$$
\begin{aligned}
(q,i) &\xrightarrow{x_0}(p_0,i_0) \xRightarrow{y_0^{m_0}} (p_0, i_0+m_0.n_0) \xRightarrow{x_1} (p_1, i_1+m_0.n_0)\\
&\xRightarrow{y_1^{m_1}} (p_1, i_1+m_0.n_0+m_1.n_1)\\
&\dots\\
&\xrightarrow{x_r}(p_r, i_r+m_0.n_0\dots+m_{r-1}.n_{r-1})\\
&\xRightarrow{y_r^{m_r}} (p_r, i_r+m_0.n_0\dots m_r.n_r)\\
&\xrightarrow{x_{r+1}}(p_{r+1}, i_{r+1}+m_0n_0\dots m_r.n_r)
\end{aligned}
$$

and with $r = N-1$ we have the desired result. □

Now, we use an argument that generalizes Lemma 6 in order to show that:

Lemma 44. *Let w be any word accepted by the automaton $\mathcal{A}_U$. Then, there is a word $w' \in \mathcal{A}$ such that $w \preceq w'$.*

Proof. If states in Q_2 are not visited in the accepting run of $\mathcal{A}_U$ on w then we can use Lemma 41 to conclude that $w \in \mathcal{A}$. Otherwise, we break up the run of $\mathcal{A}_U$ on w into three parts as follows:

$$
\begin{aligned}
(s,0,1) &\xrightarrow{w_1}(p,K,1) \xrightarrow{a_1}(q,K+1,2) \xrightarrow{w_2}(r,K+1,2)\\
&\xrightarrow{a_2}(t,K,3) \xrightarrow{w_3}(f,0,3)
\end{aligned}
$$

Using Lemma 41 we have $(s,0) \xrightarrow{w_1}(p,K)$ and $(t,K) \xrightarrow{w_3}(f,0)$. We then apply Lemmas 3 and 4 to these two segments respectively to identify k and k'. Next we use Lemma 43 to identify the positive integer N, integers $n_0, n_1, \dots n_{N-1}$ and the quasi-run

$$(q, K+1) \xRightarrow{x_0y_0\dots x_N} (r, K+1+n_0+n_1\dots+n_{N-1})$$

with $w_2 \preceq x_0y_0x_1y_1\dots x_{N-1}y_{N-1}x_N$. We identify numbers $m, m_0, m_1, \dots, m_{N-1}$, all ≥ 1, such that $(m-1).k+m_0.n_0+\dots m_{N-1}.n_{N-1} = k'.m'$ for some $m' \geq 0$. By taking $m-1$ and each m_i to be some multiple of k' we get the sum $(m-1).k+m_0.n_0+\dots m_{N-1}n_{N-1}$ to be a multiple of k', however this multiple may not be positive. Since $k > 0$, by choosing $m-1$ to be a sufficiently large multiple of k' we can ensure that $m' \geq 0$. Using these numbers we construct the quasi-run

$$
\begin{aligned}
&(q,K+1+(m-1).k) \xRightarrow{x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N}\\
&\quad (r, K+1+(m-1).k+m_0n_0+\dots m_{N-1}n_N)
\end{aligned}
$$

with $K+1+(m-1).k+m_0n_0+\dots m_{N-1}n_N = K+1+k'.m'$. Let l be the lowest value attained in this quasi-run. If $l \geq 0$ then

$$(q, K+1+(m-1).k) \xrightarrow{x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N}(r, K+1+k'.m')$$

and using Lemma 3 and 4 we get

$$
\begin{aligned}
(s,0) &\xrightarrow{w}(p, K+(m-1).k)\\
&\xrightarrow{a_1}(q, K+1+(m-1).k)\\
&\xrightarrow{x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N}(r, K+1+k'.m')\\
&\xrightarrow{a_2}(t, K+k'.m')\\
&\xrightarrow{z}(f,0,3)
\end{aligned}
$$

with $w_1 \preceq w$, $w_2 \preceq x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N$ and $w_3 \preceq z$ as required.

Suppose $l < 0$. Then, we let I be a positive integer such that $I.k+l > 0$ and $I = k'.m''$ (i.e. I is divisible by k') which must exist since $k > 0$. Then

$$(q, K{+}1{+}(m{-}1).k{+}I.k) \xRightarrow{x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N} (r, K{+}1{+}I.k{+}k'.m')$$

is a quasi-run in which the counter values are always ≥ 0 and is thus a run. Once again, we may use Lemmas 3 and 4 (since $I.k$ is a multiple of k') to get

$$
\begin{aligned}
(s,0) &\xrightarrow{w}(p, K+(m-1).k+I.k)\\
&\xrightarrow{a_1}(q, K+1+(m-1).k+I.k)\\
&\xrightarrow{x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N}(r, K+1+k'.m')\\
&\xrightarrow{a_2}(t, K+k'.m'+I.k)\\
&\xrightarrow{z}(f,0,3)
\end{aligned}
$$

with $w_1 \preceq w$, $w_2 \preceq x_0y_0^{m_0}x_1y_1^{m_1}\dots x_N$ and $w_3 \preceq z$. This completes the proof of the Lemma.

□

Next, we show that if $U \geq K^2+K+1$ then $\mathcal{L}(\mathcal{A}_) \subseteq \mathcal{L}(\mathcal{A}_U)$.

Lemma 45. *Let $U \geq K^2+K+1$. Let w be any word in $\mathcal{L}(\mathcal{A})$. Then, w is also accepted by $\mathcal{A}_U$.*

Proof. The proof is accomplished by examining runs of the from $(s,0) \xrightarrow{w}(f,0)$ and showing that such a run may be *simulated* by $\mathcal{A}_U$ transition by transition in a manner to be described below. Any run $\rho = (s,0) \xrightarrow{w}_S(f,0)$ can be broken up into parts as follow:

$$(s,0) \xrightarrow{x}(h,j) \xrightarrow{y}(h',j') \xrightarrow{z}(f,0)$$

where, $\rho_1 = (s,0) \xrightarrow{x}(h,j)$ is the longest prefix where the counter value does not exceed K, $\rho_3 = (h',j') \xrightarrow{z}(f,0)$, is the longest suffix, of what is left after removing ρ_1, in which the value of the counter does not exceed K, and $\rho_2 = (h,j) \xrightarrow{y}(h',j')$ is what lies in between. We note that using Lemma 41 we can conclude that there are runs $(s,0,1) \xrightarrow{x}(h,j,1)$ and $(h',j',3) \xrightarrow{z}(f,0,3)$. Further, observe that if value of the counter never exceeds K then ρ_2 and ρ_3 are empty, $x = w$, $h = f$ and $j = 0$. In this case, using Lemma 41, there is a (accepting) run $(s,0,1) \xrightarrow{w}(f,0,1)$.

If the value of the counter exceeds K then $j = j' = K$ and by Lemma 41, $(s,0,1) \xrightarrow{x}(h,K,1)$, $(h',K,3) \xrightarrow{z}(f,0,3)$ and ρ_2 is non-empty. Further suppose that, ρ_2, when written out as a sequence of transitions is of the form

$$
\begin{aligned}
\rho_2 = &(h,K) \xrightarrow{a}(p,K+1) = (p_0,i_0) \xrightarrow{a_1}(p_1,i_1)\\
&\xrightarrow{a_2}(p_2,i_2)\dots \xrightarrow{a_n}(p_n,i_n) = (q,K+1) \xrightarrow{b}(h',K)
\end{aligned}
$$

We will show by double induction on the maximum value of the counter value attained in the run ρ_2 and the number of times the maximum is attained that there is a run

$$
\begin{aligned}
\rho_2' = &(h,K,1) \xrightarrow{a}(p,K+1,2) = (p_0',i_0',2) \xrightarrow{a_1}(p_1',i_1',2)\\
&\xrightarrow{a_2}(p_2',i_2',2)\dots \xrightarrow{a_n}(p_n',i_n',2) = (q,K+1,2) \xrightarrow{b}(h',K,3)
\end{aligned}
$$

such that for all i, $0 \leq i < n$,

1. either $p_i = p_i'$ and $p_{i+1} = p_{i+1}'$ and the ith transition (on a_{i+1}) is of type $1, 2$ or 4,
2. or $p_i' = p_{i+1}'$, $i_i' = i_{i+1}'$, $p_i' \Rightarrow p_i$ and $p_{i+1} \Rightarrow p_i'$ so that the ith transition (on a_{i+1}) is a transition of type 5.

For the basis, notice that if the maximum value attained is $\leq K^2 + K + 1$ then, by Lemma 42, there is a run of $\mathcal{A}_U$ that simulates ρ_2 such that item 1 above is satisfied for all i.

Now, suppose the maximum value attained along the run is $m > K^2 + K + 1$. We proceed along the lines of the proof of Lemma 36. We first break up the run ρ_2 as

$$
\begin{aligned}
(h, K) &\xrightarrow{a} (p_0, K+1) = (q_{K+1}, K+1) \xrightarrow{y_{K+2}} (q_{K+2}, K+2) \\
&\xrightarrow{y_{K+3}} (q_{K+3}, K+3) \ldots \xrightarrow{y_m} (q_m, m) \\
&\xrightarrow{y'_{m-1}} (q'_{m-1}, m-1) \ldots \xrightarrow{y'_{K+1}} (q'_{K+1}, K+1) \\
&\xrightarrow{z} (q, K+1) \xrightarrow{b} (h', K)
\end{aligned}
$$

where

- The prefix upto (q_m, m), henceforth referred to as σ_m, is the shortest prefix after which the counter value is m.
- The prefix upto (q_i, i), $K + 1 \leq i < m$ is the longest prefix of σ_m after which the value of the counter is i.
- The prefix upto (q'_i, i), $K + 1 \leq i < m$ is the shortest prefix of ρ_2 with σ_m as a prefix after which the counter value is i.

By construction, the value of the counter in the segment of the run from $(q_i, i) \rightarrow \ldots \rightarrow (q'_i, i)$ never falls below i. Further, by simple counting, there are i, j with $K + 1 \leq i < j \leq m$ such that $q_i = q_j$ and $q'_i = q'_j$. Thus, by deleting the segment of the runs from (q_i, i) to (q_i, j) and (q'_j, j) to (q'_j, i) we get a shorter run ρ_d which looks like

$$
\begin{aligned}
(h, K) &\xrightarrow{a} (p_0, K+1) = (q_{K+1}, K+1) \ldots \\
&\xrightarrow{y_i} (q_i, i) \xrightarrow{y_{j+1}} (q_{j+1}, i+1) \ldots \\
&\xrightarrow{y_m} (q_m, m-j+i) \xrightarrow{y'_{m-1}} \ldots \\
&\xrightarrow{y'_j} (q'_j, i) \xrightarrow{y'_{i-1}} (q'_{i-1}, i-1) \ldots (q'_{K+1}, K+1) \\
&\xrightarrow{z} (q, K+1) \xrightarrow{b} (h', K)
\end{aligned}
$$

This run reaches the value m at least one time fewer than ρ_2 and thus we may apply the induction hypothesis to conclude the existence of a run ρ'_d of $\mathcal{A}_U$ that simulates this run move for move satisfying the properties indicated in the induction hypothesis. Let this run be:

$$
\begin{aligned}
(h, K, 1) &\xrightarrow{a} (r_0, K+1, 2) \ldots \\
&\xrightarrow{y_i} (r_i, c_i, 2)) \xrightarrow{y_{j+1}} (r_{j+1}, c_{j+1}, 2) \ldots \\
&\xrightarrow{y_m} (r_m, c_m, 2) \xrightarrow{y'_{m-1}} \ldots \\
&\xrightarrow{y'_j} (r'_j, c'_j, 2) \xrightarrow{y'_{i-1}} (r'_{i-1}, c'_{i-1}, 2) \ldots (r'_{K+1}, c'_{K+1}, 2) \\
&\xrightarrow{z} (r', K+1, 2) \xrightarrow{b} (h', K)
\end{aligned}
$$

Now, if $(p_l, i_l) \xrightarrow{a_{l+1}} (p_{l+1}, i_{l+1})$ was a transition in ρ_2 in the part of the run from (q_i, i) to (q_i, j) then, $q_i \Rightarrow p_l$, $p_l \xRightarrow{a_{l+1}} q_i$ and $p_{l+1} \Rightarrow q_i$. Now, either $r_i = q_i$ or $r_i \Rightarrow q_i$, and $q_{j+1} \Rightarrow r_i$ and $(q_i, a_{j+1}, op, q_{j+1})$ is a transition for some op. In the both cases clearly $r_i \xRightarrow{a_{l+1}} r_i$. Thus every such deleted transition can be simulated by a transition of the form $(r_i, c_i, 2) \xrightarrow{a_{l+1}} (r_i, c_i, 2)$.

A similar argument shows that every transition of the form $(p_l, i_l) \xrightarrow{a_{l+1}} (p_{l+1}, i_{l+1})$ deleted in the segment (q'_j, j) to (q'_j, i) can be simulated by $(r'_j, c'_j, 2) \xrightarrow{a_{l+1}} (r'_j, c'_j, 2)$. Thus we can extend the run ρ'_d to a run ρ'_2 that simulates ρ_2 fulfilling the requirements of the induction hypothesis. This completes the proof of this lemma. □

Notice that the size of the state space of $\mathcal{A}_U$ is $K.(K^2 + K + 1)$ when $U = K^2 + K + 1$. Since downward closures of NFAs can be constructed by just adding additional (ε) transitions, Lemmas 44 and 45 imply that:

Theorem 7. *There is a polynomial-time algorithm that takes as input a simple OCA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ and computes an NFA with $O(|\mathcal{A}|^3)$ states accepting $\mathcal{L}(\mathcal{A})\downarrow$.*

A closer look reveals that the complexity remains the same even for general OCA. We only need one copy of $\mathcal{A}_U$.

C. Bounded alphabet

The result of this section is the following theorem.

Theorem 46. *For any fixed alphabet Σ there is a polynomial-time algorithm that, given as input a one-counter automaton over Σ with n states, computes a Parikh-equivalent NFA.*

Note that in Theorem 46 the size of the alphabet Σ is fixed. The theorem implies, in particular, that any one-counter automaton over Σ with n states has a Parikh-equivalent NFA of size $\mathrm{poly}_\Sigma(n)$, where poly_Σ is a polynomial of degree bounded by $f(|\Sigma|)$ for some computable function f.

The numbering sequence from the conference version is mapped as follows:

- Theorem 8 is Theorem 46,
- Definition 9 is Definition 50,
- Definition 10 is Definition 51,
- Definition 11 is Definition 52,
- Lemma 12 is Lemma 54,
- Definition 13 is Definition 55,
- Lemma 14 is Lemma 56,
- Lemma 15 is Lemma 57,
- Lemma 16 is Lemma 64,
- Lemma 17 is Lemma 60.

Remark 47. We start with yet another simplifying assumption (in addition to that of subsection 2.3). In one part of our proof (subsection C.2 below) we will need to rely on the fact that short input words can only be observed along short runs; this is, of course, always true if all OCA in question have to ε-transitions. We note here that, for the purpose of computing the Parikh image, we can indeed assume without loss of generality that this is the case. To see this, replace all ε on transitions of an OCA $\mathcal{A}$ with a fresh letter $e \notin \Sigma$; i.e., increase the cardinality of Σ by 1. Now construct an NFA Parikh-equivalent to the new OCA over the extended alphabet; it is easy to see that replacing all occurrences of e in the NFA by ε will give us an appropriate NFA. In this NFA ε-transitions can be eliminated in a standard way.

C.1 Basic definitions

We start from a sequence of definitions which are necessary to describe pumping schemas for one-counter automata, that are crucial to capture the linear structure of the Parikh image of the language accepted by a given one-counter automaton.

Definition 48 (attributes of runs and walks)**.** For a run $\pi = (p_0, c_0), t_1, (p_1, c_1), t_2, \ldots, t_m, (p_m, c_m)$ or a walk (if it has sense)[1] we define the following attributes:

$$|\pi| = m, \qquad (\textit{length})$$
$$\mathsf{init.state}(\pi) = p_0, \qquad (\textit{initial control state})$$
$$\mathsf{final.state}(\pi) = p_m, \qquad (\textit{final control state})$$
$$\mathsf{init.counter}(\pi) = c_0, \qquad (\textit{initial counter value})$$
$$\mathsf{final.counter}(\pi) = c_m, \qquad (\textit{final counter value})$$
$$\mathsf{high}(\pi) = \max\{c_i \mid 0 \le i \le m\},$$
$$\mathsf{low}(\pi) = \min\{c_i \mid 0 \le i \le m\},$$
$$\mathsf{drop}(\pi) = \mathsf{init.counter}(\pi) - \mathsf{low}(\pi),$$
$$\mathsf{height}(\pi) = \mathsf{high}(\pi) - \mathsf{low}(\pi),$$
$$\mathsf{effect}(\pi) = \mathsf{final.counter}(\pi) - \mathsf{init.counter}(\pi).$$

We also use the following terms for quasi runs and walks:

- the *induced word* is $w = a_1 a_2 \ldots a_m \in \Sigma^*$ where $t_i = (p_{i-1}, a_i, s, p_i) \in \delta$ with $a_i \in \Sigma \cup \{\varepsilon\}$,
- the *Parikh image*, denoted $\psi(\pi)$, is the Parikh image of the induced word.

Note that a run of length 0 is a single configuration.

Definition 49 (concatenation)**.** The *concatenation* of a run π_1 and a quasi run π_2

$$\pi_1 = (p_0, c_0), t_1, (p_1, c_1), t_2, \ldots, t_m, (p_m, c_m) \text{ and}$$
$$\pi_2 = (q_0, \bar{c}_0), \bar{t}_1, (q_1, \bar{c}_1), \bar{t}_2, \ldots, \bar{t}_k, (q_k, \bar{c}_k)$$

where $p_m = q_0$ and $c_m \stackrel{\text{def}}{=} \mathsf{final.counter}(\pi_1) \ge \mathsf{drop}(\pi_2)$ is the sequence

$$\pi_3 = (p_0, c_0), t_1, (p_1, c_1), t_2, \ldots, t_m, (p_m, c_m), \bar{t}_1, (q_1, \bar{c}'_1), \bar{t}_2, \ldots, \bar{t}_k, (q_k, \bar{c}'_k)$$

where $\bar{c}'_i = \bar{c}_i - \bar{c}_0 + c_m$; note that this sequence π_3 is a run. The concatenation of π_1 and π_2 is denoted by $\pi_3 = \pi_1 \cdot \pi_2$; we also write $\pi_3 = \pi_1 \pi_2$ when we want no additional emphasis on this operation.

Note that in our definition of concatenation the value $\mathsf{final.counter}(\pi_1)$ can be different from $\mathsf{init.counter}(\pi_2)$, in which case the counter values all configurations in π_2 are adjusted accordingly. The condition $\mathsf{final.counter}(\pi_1) \ge \mathsf{drop}(\pi_2)$ ensures that all the adjusted values stay non-negative, i.e., that the concatenation $\pi_1 \cdot \pi_2$ is indeed a run.

We extend Definition 49 of concatenation to a concatenation of a run π_1 and a walk α. Let α be a sequence of transitions in some quasi-run π_2. The concatenation π_1 and α is allowed only if $\pi_1 \cdot \pi_2$ is well-defined, and the effect of $\pi_1 \cdot \alpha \stackrel{\text{def}}{=} \pi_1 \cdot \pi_2$, so it is a run. To sum up, we can concatenate runs, quasi-runs, and walks, using the notation $\pi_1 \cdot \pi_2$ and sometimes dropping the dot. If π_2 is a walk and π_1 is a run, then $\pi_1 \cdot \pi_2$ will also denote a run. In this and other cases, we will often assume that the counter values in π_2 are picked or adjusted automatically to match the last configuration of π_1. However, whenever we introduce parts of the run, e.g., by writing "suppose $\pi = \pi_1 \cdot \pi_2$", we always assume that π_2 is just a sub-run of π, that is, no implicit shifting occurs in this particular concatenation.

We say that a run π_2 is *in* π if $\pi = \pi_1 \pi_2 \pi_3$ for some runs π_1, π_3 and $\mathsf{final.counter}(\pi_1) = \mathsf{init.counter}(\pi_2)$, $\mathsf{final.counter}(\pi_3) = \mathsf{init.counter}(\pi_3)$.

We say that runs $\pi_1, \pi_2, \ldots, \pi_k$ are *disjoint* in π if $\pi = \pi'_1 \pi_1 \pi'_2 \pi_2 \pi'_3 \ldots \pi'_k \pi_k \pi'_{k+1}$ for some runs $\pi'_1, \pi'_2 \ldots \pi'_{k+1}$, where $\mathsf{final.counter}(\pi'_i) = \mathsf{init.counter}(\pi_{\mathsf{i}})$ and $\mathsf{final.counter}(\pi_i) = \mathsf{init.counter}(\pi'_{\mathsf{i}+1})$ for all $1 \le i \le k$.

C.2 Semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$

Definition 50 (split run)**.** A *split run* is a pair of runs (ρ, σ) such that $\mathsf{effect}(\rho) \ge 0$ and $\mathsf{effect}(\sigma) \le 0$.

In fact, we can even drop these inequalities in the definition, but we believe it's more visual this way.

Definition 51 (direction)**.** A *direction* is a pair of walks α and β, denoted $d = \langle \alpha, \beta \rangle$, such that:

- $\mathsf{init.state}(\alpha) = \mathsf{final.state}(\alpha)$,
- $\mathsf{init.state}(\beta) = \mathsf{final.state}(\beta)$,
- $0 < |\alpha| + |\beta| < n(2n^2 + 3)(n^3) + 1$,
- $0 \le \mathsf{effect}(\alpha) \le n^3$,
- $\mathsf{effect}(\alpha) + \mathsf{effect}(\beta) = 0$,
- if $\mathsf{effect}(\alpha) = 0$, then either $|\alpha| = 0$ or $|\beta| = 0$.

One can think of a direction as a pair of short loops with zero total effect on the counter. Pairs of words induced by these loops are sometimes known as iterative pairs. Directions of the first kind are essentially just individual loops; in a direction of the second kind, the first loop increases and the second loop decreases the counter value (even though the values $\mathsf{drop}(\alpha)$ and $\mathsf{drop}(\beta)$ are allowed to be strictly positive). The condition that $\mathsf{effect}(\alpha) \le n^3$ is a pure technicality and is only exploited at a later stage of the proof; in contrast, the upper bound $|\alpha| + |\beta| < n(2n^2 + 3)(n^3) + 1$ is crucial.

We also use the following terms:

- the Parikh image of a split run (ρ, σ) is $\psi((\rho, \sigma)) \stackrel{\text{def}}{=} \psi(\rho) + \psi(\sigma)$,
- similarly, the Parikh image of a direction $\langle \alpha, \beta \rangle$ is $\psi(\langle \alpha, \beta \rangle) \stackrel{\text{def}}{=} \psi(\alpha) + \psi(\beta)$,
- split runs $(\rho_1, \sigma_1) \ldots (\rho_k, \sigma_k)$ are *disjoint in* a run π iff $\rho_1, \sigma_1, \ldots, \rho_k, \sigma_k$ form disjoint subsequences of transitions in the sequence π;

Remark: the number of directions can be exponential, but the number of their Parikh images is at most $(n(2n^2 + 3)(n^3) + 1)^{|\Sigma|}$. Since $|\Sigma|$ is fixed, this is polynomial in n.

Definition 52 (availability of directions)**.** Suppose π is a run. A direction $d = \langle \alpha, \beta \rangle$ is *available* at π if there exists a factorization $\pi = \pi_1 \cdot \pi_2 \cdot \pi_3$ such that $\pi' = \pi_1 \cdot \alpha \pi_2 \beta \cdot \pi_3$ is also a run.

In the context of the definition above we write $\pi + d$ to refer to π'. This plus operation is non-commutative and binds from the left. Whenever we use this notation, we implicitly assume that the direction d is available at π.

Note that for a particular run π there can be more than one factorization of π into π_1, π_2, π_3 such that $\pi_1 \alpha \pi_2 \beta \pi_3$ is a valid run. In such cases the direction d can be introduced at different points inside π. In what follows we only use the notation $\pi + d$ to refer to a single run π' obtained in this way, without specifying a particular factorization of π. We

[1] To define attributes for walk we take a quasi run such that its sequence of transitions is equal to the walk. All attributes except of $\mathsf{init.counter}(), \mathsf{final.counter}(), \mathsf{high}(), \mathsf{low}()$ are well defined for walk, as their value is purely determined by a sequence of transitions.

ensure that all statements that refer to $\pi + d$ hold regardless of which factorization is chosen of π.

Lemma 53 (characterization of availability)**.** *A direction $\langle \alpha, \beta \rangle$ is available at a run π if and only if π has two configurations* $(\mathsf{init.state}(\alpha), c_1)$ *and* $(\mathsf{init.state}(\beta), c_2)$*, occurring in this particular order, such that $c_1 \geq \mathsf{drop}(\alpha)$ and $c_2 + \mathsf{effect}(\alpha) \geq \mathsf{drop}(\beta)$.*

Proof. Denote $p = \mathsf{init.state}(\alpha)$ and $q = \mathsf{init.state}(\beta)$. It is immediate that for a direction $\langle \alpha, \beta \rangle$ to be available, it is necessary that π have some configurations of the form (p, c_1) and (q, c_2) that occur in this order. Let $\pi = \pi_1 \pi_2 \pi_3$ where π_1 ends in (p, c_1) and π_2 ends in (q, c_2). We now show that the direction $\langle \alpha, \beta \rangle$ is available if and only if $c_1 \geq \mathsf{drop}(\alpha)$ and $c_2 + \mathsf{effect}(\alpha) \geq \mathsf{drop}(\beta)$.

We first suppose that these two inequalities hold. Observe that $\pi_1 \alpha$ is then a run as $c_1 \geq \mathsf{drop}(\alpha)$; furthermore, $\mathsf{final.counter}(\pi_1 \alpha) = c_1 + \mathsf{effect}(\alpha) \geq c_1$: by our definition of a direction, we have $\mathsf{effect}(\alpha) \geq 0$. Hence, $\mathsf{final.counter}(\pi_1 \alpha) \geq c_1 \geq \mathsf{drop}(\pi_2)$, where the last inequality holds because $\pi_1 \pi_2$ is a run. It follows that $\pi_1 \alpha \pi_2$ is also a run; we note that $\mathsf{final.counter}(\pi_1 \alpha \pi_2) = \mathsf{final.counter}(\pi_1 \pi_2) + \mathsf{effect}(\alpha) = c_2 + \mathsf{effect}(\alpha) \geq \mathsf{drop}(\beta)$. This, in turn, implies that $\pi_1 \alpha \pi_2 \beta$ is a run. Moreover $\mathsf{effect}(\alpha) = -\mathsf{effect}(\beta)$ by our definition of a direction, so $\mathsf{final.counter}(\pi_1 \alpha \pi_2 \beta) = \mathsf{final.counter}(\pi_1 \pi_2) = c_2$. Since $\pi_1 \pi_2 \pi_3$ is a run, we have $c_2 \geq \mathsf{drop}(\pi_3)$; hence, $\pi_1 \alpha \pi_2 \beta \pi_3$ is also a run.

Conversely, suppose $\pi_1 \alpha \pi_2 \beta \pi_3$ is a run. Recall that π_1 ends in (p, c_1); we conclude that $c_1 \geq \mathsf{drop}(\alpha)$. Also recall that in the run $\pi_1 \pi_2 \pi_3$ the fragment π_2 ends in (q, c_2); here $c_2 = \mathsf{final.counter}(\pi_1 \pi_2)$. But $\pi_1 \alpha \pi_2 \beta \pi_3$ is a run, so $\mathsf{final.counter}(\pi_1 \alpha \pi_2) \geq \mathsf{drop}(\beta)$; since $\mathsf{final.counter}(\pi_1 \alpha \pi_2) = \mathsf{final.counter}(\pi_1 \pi_2) + \mathsf{effect}(\alpha)$, we also conclude that $c_2 + \mathsf{effect}(\alpha) \geq \mathsf{drop}(\beta)$. This completes the proof. □

By $\mathrm{avail}(\pi)$ we denote the set of all directions available at π.

Lemma 54 (monotonicity of availability)**.** *If π is a run of a one-counter automaton and d is a direction available at π, then* $\mathrm{avail}(\pi) \subseteq \mathrm{avail}(\pi + d)$*.*

Proof. By Lemma 53 it suffices to show that for every pair of configurations $(p, c_1), (q, c_2)$ that appear in π in this particular order there is a pair of configurations $(p, c_1'), (q, c_2')$ that appear in $\pi' = \pi + d$, in this particular order, such that $c_1' \geq c_1$ and $c_2' \geq c_2$. Now, this claim is not difficult to substantiate. Suppose $d = \langle \alpha, \beta \rangle$. Indeed, for any decomposition of $\pi = \pi_1 \pi_2 \pi_3$ such that there is a $\pi' = \pi_1 \alpha \pi_2 \beta \pi_3$, define $\pi_1' = \pi_1$, $\pi_2' = (\mathsf{init.state}(\pi_2), \mathsf{init.counter}(\pi_2) + \mathsf{effect}(\alpha))\pi_2$, $\pi_3' = \pi_3$. Now π_2' is simply π_2 shifted up, as $\mathsf{effect}(\alpha) \geq 0$. Observe that now $\pi' = \pi_1' \alpha \pi_2' \beta \pi_3'$. Thus for any pair of configurations in π there is a corresponding pair of configurations with the needed properties in π'. This completes the proof.

□

Definition 55 (unpumping)**.** A run π' *can be unpumped* if there exist a run π and a direction d such that $\pi' = \pi + d$.

If additionally $\mathrm{avail}(\pi') = \mathrm{avail}(\pi)$, then we say that π' *can be safely unpumped*. Note that $\mathrm{avail}(\pi')$ is always a superset of $\mathrm{avail}(\pi)$ by Lemma 54.

Lemma 56 (safe unpumping lemma)**.** *Every accepting run π' of $\mathcal{A}$ of length greater than $n^2((2n^2 + 3)(n^3))^3$ can be safely unpumped.*

Lemma 56 is the key lemma in the entire Appendix C; we prove it in subsection C.4.

Recall that a set $A \subseteq \mathbb{N}^{|\Sigma|}$ is called *linear* if it is of the form $\mathrm{Lin}(b; P) \stackrel{\text{def}}{=} \{b + \lambda_1 p_1 + \ldots + \lambda_r p_r \mid \lambda_1, \ldots, \lambda_r \in \mathbb{N}, p_1, \ldots, p_r \in P\}$ for some vector $b \in \mathbb{N}^{|\Sigma|}$ and some finite set $P \subseteq \mathbb{N}^{|\Sigma|}$; this vector b is called the *base* and vectors $p \in P$ *periods*. A set $S \subseteq \mathbb{N}^d$ is called *semilinear* if it is a finite union of linear sets, $S = \cup_{i \in I} \mathrm{Lin}(b_i; P_i)$. Semilinear sets were introduced by FIXME in 1960s and have since received a lot of attention in formal language theory and its applications to verification. They are precisely the sets definable in Presburger arithmetic, the first-order theory of natural numbers with addition. Intuitively, semilinear sets are a multi-dimensional analogue of ultimately periodic sets in $\mathbb{N}$.

The following lemma characterizes the semilinear set $\psi(\mathcal{L}(\mathcal{A}))$ through sets of directions available at short accepting runs.

Lemma 57. *For any one-counter automaton $\mathcal{A}$, it holds that*

$$\psi(\mathcal{L}(\mathcal{A})) = \bigcup_{|\pi| \,\leq\, small_1} \mathrm{Lin}(\psi(\pi); \psi(\mathrm{avail}(\pi))),$$

where the union is taken over all runs of $\mathcal{A}$ of length at most $small_1 \stackrel{def}{=} n^2((2n^2 + 3)(n^3))^3$.

Proof. Start with the $\supseteq$ part. It suffices to show that, for any run π of $\mathcal{A}$ and any vector $v \in \mathrm{Lin}(\psi(\pi); \psi(\mathrm{avail}(\pi)))$, the set $\mathcal{L}(\mathcal{A})$ contains at least one word with Parikh image v. Indeed, take such a v and suppose that $v = v_0 + \sum_{i=1}^{m} \lambda_i v_i$, where the vector v_0 is the Parikh image of the word induced by the run π, vectors $v_1, \ldots, v_m$ are Parikh images of the words induced by some directions $d_1, \ldots, d_m$ available at π, and $\lambda_1, \ldots, \lambda_m$ are nonnegative integers. Lemma 54 ensures that we can form a run

$$\pi + \underbrace{d_1 + \ldots + d_1}_{\lambda_1 \text{ times}} + \ldots + \underbrace{d_m + \ldots + d_m}_{\lambda_m \text{ times}}.$$

This run induces a word accepted by $\mathcal{A}$, and the Parikh image of this word is v, as desired.

Now turn to the $\subseteq$ part and take some vector v in $\psi(\mathcal{L}(\mathcal{A}))$. This vector v is the Parikh image of a word in $\mathcal{L}(\mathcal{A})$, which is induced by some accepting run π_0 of $\mathcal{A}$. If the length of π_0 does not exceed $n^2((2n^2 + 3)(n^3))^3$, there is nothing to prove, so assume otherwise. By the main lemma 56, π_0 can be safely unpumped. This implies that $\pi_0 = \pi_1 + d_1$ for some direction d_1. Note that the length of π_1 is strictly less than the length of π_0; if it is greater than $n^2((2n^2 + 3)(n^3))^3$, then we apply the safe unpumping lemma again: $\pi_1 = \pi_2 + d_2$. We repeat the process until the length of the run drops to $n^2((2n^2 + 3)(n^3))^3$ or below:

$$\pi_0 = \pi_k + d_k + d_{k-1} + \ldots + d_1, \tag{2}$$

where $|\pi_k| \leq n^2((2n^2 + 3)(n^3))^3$. Take Parikh images of the words that are induced by the runs on both sides of (2): on the left-hand side, we obtain v; we claim that on the right-hand side we obtain a vector from $\mathrm{Lin}(\psi(\pi_k); \psi(\mathrm{avail}(\pi_k)))$. Indeed, recall that Lemma 56 guarantees that all these unpumpings are safe, i.e., $\mathrm{avail}(\pi_i) = \mathrm{avail}(\pi_{i-1})$ for $0 < i \leq k$. Since each direction d_i is available at the run π_i, it follows that all the directions $d_1, \ldots, d_k$ are available at π_k, so the Parikh image of the word induced by the run on the right-hand side of (2) indeed belongs to $\mathrm{Lin}(\psi(\pi_k); \psi(\mathrm{avail}(\pi_k)))$.

But, by our choice above, the run π_k has length at most $n^2((2n^2+3)(n^3))^3$. This concludes the proof. $\square$

Up to now we were focused on building some semilinear representation of $\psi(\mathcal{L}(\mathcal{A}))$. Our next goal is to improve this representation, before we start we introduce two useful notions. For a given vector v we define $\|v\|_\infty \stackrel{\text{def}}{=} \max\{v(a) \mid a \in \Sigma\}$ and $\|v\|_1 \stackrel{\text{def}}{=} \sum_{a\in\Sigma} v(a)$. For a set of vectors F by $\|F\|_\infty$ we denote $\max\{\|v\|_\infty \mid v \in F\}$. We also denote the cardinality of a finite set F by $\#F$.

The following lemma uses results from Huynh [24] and in Kopczyński and To [25], which rely on the Carathéodory theorem for cones in a multi-dimensional space. Essentially, the underlying idea is that if a vector is a linear combination of more than $|\Sigma|$ vectors in a $|\Sigma|$-dimensional space, then, by using linear dependencies, one can reduce this number to just $|\Sigma|$. For non-negative integer combinations, the situation is slightly more complicated, but the same idea can be carried through.

Lemma 58. *Let $S \subseteq \mathbb{N}^{|\Sigma|}$ be a semilinear set with representation $S = \bigcup_{i\in I} \mathrm{Lin}(c_i; P_i)$ and suppose $M \in \mathbb{N}$ is such that $\|c_i\|_\infty, \|P_i\|_\infty \le M$ for all $i \in I$. Then S also has a representation $S = \bigcup_{j\in J} \mathrm{Lin}(b_j; Q_j)$, where $\#J \le (M+1)^{\mathrm{poly}(|\Sigma|)}$ and for each $j \in J$ there exists an $i \in I$ such that the following conditions hold:*

- *$\mathrm{Lin}(b_j; Q_j) \subseteq \mathrm{Lin}(c_i; P_i)$,*
- *Q_j is a linearly independent subset of P_i, and*
- *$\|b_j\|_\infty \le (M+1)^{\mathrm{poly}(|\Sigma|)}$.*

Proof. The case $\#I = 1$ appears in Huynh [24, Lemma 2.8] and in Kopczyński and To [25, Theorem 6]. In these results, upper bounds on $\#J$ are only stated implicitly and only for $I = 1$. We will rely on results of Kopczyński and To, as it seems simpler to extract an explicit upper bound from their arguments. Their results can be stated as follows: If $S = \mathrm{Lin}(c_1, P_1;)$, then S has also a representation $S = \bigcup_{j\in J_1} \mathrm{Lin}(b_j; Q_j)$ such that for each $j \in J_1$ the following conditions hold:

- $\mathrm{Lin}(b_j; Q_j) \subseteq \mathrm{Lin}(c_1; P_1)$,
- Q_j is a linearly independent subset of P_1, and
- $\|b_j\|_\infty \le (M+1)^{\mathrm{poly}(|\Sigma|)}$.

In fact, the precise bound we get from [25, Theorem 6] is as follows:

$$\|b_j\|_\infty \le (2M+1)^{|\Sigma|} \cdot (M^{|\Sigma|} \cdot |\Sigma|^{\frac{|\Sigma|}{2}})^2.$$

Observe that if $S = \bigcup_{i\in I} \mathrm{Lin}(c_i, P_i;)$, then we can transfer the above result directly: $S = \bigcup_{i\in I} \bigcup_{j\in J_i} \mathrm{Lin}(b_j; Q_j) = \bigcup_{j\in J} \mathrm{Lin}(b_j; Q_j)$ and the above bounds are transfered as well.

So the only remaining part is to prove that $\#J \le (M+1)^{\mathrm{poly}(|\Sigma|)}$. To do this, we start from the observation that every element $j \in J$ is uniquely determined by the pair (b_j, Q_j), so $\#J$ is bounded by the cardinality of a set of all possible pairs (b, Q) where

- $b \in \mathbb{N}^{|\Sigma|}$ and $\|b\|_\infty \le (M+1)^{\mathrm{poly}(|\Sigma|)}$
- Q is an r-tuple of vectors $(v_1, \dots v_r)$ where $r \le |\Sigma|$ and each $v_i \in \mathbb{N}^{|\Sigma|}$ and $\|v_i\|_\infty \le M$.

Due to the above characterization, we can bound the number of possible pairs (b, Q) by

$$((M+1)^{\mathrm{poly}(|\Sigma|)})^{|\Sigma|} \cdot ((M+1)^{|\Sigma|})^{|\Sigma|},$$

which is again a polynomial in M, because $|\Sigma|$ is fixed. Thus, $\#J \le (M+1)^{(\mathrm{poly}(|\Sigma|)+|\Sigma|)\cdot|\Sigma|}$. This completes the proof. $\square$

Lemma 59. *For any one-counter automaton $\mathcal{A}$, it holds that*

$$\psi(\mathcal{L}(\mathcal{A})) = \bigcup_{|\pi| \le small_2} \mathrm{Lin}(\psi(\pi); \psi(D_\pi)),$$

where the union is taken over all runs of $\mathcal{A}$ of length at most $small_2 \stackrel{def}{=} poly(n^2((2n^2+3)(n^3))^3)$ and, for each π, D_π is a subset of $\mathrm{avail}(\pi)$ of cardinality at most $|\Sigma|$.

Proof. From Lemma 57 we know that

$$\psi(\mathcal{L}(\mathcal{A})) = \bigcup_{|\pi'| \le small_1} \mathrm{Lin}(\psi(\pi'); \psi(\mathrm{avail}(\pi'))).$$

If we apply to it Lemma 58 we get the desired statement. It is worth emphasizing that $D_\pi \subseteq \mathrm{avail}(\pi)$ due to the following reasoning:
First due to Lemma 58, $\psi(\pi) \in \mathrm{Lin}(\psi(\pi'); \psi(\mathrm{avail}(\pi')))$ for some π' and π can be obtained from π' by pumping some of directions in the set $\mathrm{avail}(\pi')$. Thus according to Lemma 58 the set $\mathrm{avail}(\pi') \subseteq \mathrm{avail}(\pi)$ and consequently (according to Lemma 58) as $D_\pi \subseteq \mathrm{avail}(\pi')$ we get $D_\pi \subseteq \mathrm{avail}(\pi)$. $\square$

C.3 Computing the semilinear representation

Below we state sub-procedures used in the algorithm.

Lemma 60. *For every fixed Σ there is a polynomial-time algorithm for the following task: given a one-counter automaton $\mathcal{A}$ over Σ, two configurations (q_1, c_1) and (q_2, c_2) and a vector $v \in \mathbb{N}^\Sigma$ with all numbers written in unary, decide if $\mathcal{A}$ has a run $\pi = (q_1, c_1) \to (q_2, c_2)$ with $\psi(\pi) = v$.*

Proof. Our algorithm leverages the unary representation of c_1, c_2, and components of v. Define $H_0 = \max(c_1, c_2) + \|v\|_1 + 1$. Observe that for any run $\pi = (q_1, c_1) \to (q_2, c_2)$ such that $\psi(\pi) = v$ the counter value stays below H_0, as in one move the counter can not be changed by more than 1 and the number of moves is bounded by $\|v\|_1$.

The algorithm constructs a multi-dimensional table that for *all* pairs of configurations (q_1', c_1'), (q_2', c_2') with $c_1', c_2' < H_0$ and all vectors $v' \in \{0, \dots, H_0\}^\Sigma$ keeps the information whether there exists a $w' \in \Sigma^*$ such that $\mathcal{A}$ has a run $(q_1', c_1') \xrightarrow{w'} (q_2', c_2')$ where the counter value stays below H_0 and $\psi(w') = v'$. The size of the table is at most $(|\mathcal{A}| \cdot H_0)^2 \cdot (H_0+1)^{|\Sigma|}$, which is polynomial in the size of the input for a fixed Σ.

The algorithm fills the entries of the table using dynamic programming. To begin with, runs whose Parikh image is the zero vector only connect pairs where $(q_1', c_1') = (q_2', c_2')$. Now take a non-zero vector $v' \in \{0, \dots, H_0\}^\Sigma$; a run $(q_1', c_1') \xrightarrow{w'} (q_2', c_2')$ with $\psi(w') = v'$ exists if and only if there exists an intermediate configuration $(\bar{q}, \bar{c})$ such that $(q_1', c_1) \xrightarrow{\bar{w}} (\bar{q}, \bar{c}) \xrightarrow{a} (q_2', c_2')$, where $\bar{w} \in \Sigma^*$, $a \in \Sigma$, and $\psi(\bar{w}) + \psi(a) = v'$; note that the vector $\psi(a)$ has 1 in exactly one component and 0 in other components, so $\psi(\bar{w}) \in \{0, \dots, H_0\}^\Sigma$ and $\|\psi(\bar{w})\|_1 = \|v'\|_1 - 1$. This completes the description of the algorithm. $\square$

Corollary 61. *For a fixed alphabet Σ the Parikh membership problem for (simple) one-counter automata is in* P*: there exists a polynomial-time algorithm that takes as an input a (simple) one-counter automaton $\mathcal{A}$ over Σ with n states and a vector $v \in \mathbb{N}^{|\Sigma|}$ with components written in unary, and outputs some accepting run π of $\mathcal{A}$ with $\psi(\pi) = v$ if such a run exists or "none" otherwise.*

Lemma 62. *For every fixed Σ there is a polynomial-time algorithm for the following task: given a one-counter automaton $\mathcal{A}$ over Σ, a sequence of $2|\Sigma|$ configurations $C \stackrel{def}{=} (q_1, c_1), (q_2, c_2) \ldots (q_{2|\Sigma|}, c_{2|\Sigma|})$ and a vector $v \in \mathbb{N}^{\Sigma}$ with all numbers written in unary, decide if $\mathcal{A}$ has an accepting run π such that $\psi(\pi) = v$ and π contains C as a subsequence.*

Proof. Observe, that if there is such a run π then configurations in C are cutting π into $2|\Sigma| + 1$ fragments. Each of these fragments have its own Parikh image $u_1, u_2 \ldots u_{2|\Sigma|+1}$ that sum up to v. Thus the procedure iterates through all possible partitions of v into $2|\Sigma| + 1$ vectors, and for each such partition it checks if there exists a required set of runs: between consecutive elements of C, a run from the initial configuration $(q_0, 0)$ to the first element of C and run form the last element of C to some accepting configuration. Each of those small test can be done in polynomial time using algorithm form Lemma 60. The crucial fact is that the number of possible partitions of v into $2|\Sigma|+1$ vectors is polynomial in $\|v\|_1$ as the dimension is fixed i.e. number of partitions is bounded by $(\|v\|_1^{|\Sigma|})^{2|\Sigma|+1}$. This implies that the presented procedure works in polynomial time. ☐

Lemma 63. *For every fixed Σ there is a polynomial-time algorithm for the following task: given a one-counter automaton $\mathcal{A}$ over Σ, a sequence of $2|\Sigma|$ configurations $C \stackrel{def}{=} (q_1, c_1), (q_2, c_2) \ldots (q_{2|\Sigma|}, c_{2|\Sigma|})$ and a vector v with all numbers written in unary, decide if in $\mathcal{A}$ there exist a direction $\langle \alpha, \beta \rangle$ and two configurations $(p_i, c_i), (p_j, c_j) \in C$ where $i \leq j$ such that:*

- $\psi(\langle \alpha, \beta \rangle) = v$,
- $\mathsf{init.state}(\alpha) = p_i$, $\mathsf{low}(\alpha) \leq c_i$ *and*
- $\mathsf{init.state}(\beta) = p_j$, $\mathsf{low}(\beta) \leq c_j + \mathsf{effect}(\alpha)$

Proof. First observe that there are $\binom{2+(2|\Sigma|)-1}{2}$ possibilities for a pair $(p_i, c_i), (p_j, c_j)$ of configurations, thus it suffice to provide a polynomial time procedure to check if there exist a direction $\langle \alpha, \beta \rangle$ for a given pair of configurations. Indeed we can iterate through all possible pairs and for each of them use the procedure.

Second if there is a direction $\langle \alpha, \beta \rangle$ then $\mathsf{effect}(\alpha) \leq \|v\|_1$; thus in order to check existence of $\langle \alpha, \beta \rangle$ we proceed as follows: for every $x \in \{0 \ldots \|v\|_1\}$ check if there are runs α from (p_i, c_i) to $(p_i, c_i + x)$ and β from $(p_j, c_j + x)$ to (p_j, c_j) such that $\psi(\alpha) + \psi(\beta) = v$.

This question is very close to the question considered in Lemma 60, the problem is that we don't know the partition of v into $\psi(\alpha)$ and $\psi(\beta)$.

However, observe that the number of possible partitions is polynomial (bounded by $\|v\|_1^{|\Sigma|} \cdot \|v\|_1^{|\Sigma|}$). Thus basically for every possible partition of v into v_1 and v_2 we check if there are runs α from (p_i, c_i) to $(p_i, c_i + x)$ and β from $(p_j, c_j + x)$ to (p_j, c_j) such that $\psi(\alpha) = v_1$ and $\psi(\beta) = v_2$. This can be done in polynomial time according to Lemma 60.

In conclusion we iterate though all pairs of configurations in C through all possible effects of α and all possible splittings of v into v_1 and v_2; for each such choice we check if there are runs α and β using algorithm from Lemma 60. The number of possible choices is polynomial and for each choice we execute a polynomial time procedure so the algorithm works in polynomial time. ☐

Lemma 64. *For every fixed Σ there is a polynomial-time algorithm for the following task: given a one-counter automaton $\mathcal{A}$ over Σ and vectors $v, v_1, \ldots, v_r \in \mathbb{N}^{\Sigma}$, $0 \leq r \leq |\Sigma|$, with all numbers written in unary, decide if $\mathcal{A}$ has an accepting run π and directions $d_1, \ldots, d_r$ available at π such that $\psi(\pi) = v$ and $\psi(d_i) = v_i$ for all i.*

Proof. First observe that if there is such a run π then according to Lemma 53 for every $v_i \in \{v_1 \ldots v_{|\Sigma|}\}$ there is a pair of configurations such that it makes available a direction d_i where $\psi(d_i) = v_i$. Thus for a set of vectors $\{v_1 \ldots v_{|\Sigma|}\}$ there is a sequence C of $2|\Sigma|$ configurations such that it is a subsequence of π and for every vector v_i there is a pair of configurations in C which makes some direction d_i available in sense of Lemma 53.

Second observation is that the counter value of any configuration C can not exceed $\|v\|_1$; thus the sequence C has to be an element of a family of $2|\Sigma|$-sequences of configurations bounded by $\|v\|_1$. The size of this family is at most $(|\mathcal{A}| \cdot \|v\|_1)^{2|\Sigma|}$.

From above we derive a following procedure. For every possible choice of the sequence C test:

- if there is an accepting run π that contains C as a subsequence and where $\psi(\pi) = v$,
- if for every $v_i \in \{v_1 \ldots v_{|\Sigma|}\}$ there is a pair of configurations in C which makes available (in sense of Lemma 53) some direction d_i such that $\psi(d_i) = v_i$.

The first item is handle by algorithm from Lemma 62, the second by Lemma 63.

The proposed algorithm is polynomial time as the number of possible C is polynomial in the size of the input and for each possible C we execute small number of times polynomial time algorithms from Lemmas 62 and 63. ☐

Lemma 65. *For a fixed alphabet Σ there exists a polynomial-time algorithm that takes as an input a simple one-counter automaton over Σ with n states and outputs (in unary) an integer $k \geq 0$, vectors $b_i \in \mathbb{N}^{|\Sigma|}$, and sets of vectors $P_i \subseteq \mathbb{N}^{|\Sigma|}$, $|P_i| \leq |\Sigma|$ for $1 \leq i \leq k$, such that*

$$\psi(\mathcal{L}(\mathcal{A})) = \bigcup_{1 \leq i \leq k} \mathrm{Lin}(b_i; P_i)$$

and the following property is satisfied: for each i there exists an accepting run π of $\mathcal{A}$ and directions $d_1, \ldots, d_r$ available at π such that $\psi(\pi) = b_i$ and $\psi(\{d_1, \ldots, d_r\}) = P_i$.

Proof. Proof bases on two Lemmas 59 and 64. From the first one we conclude that it suffices to characterize polynomially many linear sets l_i such that $\psi(\mathcal{L}(\mathcal{A})) = \bigcup_i l_i$. According to Lemma 59 for each linear set $l_i = \mathrm{Lin}(b_i; Q_i)$ holds:

- there exist an accepting run π_i such that $\psi(\pi_i) = b_i$ and $\|b_i\|_1 \leq poly(n^2((2n^2 + 3)(n^3))^3)$,
- the number of elements of Q_i is bounded by $|\Sigma|$
- for every $v_j \in Q_i$ there exist a direction $d_j \in \mathrm{avail}(\pi)$, such that $\psi(d_j) = v_j$.

The last bullet point combined with Definition 51 of direction gives upper-bound on the $v_j \in Q_i$ for any i, precisely $\|v_j\|_1 \leq n(2n^2+3)(n^3)+1$.

Thus in order to compute the linear sets that characterize $\psi(\mathcal{L}(\mathcal{A}))$ we iterate through all possible vectors for b, where $\|b\|_1 \leq poly(n^2((2n^2+3)(n^3))^3)$, and all possible sets Q, where $|Q| \leq |\Sigma|$, and for every $v \in Q$ hold $\|v\|_1 \leq n(2n^2+3)(n^3)+1$. For each combination we check independently if $\mathrm{Lin}(b;Q)$ satisfies three bullet points. Second bullet point is satisfied by the definition. To check first and third we use the algorithm proposed in Lemma 64.

To show that above procedure terminates in polynomial time we observe that

- number of possible choices for b and Q is bounded by $poly(n^2((2n^2+3)(n^3))^3)^{|\Sigma|} \cdot ((n(2n^2+3)(n^3)+1)^{|\Sigma|})^{|\Sigma|}$,
- length of description of b and Q in unary encoding is polynomial in $\mathcal{A}$ (for example can be bounded by number of possible choices).
- Algorithm from Lemma 64 terminates in time polynomial in the input and as input is polynomial in $|\mathcal{A}|$ then it terminates in time polynomial in $|\mathcal{A}|$. ☐

Lemma 66. *For a fixed alphabet Σ there exist a polynomial time algorithm that takes as an input a one-counter automaton over Σ with n states and returns a Parikh-equivalent NFA.*

Proof. Observe that it suffice to show how to change one linear set to NFA; indeed in the end we can take union of automata designed for polynomialy many linear sets l_i. To build NFA that accepts a language Parikh equivalent to $l_i = \mathrm{Lin}(b;Q)$ we start form building an automaton that accepts only one word which is Parikh equivalent to b. Next to the unique accepting state we add one loop for each $v_i \in Q$. Word that can be read along $i-th$ loop is Parikh equivalent to $v_i \in Q$. It is easy to see that such automaton accepts a language Parikh equivalent with $\mathrm{Lin}(b;l_i)$. ☐

Note that DFA instead of NFA would not suffice for this construction, because even transforming unary NFA into unary NFA induces a super-polynomial blowup (a standard example has several cycles whose lengths are different prime numbers, with lcm (least common multiple) of super-polynomial magnitude).

C.4 Proof of the main lemma

In this subsection we prove Lemma 56. We consider two cases, depending on whether the height (largest counter value) of π' exceeds a certain polynomial in n. The strategy of the proof is the same for both cases (although the details are somewhat different).

Lemma 67. *Every accepting run π' of height greater than $(2n^2+3)(n^3)$ can be safely unpumped.*

Lemma 68. *Every accepting run π' of height at most $(2n^2+3)(n^3)$ and length greater than $n^2((2n^2+3)(n^3))^3$ can be safely unpumped.*

We first show that sufficiently large parts (runs or split runs) of π' can always be unpumped (as in standard pumping arguments). We notice that for such an unpumping to be *unsafe*, it is necessary that the part contain a configuration whose removal shrinks the set of available directions—a reason for non-safety; this *important* configuration cannot appear anywhere else in π'. We prove that the total number of important configurations is at most poly(n). As a result, if we divide the run π' into sufficiently many sufficiently large parts, at least one of the parts will contain no important configurations and, therefore, can be unpumped safely.

C.4.1 High runs: Proof of Lemma 67

The idea behind bounding the height of runs bases on two concepts. First is that if a run is high then there is a direction in it that can be unpumped. Second is that if a given run is even higher then there are a lot of different directions which can be unpumped and among them at least one can be unpumped in a safe way. As unpumping intuitively reduces the hight of a run, then iterative application of it lead to a path of bounded hight.

Claim 69. *Let (ρ, σ) be a split run such that $\mathsf{effect}(\rho) \geq n^3$ and $-\mathsf{effect}(\sigma) \geq n^3$, then $(\rho,\sigma) = (\rho_1\alpha\rho_2, \sigma_1\beta\sigma_2)$ such that $\langle\alpha,\beta\rangle$ is a direction.*

Proof. Let ρ' and σ' be a pair of sub-runs of ρ and σ, respectively, such that $|\rho'| = |\sigma'| = n^3$; such sub-runs exists because $\mathsf{effect}(\rho) \geq n^3$ and $-\mathsf{effect}(\sigma) \geq n^3$. Consider three possibilities:

1) there is a non-empty walk α such that $\rho' = \rho'_1\alpha\rho'_2$ that starts and ends in the same configuration, i.e. $\mathsf{final.counter}(\rho'_1) = \mathsf{final.counter}(\rho'_1\alpha)$ and $\mathsf{init.state}(\alpha) = \mathsf{final.state}(\alpha)$;
2) there is a non-empty walk β such that $\sigma' = \sigma'_1\beta\sigma'_2$ that starts and ends in the same configuration, i.e. $\mathsf{final.counter}(\sigma'_1) = \mathsf{final.counter}(\sigma'_1\beta)$ and $\mathsf{init.state}(\beta) = \mathsf{final.state}(\beta)$;
3) $\mathsf{effect}(\rho') \geq n^2$ and $-\mathsf{effect}(\sigma') \geq n^2$.

(Note that at least one of these three statements must hold, because, for example, the inequality $\mathsf{effect}(\rho') < n^2$ implies that the run ρ' traverses at most $n^2 \cdot |Q| = n^3$ different configurations; however, $|\rho'| = n^3$ implies that the total number of configurations that ρ' traverses is n^3+1. Hence, by the pigeonhole principle the run ρ' should traverse some configuration at least twice—which is the first possibility in the list above.) For each of these three possibilities, we now show how to find some direction $\langle\alpha,\beta\rangle$ inside the split run (ρ',σ').

Consider the first possibility, the direction $\langle\alpha,\varepsilon\rangle$ suffices for our purposes. Indeed, it is a direction of the first kind as $\mathsf{effect}(\alpha) = \mathsf{effect}(\varepsilon) = 0$ and $|\varepsilon| = 0$; the reader will easily check that all the conditions in the definition of a direction (Definition 51) are satisfied. The second possibility is completely analogous: the direction has the form $\langle\varepsilon,\beta\rangle$.

Now consider the third possibility. First, for every $i \in \{0,\ldots,n^2\}$ pick one configuration in the run ρ' with the counter value $\mathsf{init.counter}(\rho') + i$; call these configurations *red*. As there are n^2+1 red configurations, at least $n+1$ of them have the same state; we call these $n+1$ configurations *blue*. The corresponding indices i (for which the selected red configuration is also blue) are called blue too. Now for every blue i we pick in the other run, σ', some configuration with the counter value equal to $\mathsf{init.counter}(\sigma') - i$; we call these $n+1$ configurations *green*. Among green configurations there are at least two with the same state, say for $i = i_1$ and $i = i_2$. Let σ_β be the run between them contained in σ'; now β is a walk induced by σ_β. But for α we can, in turn, take a walk induced by a run between blue configurations with indices $i = i_1$ and $i = i_2$. By construction, $\mathsf{effect}(\alpha) = -\mathsf{effect}(\beta) > 0$, and it is easy to check that $\langle \alpha, \beta \rangle$ is indeed a direction of the second kind. This completes the proof. $\square$

Definition 70 (promising split run)**.** A split run (ρ, σ) in the run $\pi' = \pi_1 \rho \pi_2 \sigma \pi_3$ is *promising* if $\mathsf{low}(\rho \pi_2 \sigma) \geq n^3$, $\mathsf{effect}(\rho) \geq n^3$, and $-\mathsf{effect}(\sigma) \geq n^3$.

Definition 71 (unpumping a split run)**.** A split run (ρ', σ') in an accepting run $\pi' = \pi_1 \cdot \rho' \cdot \pi_2 \cdot \sigma' \cdot \pi_3$ *can be unpumped* if there exist a split run (ρ, σ) and a direction d such that the following conditions hold:

- $\rho' = \rho_1 \alpha \rho_2$ for some runs ρ_1, ρ_2,
- $\sigma' = \sigma_1 \beta \sigma_2$ for some runs σ_1, σ_2,
- $\pi = \pi_1 \cdot \rho_1 \rho_2 \cdot \pi_2 \cdot \sigma_1 \sigma_2 \cdot \pi_3$ is an accepting run.

One can conclude in such a case that $\pi' = \pi + d$.

Claim 72. *Any promising split run in an accepting run π' can be unpumped.*

Proof. Let $\pi' = \pi_1 \rho \pi_2 \sigma \pi_3$ where the split run (ρ, σ) is promising. Note that by the definition of a promising split run, (ρ, σ) satisfies the conditions of Claim 69. Therefore,

$$\pi' = \pi_1 \cdot \rho_1 \alpha \rho_2 \cdot \pi_2 \cdot \sigma_1 \beta \sigma_2 \cdot \pi_3$$

where $\langle \alpha, \beta \rangle$ is a direction, so it remains to prove that

$$\pi = \pi_1 \cdot \rho_1 \rho_2 \cdot \pi_2 \cdot \sigma_1 \sigma_2 \cdot \pi_3$$

is, first, a run and, second, an accepting run. It is straightforward to see that in this new concatenation the control states match, so it suffices to check that the counter values in π stay non-negative; by Definition 49, the (necessary and) sufficient condition for this is that the $\mathsf{drop}(\cdot)$ of each subsequent run does not exceed the $\mathsf{final.counter}(\cdot)$ of the prefix. To simplify notation, we denote $\pi_2' \stackrel{\text{def}}{=} \rho_2 \pi_2 \sigma_1$.

As $\pi_1 \cdot \rho_1$ is a prefix of π', we know that it is a run; the first thing that has to be checked is that $\pi_1 \cdot \rho_1 \pi_2'$ is a run. This holds if

$$\mathsf{final.counter}(\pi_1 \cdot \rho_1) \geq \mathsf{drop}(\pi_2'), \text{ i.e., if}$$
$$\mathsf{final.counter}(\pi_1 \cdot \rho_1) - \mathsf{drop}(\pi_2') \geq 0.$$

We have

$$\begin{aligned}
&\mathsf{final.counter}(\pi_1 \cdot \rho_1) - \mathsf{drop}(\pi_2') \\
&= \mathsf{final.counter}(\pi_1 \cdot \rho_1 \alpha) - \mathsf{effect}(\alpha) - \mathsf{drop}(\pi_2') \\
&= \mathsf{final.counter}(\pi_1 \cdot \rho_1 \alpha) - \mathsf{effect}(\alpha) - \mathsf{drop}(\rho_2 \cdot \pi_2 \cdot \sigma_1) \\
&= \mathsf{low}(\rho_2 \cdot \pi_2 \cdot \sigma_1) - \mathsf{effect}(\alpha) \\
&\geq \mathsf{low}(\rho \cdot \pi_2 \cdot \sigma) - \mathsf{effect}(\alpha) \\
&\geq n^3 - \mathsf{effect}(\alpha) \\
&\geq n^3 - n^3 = 0,
\end{aligned}$$

where the equalities and inequalities follow from Definitions 48 and 51 and from the conditions of the claim. Hence, $\pi_1 \cdot \rho_1 \pi_2'$ is a run.

Since $\mathsf{effect}(\alpha) = -\mathsf{effect}(\beta)$, the equality

$$\mathsf{final.counter}(\pi_1 \cdot \rho_1 \pi_2') = \mathsf{final.counter}(\pi_1 \cdot \rho_1 \alpha \pi_2' \beta)$$

holds. Therefore, as $\pi_1 \rho_1 \alpha \pi_2' \beta \sigma_2 \pi_3 = \pi'$ is a run, $\pi_1 \rho_1 \pi_2' \sigma_3 \pi_3 = \pi$ is a run too. Moreover,

$$\mathsf{final.counter}(\pi) = \mathsf{final.counter}(\pi'),$$

so π is an accepting run, for π' is accepting as well. This completes the proof. $\square$

Definition 73 (state fingerprint)**.** Let τ be a run. The *state fingerprint* of τ is the set of all pairs $(q_1, q_2) \in Q \times Q$ such that τ contains configurations $c_1 = (q_1, r_1)$ and $c_2 = (q_2, r_2)$ for some r_1 and r_2, and, moreover, there exists at least one occurrence of c_1 before (possibly coinciding with) some occurrence of c_2.

Claim 74. *Suppose*

$$\pi' = \pi_1 \pi_2 \cdot \alpha \pi_3 \beta \cdot \pi_4 \pi_5 \textit{ and}$$
$$\pi = \pi_1 \pi_2 \cdot \pi_3 \cdot \pi_4 \pi_5$$

are accepting runs and $\langle \alpha, \beta \rangle$ is a direction, so that $\pi' = \pi + \langle \alpha, \beta \rangle$. Also suppose that

$$\mathsf{low}(\pi_2 \cdot \alpha \pi_3 \beta \cdot \pi_4) \geq 2(n^3).$$

If the runs $\pi_2 \cdot \alpha \pi_3 \beta \cdot \pi_4$ and $\pi_2 \cdot \pi_3 \cdot \pi_4$ have identical state fingerprints, then $\text{avail}(\pi') = \text{avail}(\pi)$.

Proof. Since $\langle \alpha, \beta \rangle$ is a direction, we have $\mathsf{effect}(\alpha) = -\mathsf{effect}(\beta) \leq n^3$. Thus for all configurations observed along $\pi_2 \pi_3 \pi_4$ their counter values are at least $2(n^3) - \mathsf{effect}(\alpha) \geq n^3$. Now we can use our characterization of availability (Lemma 53). Let a direction d be available in π' due to a pair of configurations $(q_1, c_1), (q_2, c_2)$. We have to consider three cases depending on where $(q_1, c_1), (q_2, c_2)$ are: both configurations are in parts π_1, π_5, both configurations are in $\pi_2 \alpha \pi_3 \beta \pi_4$ and one is in π_1 or π_5 and the second in $\pi_2 \alpha \pi_3 \beta \pi_4$. In first case exactly the same pair of configurations can be found in π. In the second case due to the assumption about equality of state fingerprints we can find a pair $(q_1, c_1'), (q_2, c_2')$ where $c_1', c_2' \geq n^3 \geq \mathsf{drop}(\alpha), \mathsf{drop}(\beta)$ so d is available in π. In the third case we have to combine both previous cases. There are two symmetric situations, first if the pair of configurations (q_1, c_1) is in π_1 and (q_2, c_2) in $\pi_2 \alpha \pi_3 \beta \pi_4$ or (q_1, c_1) is in $\pi_2 \alpha \pi_3 \beta \pi_4$ and (q_2, c_2) is in π_5. Here we consider only the first one of them, the second is analogous. We need to find a pair of configurations in π that witnesses the availability of d. The first element of the pair is the same (q_1, c_1) that can be found in π_1 as a subrun of π'. To find the second element, consider this configuration (q_2, c_2) in π'; it occurs in $\pi_2 \alpha \pi_3 \beta \pi_4$, so the state fingerprint contains the pair (q_2, q_2), simply by definition. Thus, in $\pi_2 \pi_3 \pi_4$ we can find a configuration (q_2, c_3) such that $c_3 \geq n^3$. Now this moves us from the pair of configurations $(q_1, c_1), (q_2, c_2)$ to the pair $(q_1, c_1), (q_2, c_3)$, which completes the proof. $\square$

Claim 75. *Let $\pi = \pi_1 \pi_2 \pi_3$ be an accepting run. Suppose the run π_2 satisfies $\mathsf{low}(\pi_2) \geq 2(n^3)$ and contains $2n^2 + 1$ pairwise disjoint promising split runs. Then π can be safely unpumped.*

Proof. From Claim 72 we know that each of these split runs can be unpumped; so the accepting run π can be unpumped in at least $2n^2 + 1$ different ways. Furthermore, by Claim 74, if the state fingerprint of π_2 after unpumping one of these split runs does not change, then this unpumping is safe. Hence, it remains to show that such an unpumping indeed exists.

For any pair of states in the state fingerprint of π_2, *mark* two configurations of π_2 that witness this pair (in sense of Lemma 53). In total, at most $2n^2$ configurations of π_2 will be marked. Since π_2 contains at least $2n^2 + 1$ disjoint promising split runs, there is a promising split run in π_2 that contains no marked configuration. Our choice of this split run ensures that its unpumping will not change the state fingerprint of π_2: every pair of states in the state fingerprint of π_2 is witnessed by a pair of configurations outside this split run. This completes the proof. ☐

Claim 76. *Let π be a run of height at least $(2n^2+3)(n^3)$. Then $\pi = \pi_1\pi_2\pi_3$ for some runs π_1, π_2, π_3 such that $\mathsf{low}(\pi_2) \geq 2(n^3)$ and π_2 contains at least $2n^2+1$ pairwise disjoint promising split runs.*

Proof. First factorize π as $\pi = \pi_1\pi_2\pi_3$ where $\mathsf{init.counter}(\pi_2) = 2(n^3)$, $\mathsf{final.counter}(\pi_2) = 2(n^3)$ and $\mathsf{low}(\pi_2) = 2(n^3)$. We can now split π_2 into two parts, $\pi_2 = \pi_2'\pi_2''$, so that $\mathsf{final.counter}(\pi_2') = \mathsf{init.counter}(\pi_2'') = \mathsf{high}(\pi_2)$. Let ρ_i for $0 \leq i \leq 2n^2+1$ be the sub-run of π_2' which starts at the last configuration of π_2' with counter value $2(n^3) + i \cdot (n^3) + 1$ and ends at the last configuration with counter value $2(n^3) + (i+1) \cdot (n^3)$. Similarly, let σ_i be the sub-run of π_2'' which ends at the last configuration with counter value $2(n^3) + i \cdot (n^3) + 1$ and starts at the last configuration with counter value $2(n^3) + (i+1) \cdot (n^3)$.

It is obvious that $(\rho_i, \sigma_i) \cap (\rho_j, , \sigma_j) = \emptyset$ for $i \neq j$ and that (ρ_i, σ_i) is a promising split run for any $i \leq 2n^2+1$. ☐

Lemma 67 follows from Claims 75 and 76.

C.4.2 Low runs: Proof of Lemma 68

The idea behind bounding the length of low runs (not high runs) bases on similar concept as bounding the hight of runs. First we show that long enough low run can be unpumped and next that if the run is even longer then there are a lot different directions that can be unpumped independently; finally one of them can be unpumped in a safe way. Unpumping makes a run shorter so any low run that can not be unpumped anymore can not be very long.

Definition 77 (promising run)**.** A run τ is *promising* if $\mathsf{high}(\tau) < (2n^2+3)(n^3)$ and $|\tau| > n(2n^2+3)(n^3)+1$.

Claim 78. *Any promising run τ can be unpumped.*

Proof. As the counter value in the run τ is bounded by $(2n^2+3)(n^3)$ and the length of τ is at least $n(2n^2+3)(n^3)+1$ then among first $n(2n^2+3)(n^3)+1$ configurations at least one configuration has to repeat. Thus $\tau = \pi_1\alpha\pi_2$ where $\mathsf{init.state}(\alpha) = \mathsf{final.state}(\alpha)$ and $\mathsf{init.counter}(\alpha) = \mathsf{final.counter}(\alpha)$. As a result, the fragment α between two occurrences of this repeating configuration can be unpumped, so the outcome of this operation is a run $\pi_1 \cdot \pi_2$. Note that the unpumped direction is of the form $\langle \alpha, c \rangle$, where c is path of length zero (a configuration that occurs to the right of α). In this case the $\mathsf{effect}(\alpha) = 0$ and it does not violates the upper bound on the length. ☐

Now, instead of the state fingerprint introduced in the previous subsubsection C.4.1, we will use configuration fingerprint, defined as follows.

Definition 79 (configuration fingerprint)**.** Let τ be a run. The *configuration fingerprint* of τ is the set of all pairs of configurations (c_1, c_2) such that in τ there exists at least one occurrence of c_1 before (possibly coinciding with) some occurrence of c_2.

Remark 80. Suppose τ is an accepting run of $\mathcal{A}$ of height at most $(2n^2+3)(n^3)$, then the cardinality of its configuration fingerprint does not exceed $\mathrm{poly}_1 \stackrel{\mathrm{def}}{=} n^2((2n^2+3)(n^3))^2$.

Claim 81. *If runs τ and τ' have identical configuration fingerprints, then* $\mathrm{avail}(\tau') = \mathrm{avail}(\tau)$.

Claim 81 follows from Lemma 53.

Claim 82. *Any accepting run τ that satisfies $\mathsf{high}(\tau) \leq (2n^2+3)(n^3)$ and contains, as sub-runs, $2(\mathrm{poly}_1)$ pairwise disjoint promising runs, can be safely unpumped.*

Proof. Let S be the configuration fingerprint of τ. For every element of $x \in S$ we choose two configurations that witnesses x. Observe that set of chosen configurations is of size at most $2 \cdot |S| \leq 2 \cdot (\mathrm{poly}_1)$; thus there exist a promising run σ which does not contain any chosen configuration. Now unpumping σ does not change set of configurations fingerprint, so it is safe unpunping. ☐

Claim 83. *In any accepting run τ which is a promising run of length at least $2\mathrm{poly}_1 \cdot (n(2n^2+3)(n^3)+1)$ there exist at least $2\mathrm{poly}_1$ pairwise disjoint promising runs.*

Proof. Immediate consequence of Claim 78. ☐

Lemma 68 follows from Claims 82 and 83.

D. Parikh image: Unbounded alphabet

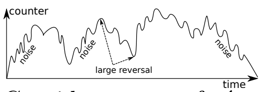
In this section we describe an algorithm to construct an NFA Parikh-equivalent to an OCA $\mathcal{A}$ without assumptions $|\Sigma|$. The NFA has at most $O(|\Sigma| K^{O(\log K)})$ states where $K = |\mathcal{A}|$, a significant improvement over $O(2^{\mathrm{poly}(K,|\Sigma|)})$ for PDA.

We establish this result in two steps. In the first step, we show that we can focus our attention on computing Parikh-images of words recognized along *reversal bounded* runs. A reversal in a run occurs when the OCA switches to incrementing the counter after a non-empty sequence of decrements (and internal moves) or when it switches to decrementing the counter after a non-empty sequence of increments (and internal moves). For a number R, a run is R reversal bounded, if the number of reversals along the run is $\leq R$. Let us use $\mathcal{L}_R(\mathcal{A})$ to denote the set of words accepted by $\mathcal{A}$ along runs with at most R reversals.

We construct a new polynomial size simple OCA from $\mathcal{A}$ and show that we can restrict our attention to runs with at most R reversals of this OCA, where R is a polynomial in K. In the second step, from any simple OCA $\mathcal{A}$ with K states and any integer R we construct an NFA of size $O(K^{O(log(R))})$ whose Parikh image is $\mathcal{L}_R(\mathcal{A})$. Combination of the two steps gives a $O(K^{O(logK)})$ construction.

D.1 Reversal bounding

We establish that, up to Parikh-image, it suffices to consider runs with $2K^2 + K$ reversals. We use two constructions: one that eliminates *large* reversals (think of a waveform) and another that eliminates *small* reversals (think of the noise on a noisy waveform). For the large reversals, the idea used is the following: we can reorder the transitions used along a run, hence preserving Parikh-image, to turn it into one with few large reversals (a noisy waveform with few reversals). The key idea used is to move each simple cycle at state q with a positive (resp. negative) effect on the counter to the first (resp. last) occurrence of the state along the run. To eliminate the smaller reversals (noise), the idea is to maintain the changes to the counter in the state and transfer it only when necessary to the counter to avoid unnecessary reversals.



Consider a run of $\mathcal{A}$ starting at a configuration (p, c) and ending at some configuration (q, d) such that the value of the counter e in any intermediate configuration satisfies $c - D \leq e \leq c + D$ (where D is some positive integer). We refer to such a run as an *D-band run*. Reversals along such a run are not important and we get rid of them by maintaining the (bounded) changes to the counter within the state.

We construct a simple OCA $\mathcal{A}[D]$ as follows: its states are $Q \cup Q_1 \cup Q_2$ where $Q_1 = Q \times [-D, D]$ and $Q_2 = [-D, D] \times Q$. All transitions of $\mathcal{A}$ are transitions of $\mathcal{A}[D]$ as well and thus using Q it can simulate any run of $\mathcal{A}$ faithfully. From any state $q \in Q$ the automaton may move nondeterministically to $(q, 0)$ in Q_1. The states in Q_1 are used to simulate D-band runs of $\mathcal{A}$ without altering the counter and by keeping track of the net change to the counter in the second component of the state. For instance, consider the D-band run of $\mathcal{A}$ described above: $\mathcal{A}[D]$ can move from (p, c) to $((p, 0), c)$ then simulate the run of $\mathcal{A}$ to (q, d) to reach $((q, d - c), c)$. At this point it needs to transfer the net effect back to the counter (by altering it appropriately). The states Q_2 are used to perform this role. From a state (q, j) in Q_1, $\mathcal{A}[D]$ is allowed to nondeterministically move to (j, q) indicating that it will now transfer the (positive or negative) value j to the counter. After completing the transfer it reaches a state $(0, q)$ from where it can enter the state q via an internal move to continue the simulation of $\mathcal{A}$.

The nice feature of this simulated run via Q_1 and Q_2 is that there are no reversals in the simulation and it involves only increments (if $d > c$) or only decrements (if $d < c$). We now formalize the automaton $\mathcal{A}[D]$ and its properties. The simple OCA $\mathcal{A}[D] = (Q_D, \Sigma, \delta_D, s, F)$ is defined as follows:

$$Q_D = Q \cup (Q \times \{-D, \ldots, D\}) \cup (\{-D, \ldots, D\} \times Q)$$

and δ_D is defined as follows:

1. $\delta \subseteq \delta_R$. Simulate runs of $\mathcal{A}$.
2. $(q, \varepsilon, i, (q, 0)) \in \delta_D$. Begin a summary phase.
3. $((q, j), a, i, (q', j)) \in \delta_D$, if $(q, a, i, q') \in \delta$. Simulate an internal move.
4. $((q, j), a, i, (q', j + 1)) \in \delta_D$, if $(q, a, +1, q') \in \delta$. Simulate an increment.
5. $((q, j), a, i, (q', j - 1)) \in \delta_D$, if $(q, a, -1, q') \in \delta$. Simulate a decrement.
6. $((q, j), \varepsilon, i, (j, q)) \in \delta_D$. Finish summary run.
7. $((j, q), \varepsilon, +1, (j - 1, q)) \in \delta_D$, if $j > 0$. Transfer a positive effect.
8. $((j, q), \varepsilon, -1, (j + 1, q)) \in \delta_D$, if $j < 0$. Transfer a negative effect.
9. $((0, q), \varepsilon, i, q) \in \delta_D$. Transfer control back to copy of $\mathcal{A}$.

The following Lemma is the first of a sequence that relate $\mathcal{A}$ and $\mathcal{A}[D]$.

Lemma 84. *1. For any $p, q \in Q$ and any $c, d \in \mathbb{N}$, if $(p, c) \xrightarrow{w} (q, d)$ in $\mathcal{A}$ then $(p, c) \xrightarrow{w} (q, d)$ in $\mathcal{A}[D]$.*
2. For any $p, q \in Q$ and any $c, d \in \mathbb{N}$ if $(p, c) \xrightarrow{w} (q, d)$ in $\mathcal{A}[D]$ then $(p, c + D) \xrightarrow{w} (q, d + D)$ in $\mathcal{A}$. In particular, if $(p, 0) \xrightarrow{w} (q, 0)$ in $\mathcal{A}[D]$ then $(p, D) \xrightarrow{w} (q, D)$ in $\mathcal{A}$.

Proof. The first statement simply follows from the fact that $\delta \subseteq \delta_D$.

Let $\rho = (p, c) \xrightarrow{w} (q, d)$ be a run in $\mathcal{A}[D]$. The second statement is proved by induction on the number of transitions of type 2 taken along ρ (i.e. the number of summary simulations used in ρ). If this number is 0 then all the transitions used are of type 1 thus ρ is a run in $\mathcal{A}$ and thus $\rho[D]$ satisfies the requirements of the Lemma.

Otherwise, let ρ must be of the form

$$\rho = (p, c) \xrightarrow{w_1} (p_1, c_1) \xrightarrow{\varepsilon, i} ((p_1, 0), c_1) \xrightarrow{w_2} ((0, q_1), d_1) \xrightarrow{\varepsilon, i} (q_1, d_1) \xrightarrow{w_3} (q, d)$$

where we have identified the first occurrence of the transition of type 2 and as well as the first occurrence of a transition of type 9. Now, by the induction hypothesis, we have runs $(p, c + D) \xrightarrow{w_1} (p_1, c_1 + D)$ and $(q_1, d_1 + D) \xrightarrow{w_3} (q, d + D)$ in $\mathcal{A}$.

From the definition of δ_D, run $((p_1, 0), c_1) \xrightarrow{w_2} ((0, q_1), d_1)$ must be of the form

$$((p_1, 0), c_1) \xrightarrow{w_2} ((p_2, c_2), c_1) \xrightarrow{\varepsilon} ((c_2, p_2), c_1) \xrightarrow{\varepsilon} ((0, p_2), c_1 + c_2)$$

with $p_2 = q_1$ and $d_1 = c_1 + c_2$ and where the run $((p_1, 0), c_1) \xrightarrow{w_2} ((p_2, c_2), c_1)$ involves only transitions of the form 3, 4 or 5.

Claim: Let $((g, 0), e) \xrightarrow{x} ((h, i), e)$ be a run in $\mathcal{A}[D]$ using only transitions of type 3, 4 or 5. Then $(g, e) \xrightarrow{x} (h, e + i)$ in $\mathcal{A}$ for any $e \geq D$.

Proof of Claim: By induction on the length of the run. The base case is trivial. For the inductive case, suppose $((g, 0), e) \xrightarrow{x'} ((h', i'), e) \xrightarrow{a} ((h, i), e)$ and by the induction hypothesis $(g, e) \xrightarrow{x'} (h', e + i')$ for any $e \geq D$. Now, if the last transition is an internal transition then, $(h', a, i, h) \in \delta$ and $i = i'$. Thus $(h', e + i) \xrightarrow{a} (h, e + i)$ in $\mathcal{A}$. If the last transition is an increment then $(h', a, +1, h) \in \delta$ and $i = i' + 1$. Thus, once again we have $(h', e + i) \xrightarrow{a} (h, e + i)$ in $\mathcal{A}$. Finally, if the last transition is a decrement transition then, $(h', a, -1, h) \in \delta$. Then, $i = i' - 1$ and $i \geq -D$. Thus, $e + i \geq 0$ and thus $(h', e + i') \xrightarrow{a} (h, e + i)$ in $\mathcal{A}$, completing the proof of the claim. ☐

Since, $c_1 + D \geq D$, we may apply the claim to conclude that $(p_1, c_1 + D) \xrightarrow{w_2} (p_2 = q_1, c_1 + D + c_2 = d_1 + D)$ in $\mathcal{A}$. This completes the proof of the Lemma. ☐

Next we show that $\mathcal{A}[D]$ can simulate any D-band run without reversals.

Lemma 85. *Let $(p, c) \xrightarrow{w} (q, d)$ be an D-band run in $\mathcal{A}$. Then, there is a run $(p, c) \xrightarrow{w} (q, d)$ in $\mathcal{A}[D]$ in which the counter value is never decremented if $c \leq d$ and never incremented if $c \geq d$.*

Proof. The idea is to simply simulate the run as a summary run in $\mathcal{A}[D]$. Let the given run be

$$(p, c) = (p_0, c_0) \xrightarrow{a_1} (p_1, c_1) \xrightarrow{a_2} (p_2, c_2) \ldots \xrightarrow{a_n} (p_n, c_n) = (q, d)$$

Then, it is easy to check that the following is a run in $\mathcal{A}[D]$

$$(p_0, c_0) \xrightarrow{\varepsilon} ((p_0, 0), c_0) \xrightarrow{a_1} ((p_1, c_1 - c_0), c_0) \xrightarrow{a_2} \ldots \xrightarrow{a_n} ((p_n, c_n - c_0), c_0) \xrightarrow{\varepsilon} ((c_n - c_0, p_n), c_0)$$

It is also easy to verify that for any configuration with $((j, p), e)$ with $e + j \geq 0$, $((j, p), e) \xrightarrow{\varepsilon} (p, e + j)$ is a run in $\mathcal{A}[D]$ consisting only of increments if $j > 0$ and consisting only of decrements if $j < 0$. Since $c_n \geq 0$, $(c_n - c_0) + c_0 \geq 0$ and the result follows. ☐

Actually this automaton $\mathcal{A}[D]$ does even better. Concatenation of D-band runs is often not an D-band run but the idea of reversal free simulation extends to certain concatenations. We say that a run $(p_0, c_0) \xrightarrow{w} (p_n, c_n)$ is an increasing (resp. decreasing) *iterated D-band run* if it can be decomposed as

$$(p_0, c_0) \xrightarrow{w_1} (p_1, c_1) \xrightarrow{w_2} \ldots (p_{n-1}, c_{n-1}) \xrightarrow{w_n} (p_n, c_n)$$

where each $(p_i, c_i) \xrightarrow{w_{i+1}} (p_{i+1}, c_{i+1})$ is an D-band run and $c_i \leq c_{i+1}$ (resp. $c_i \geq c_{i+1}$). We say it is an iterated D-band run if it is an increasing or decreasing iterated D-band run.

Lemma 86. *Let $(p, c) \xrightarrow{w} (q, d)$ be an increasing (resp. decreasing) D-band run in $\mathcal{A}$. Then, there is a run $(p, c) \xrightarrow{w} (q, d)$ in $\mathcal{A}[D]$ along which the counter value is never decremented (resp. incremented).*

Proof. Simulate each ρ_i by a run that only increments (resp. decrements) the counter using Lemma 85. □

While, as a consequence of item 1 of Lemma 84, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}[D])$, the converse is not in general true as along a run of $\mathcal{A}[D]$ the real value of the counter, i.e. the current value of the counter plus the offset available in the state, may be negative, leading to runs that are not simulations of runs of $\mathcal{A}$. The trick, as elaborated in item 2 of Lemma 84, that helps us get around this is to relate runs of $\mathcal{A}[D]$ to $\mathcal{A}$ with a shift in counter values. We need a bit more terminology to proceed. We say that a run of is an $D_{\leq}$ run (resp. $D_{\geq}$ run) if the value of the counter is bounded from above (resp. below) by D in every configuration encountered along the run. We say that a run of $\mathcal{A}$ is an $D_{>}$ run if it is of the form $(p, D) \xrightarrow{w} (q, D)$, it has at least 3 configurations and the value of the counter at every configuration other than the first and last is $> D$. Consider any run from a configuration $(p, 0)$ to $(q, 0)$ in $\mathcal{A}$. Once we identify the maximal $D_{>}$ subruns, what is left is a collection of $D_{\leq}$ subruns.

Let $\rho = (p, c) \xrightarrow{w} (q, d)$ be a run of $\mathcal{A}$ with $c, d \leq D$. If ρ is a $D_{\leq}$ run then its *D-decomposition* is ρ. Otherwise, its D-decomposition is given by a sequence of runs $\rho_0, \rho_0', \rho_1, \rho_1' \ldots \rho_{n-1}', \rho_n$ with $\rho = \rho_0.\rho_0'.\rho_1.\rho_1' \ldots . \rho_{n-1}'.\rho_n$, where each ρ_i is a $D_{\leq}$ run and each ρ_i' is a $D_{>}$ run for $0 \leq i \leq n$. Notice, that some of the ρ_i's may be trivial. Since the $D_{>}$ subruns are uniquely identified this definition is unambiguous. We refer to the ρ_i''s (resp. ρ_is) as the $D_{>}$ (resp. $D_{\leq}$) components of ρ.

Observe that the $D_{\leq}$ runs of $\mathcal{A}$ can be easily simulated by an NFA. Thus we may focus on transforming the $D_{>}$ runs, preserving just the Parikh-image, into a suitable form. For $D, M \in \mathbb{N}$, we say that a $D_{>}$ run ρ is a *(D, M)-good run* (think noisy waveform with few reversals) if there are runs $\sigma_1, \sigma_2 \ldots, \sigma_n, \sigma_{n+1}$ and iterated D-band runs $\rho_1, \rho_2, \ldots, \rho_n$ such that $\rho = \sigma_1 \rho_1 \sigma_2 \rho_2 \ldots \sigma_n \rho_n \sigma_{n+1}$ and $|\sigma_1| + \ldots + |\sigma_{n+1}| + 2.n \leq M$. Using Lemma 18 and that it is a $D_{>}$ run we show

Lemma 87. *Let $(p, D) \xrightarrow{w} (q, D)$ be an (D, M)-good run of $\mathcal{A}$. Then, there is a run $(p, 0) \xrightarrow{w} (q, 0)$ in $\mathcal{A}[D]$ with atmost M reversals.*

Proof. Let the given run be ρ. We first shift down ρ to $\rho[-D]$ to obtain a run from $(p, 0)$ to $(q, 0)$, which is possible since ρ is $D_{>}$ run. We then transform each of the iterated D-band runs using Lemma 18 so that there are no reversals in the transformed runs. Thus all reversals occur inside the $\sigma_i[-D]$'s or at the boundary and this gives us the bound required by the lemma. □

So far we have not used the fact that we can ignore the ordering of the letters read along a run (since we are only interested in the Parikh-image of $\mathcal{L}(\mathcal{A})$). We show that for any run ρ of $\mathcal{A}$ we may find another run ρ' of $\mathcal{A}$, that is equivalent up to Parikh-image, such that every $D_{>}$ component in the D-decomposition of ρ' is (D, M)-good, where M and D are polynomially related to K.

We fix $D = K$ in what follows. We take $M = 2K^2 + K$ for reasons that will become clear soon. We focus our attention on some $D_{>}$ component ξ of ρ which is not (D, M)-good. Let $X \subseteq Q$ be the set of states of Q that occur in at least two different configurations along ξ. For each of the states in X we identify the configuration along ξ where it occurs for the very first time and the configuration where it occurs for the last time. There are at most $2|X| (\leq 2K)$ such configurations and these decompose the run ξ into a concatenation of $2|X| + 1 (\leq 2K + 1)$ runs $\xi = \xi_1.\xi_2 \ldots \xi_m$ where $\xi_i, 1 < i < m$ is a segment connecting two such configurations. Now, suppose one of these ξ_i's has length K or more. Then it must contain a sub-run $(p, c) \rightarrow (p, d)$ with at most K moves, for some $p \in X$ (so, this is necessarily a K-band run). If $d - c \geq 0$ (resp. $d - c < 0$), then we *transfer* this subrun from its current position to the first occurrence (resp. last occurrence) of p in the run. This still leaves a valid run ξ' since ξ begins with a K as counter value and $|\xi_i| \leq K$. Moreover ξ and ξ' are equivalent upto Parikh-image.

If this ξ' continues to be a $K_{>}$ run then we again examine if it is (K, M)-good and otherwise, repeat the operation described above. As we proceed, we continue to accumulate a increasing iterated K-band run at the first occurrence of each state and decreasing iterated K-band run at the last occurrence of each state. We also ensure that in each iteration we only pick a segment that does NOT appear in these $2|X|$ iterated K-bands. Thus, these iterations will stop when either the segments outside the iterated K-bands are all of length $< K$ and we cannot find any suitable segment to transfer, or when the resulting run is no longer a $K_{>}$ run. In the first case, we must necessarily have a $(K, 2K^2 + K)$-good run. In the latter case, the resulting run decomposes as usual in $K_{\leq}$ and $K_{>}$ components, and we have that every $K_{>}$ component is strictly shorter than ξ. We formalize the ideas sketched above now.

We begin by proving a Lemma which says that any $K_{>}$ run ρ can be transformed into a Parikh-equivalent run ξ which is either a $K_{>}$ run which is $(K, 2K^2 + K)$-good or has a K-decomposition each of whose $K_{>}$ components are strictly shorter than ρ.

Lemma 88. *Let $\rho = (p, K) \xrightarrow{w} (q, K)$ be a $K_{>}$ run in $\mathcal{A}$. Then, there is a run $\xi = (p, K) \xrightarrow{w'} (q, K)$ in $\mathcal{A}$, with $|\xi| = |\rho|$, $\psi(w) = \psi(w')$ such that one of the following holds:*

1. *ξ is not a $K_{>}$ run. Thus, all $K_{>}$-components in the K-decomposition of ξ are strictly shorter than ξ (and hence ρ).*
2. *ξ is a $K_{>}$ run and $\xi = \sigma_1 \rho_1 \ldots \sigma_n \rho_n$ where $n \leq 2K + 1$, each ρ_i is an iterated K-band run and $|\sigma_i| \leq K$ for each i. Thus, ξ is $(K, 2K^2 + K)$-good.*

Proof. Let $\rho = (p_0, c_0) \xrightarrow{a_1} (p_1, c_1) \ldots \xrightarrow{a_m} (p_m, c_m)$. Let $X \subseteq Q$ be the set of controls states that repeat in the run ρ. We identify the first and last occurrences of each state $q \in X$ along the run ρ, and there are $n = 2.|X| \leq 2K$

such positions. We then decompose the run ρ as follows

$$(p_0, c_0) = (q_0, e_0)\sigma_1(q_1, e_1)\sigma_2(q_2, e_2)\ldots$$
$$\ldots(q_{n-1}, e_{n-1})\sigma_n(q_n, e_n)\sigma_{n+1}(q_{n+1}, e_{n+1}) = (q, d)$$

where configurations $(q_1, e_1), (q_2, e_2) \ldots (q_n, e_n)$ correspond to the first or last occurrence of states from X. We introduce, for reasons that will become clear in the following, an empty iterated K-band run ρ_i following each (q_i, e_i) to get

$$(q_0, e_0)\sigma_1(q_1, e_1)\rho_1(q_1, e_1)\sigma_2(q_2, e_2)\rho_2(q_2, e_2)\ldots$$
$$\ldots(q_{n-1}, e_{n-1})\sigma_n(q_n, e_n)\rho_n(q_n, e_n)\sigma_{n+1}(q_{n+1}, e_{n+1})$$

Let ξ_0 be ρ with the decomposition as written above. We shall now construct a sequence of runs ξ_i, $i \geq 0$, from (p, K) to (q, K), maintaining the length and the Parikh image as an invariant, that is, $\psi(\xi_i) = \psi(\xi_{i+1})$ and $|\xi_i| = |\rho|$. In each step, starting with a $K_>$ run ξ_i, we shall reduce the length of one of the σ_i by some $1 \leq l \leq K$ and increase the length of one iterated K-band runs ρ_j by l to obtain a run ξ_{i+1}, maintaining the invariant. If this resulting run is not a $K_>$ run then it has a K-decomposition in which every $K_>$ component is shorter than ξ_i (and hence ρ), thus satisfying item 1 of the Lemma completing the proof. Otherwise, after sufficient number of iterations of this step, we will be left satisfying item 2 of the Lemma. Let the $K_>$ run ξ_i be given by

$$(q_0, e_0)\sigma_1^i(q_1, e_1^i)\rho_1^i(q_1, f_1^i)\sigma_2^i(q_2, e_2^i)\rho_2^i(q_2, f_2^i)\ldots$$
$$\ldots(q_{n-1}, e_{n-1}^i)\sigma_n^i(q_n, e_n^i)\rho_n^i(q_n, f_n^i)\sigma_{n+1}^i(q_{n+1}, e_{n+1}^i)$$

where each ρ_j^i is an iterated K-band run. If the length of $|\sigma_j^i| \leq K$ for each $j \leq n+1$ then, we have already fulfilled item 2 of the Lemma, completing the proof. Otherwise, there is some j such that $|\sigma_j^i| \geq K$. Therefore, we may decompose σ_j^i as

$$(q_{j-1}, f_{j-1}^i)\chi_1(r, g)\chi_2(r, g')\chi_3(q_j, e_j^i)$$

where $(r, g)\chi_2(r, g')$ is a run of length $\leq K$ and $r \in X$. There are two cases to consider, depending on whether $g' - g \geq 0$ or $g' - g < 0$.

Let (q_B, e_B^i) and (q_E, f_E^i) be the first and last occurrences of r in ξ_i. We will remove the segment of the run given by χ_2 and add it to ρ_B^i if $g' \geq g$ and add it to ρ_E^i otherwise. First of all, since the first and last occurrences of r are distinct, the ρ_B^i will remain a increasing iterated K-band run while ρ_E^i remains a decreasing iterated K-band run. Clearly, such a transformation preserves the Parikh image of the word read along the run. It is easy to check that, since ξ_i is a $K_>$ run and the length of χ_2 is bounded by K, the resulting sequence ξ_{i+1} (after adjusting the counter values) will be a valid run, since the counter stays ≥ 0. However, it may no longer be a $K_>$ run. (This may happen, if $e_B^i < g$ and there is a prefix of χ_2 whose net effect is to reduce the counter by more than $e_B^i - K$.) However, in this case we may set ξ_{i+1} is a run from (p, K) to (q, K), with the same length as ξ_i and thus every $K_>$ component in its K-decomposition is necessarily shorter than ξ_i. Thus, it satisfies item 1 of the Lemma.

If ξ_{i+1} remains a $K_>$ run then we observe that $|\sigma_1^i \ldots \sigma_n^i| > |\sigma_1^{i+1} \ldots \sigma_n^{i+1}|$ and this guarantees the termination of this construction with a ξ satisfying one of the requirements of the Lemma. ☐

Starting with any run, we plan to apply Lemma 88, to the $K_>$ components, preserving Parikh-image, till we reach one in which every $K_>$ component satisfies item 2 of Lemma 88. To establish the correctness of such an argument we need the following Lemma.

Lemma 89. *Let $\rho = (p, 0) \xrightarrow{w} (q, 0)$ be a run. If $\rho = \rho_1(r, K)\rho_2$ then every $K_>$ component in the decomposition of ρ is a $K_>$ component of ρ_1 or ρ_2 and vice versa. In particular, if $\rho = \rho_1(r, K)\rho_2(r', K)\rho_3$ then, $K_>$ components of the K-decomposition of ρ are exactly the $K_>$ components of the runs ρ_1, ρ_2 or ρ_3.*

Proof. By the definition of $K_>$ run and K decompositions. ☐

We can now combine Lemmas 89 and 88 to obtain:

Lemma 90. *Let $\rho = (p, 0) \xrightarrow{w} (q, 0)$ be any run in $\mathcal{A}$. Then, there is a run $\rho' = (p, 0) \xrightarrow{w'} (q, 0)$ of $\mathcal{A}$ with $\psi(w) = \psi(w')$ such that every $K_>$ component ξ in the canonical decomposition of ρ' is $(K, 2K^2 + K)$-good.*

Proof. The proof is by double induction, on the length of the longest $K_>$ component in ρ that is not $(K, 2K^2 + K)$-good and the number of components of this size that violate it. For the basis case, observe that any $K_>$ component whose length is bounded by $2K^2 + K$ is necessarily $(K, 2K^2 + K)$-good.

For the inductive case, we pick a $K_>$ component ξ in ρ of maximum size apply Lemma 88 and replace ξ by ξ' to get ρ'. If ξ' is $(K, 2K^2 + K)$-good we have reduced the number of components of the maximum size that are not $(K, 2K^2 + 2)$-good in ρ'. Otherwise, ξ' satisfies item 2 of Lemma 88 and thus by Lemma 89 the number of $K_>$ components in the decomposition of ρ' of the size of ξ that are not $(K, 2K^2 + K)$-good is one less than that in ρ. This completes the inductive argument. ☐

Remark 91. We note that the above proof can be formulated slightly differently. The reason we work with $D_>$-runs (which is also incorporated in the definition of (D, M)-good runs) is that such runs of $\mathcal{A}$ from, say (p, D) to (q, D), can be simulated faithfully by $\mathcal{A}[D]$ from $(p, 0)$ to $(q, 0)$ while runs of $\mathcal{A}[D]$ from $(p, 0)$ to $(q, 0)$ can be simulated by $\mathcal{A}$ along runs from (p, D) to (q, D). Since, segments of any run of $\mathcal{A}$ where the counter value lies below D can be easily simulated by an NFA, D-decompositions and the above inductive argument come naturally.

A slightly different argument is the following: If we begin with $(2D)_\geq$ run of $\mathcal{A}$ then, we can carry out the above inductive argument without bothering about whether it remains a $(2D)_>$ run at each step, for it is guaranteed to remain a $D_\geq$ run. Then, instead of the automaton $A[D]$ we use a slight variant $B[D]$ which does the following: it simulates $A[D]$ and at every point where it reverses from decrements to increments it *verifies* that the counter is at least D by decrementing the counter D times and then incrementing the counter D times. Then, we can relate $\mathcal{A}$ and $\mathcal{B}[D]$ without a level shift as follows: for any $C \geq 2D$, there is a run $(p, C) \xrightarrow{w} (q, C)$ in $\mathcal{A}$ if and only if there is a run $(p, C) \xrightarrow{w} (q, C)$ in $\mathcal{B}[D]$.

We have preferred the argument where the automaton is simpler to define and it does not track reversals. The two proofs are of similar difficulty.

Let $\mathcal{A}^K$ be the NFA simulating the simple OCA $\mathcal{A}$ when the counter values lie in the range $[0, K]$, by maintaining the counter values in its local state. This automaton is of size $O(K^2)$. Now, suppose for each pair of states $p, q \in Q$ we have an NFA $\mathcal{B}^{pq}$ which is Parikh-equivalent to $\mathcal{L}_{2K^2+K}(\mathcal{A}[K]^{p,q})$, where $\mathcal{A}[K]^{p,q}$ is the automaton $\mathcal{A}[K]$

with p as the only initial state and q as the only accepting state. We combine these automata (there are K^2 of them) with $\mathcal{A}^K$ by taking their disjoint union and adding the following additional (internal) transitions. We add transitions from the states of the from (p, K) of $\mathcal{A}^K$, for $p \in Q$ to the initial state of state of all the $\mathcal{B}^{pq}$, $q \in Q$. Similarly, from the accepting states of $\mathcal{B}^{pq}$ we add internal transitions to the state (q, K) in $\mathcal{A}^K$. Finally we deem $(s, 0)$ to be the only initial state and $(f, 0)$ to be the only final state of the combined automaton. We call this NFA $\mathcal{B}$.

The next lemma confirms that $\mathcal{B}$ is the automaton we are after.

Lemma 92. $\psi(\mathcal{L}(\mathcal{B})) = \psi(\mathcal{L}(\mathcal{A}))$

Proof. Let ρ be an accepting run of $\mathcal{A}$ on a word w. We first apply Lemma 21 to construct a run ρ' on a w', with $\psi(w) = \psi(w')$, in whose K-decomposition, every $K_>$ component is $(K, 2K^2 + K)$-good. Let $\chi = (p, K) \xrightarrow{x} (q, K)$ be such a component. Then, by Lemma 20, there is a run $\chi' : (p, 0) \xrightarrow{x} (q, 0)$ in $\mathcal{A}[K]$ with at most $2K^2 + K$ reversals. Thus, there is a $x' \in \mathcal{L}(\mathcal{B}^{pq})$ with $\psi(x) = \psi(x')$. If $(s, 0) \xrightarrow{x} (p, K)$ is a $K_\leq$ component of ρ' then $(s, 0) \xrightarrow{x} (p, K)$ in $\mathcal{A}^K$. If $(p, K) \xrightarrow{x} (q, K)$ is a $K_\leq$ component of ρ' then $(p, K) \xrightarrow{x} (q, K)$ in $\mathcal{A}^K$ and finally if $(p, K) \xrightarrow{x} (f, 0)$ is a $K_\leq$ component of ρ' then $(p, K) \xrightarrow{x} (f, 0)$ in $\mathcal{A}^K$. Putting these together we get a run from $(s, 0)$ to $(f, 0)$ in $\mathcal{B}$ on a word Parikh-equivalent to w' and hence w.

For the converse, any word in $\mathcal{L}(\mathcal{B})$ is of the form $x.u_1.v_1.u_2.v_2 \ldots u_n v_n.y$ where $(s, 0) \xrightarrow{x} (p_1, K)$ in $\mathcal{A}^K$, $(q_n, K) \xrightarrow{y} (f, 0)$ in $\mathcal{A}^K$, $u_i \in \mathcal{L}(\mathcal{B}^{p_i q_i})$ and $(q_i, K) \xrightarrow{v_i} (p_{i+1}, K)$ in $\mathcal{A}^K$, for each $1 \leq i \leq n$. By construction, there is a run $(s, 0) \xrightarrow{x} (p_1, K)$ in $\mathcal{A}$ and $(q_n, K) \xrightarrow{y} (f, 0)$ in $\mathcal{A}$. Further for each i, there is a run $(q_i, K) \xrightarrow{v_i} (p_{i+1}, K)$ in $\mathcal{A}$ as well. Since $u_i \in \mathcal{L}(\mathcal{B}^{p_i q_i})$, by construction of $\mathcal{B}^{p_i q_i}$, there is a run $(p_i, 0) \xrightarrow{u'_i} (q_i, 0)$ (with a bound on the number of reversals, but that is not important here) in $\mathcal{A}[K]$ with $\psi(u_i) = \psi(u'_i)$. But then, by the second part of Lemma 84, there is a run $(p_i, K) \xrightarrow{u'_i} (q_i, K)$ in $\mathcal{A}$. Thus we can put together these different segments now to obtain an accepting run in $\mathcal{A}$ on the word $x.u'_1.v_1.u'_2.v_2 \ldots u'_n v_n$. Thus, $\psi(\mathcal{L}(\mathcal{B})) \subseteq \psi(\mathcal{L}(\mathcal{A}))$, completing the proof of the Lemma. □

The number of states in the automaton $\mathcal{B}$ is $\sum_{p,q \in Q} |\mathcal{B}^{pq}| + K^2$. What remains to be settled is the size of the automata $\mathcal{B}^{pq}$. That is, computing an upper bound on the size of an NFA which is Parikh-equivalent to the language of words accepted by an OCA (in this case $\mathcal{A}[K]$) along runs with at most R (in this case $K^2 + K$) reversals. This problem is solved in the next subsection and the solution (see Lemma 23) implies that that the size of $\mathcal{B}^{pq}$ is bounded by $O(|\Sigma| K^{O(logK)})$. Thus we have

Theorem 22. *There is an algorithm, which given an OCA with K states and alphabet Σ, constructs a Parikh-equivalent NFA with $O(|\Sigma|.K^{O(\log K)})$ states.*

D.2 Parikh image under reversal bounds

Here we show that, for an OCA $\mathcal{A}$, with K states and whose alphabet is Σ, and any $R \in \mathbb{N}$, an NFA Parikh-equivalent to $\mathcal{L}_R(\mathcal{A})$ can be constructed with size $O(|\Sigma|.K^{O(logK)})$. As a matter of fact, this construction works even for pushdown systems and not just OCAs.

Let $\mathcal{A}$ be a simple OCA. It will be beneficial to think of the counter as a stack with a single letter alphabet, with pushes for increments and pops for decrements. Then, in any run from $(p, 0)$ to $(q, 0)$, we may relate an increment move uniquely with its *corresponding* decrement move, the pop that removes the value inserted by this push.

Now, consider a *one reversal run* ρ of $\mathcal{A}$ from say $(p, 0)$ to $(q, 0)$ involving two phases, a first phase ρ_1 with no decrement moves and a second phase ρ_2 with no increment moves. Such a run can be simulated, up to equivalent Parikh image (i.e. upto reordering of the letters read along the run) by an NFA as follows: simultaneously simulate the first phase (ρ_1) from the source and the second phase, in reverse order (ρ_2^{rev}), from the target. (The simulation of ρ_2^{rev} uses the transitions in the *opposite* direction, moving from the target of the transition to the source of the transition). The simulation matches increment moves of ρ_1 against decrement moves in ρ_2^{rev} (more precisely, matching the ith increment ρ_1 with the ith decrement in ρ_2^{rev}) while carrying out moves that do not alter the counters independently in both directions. The simulation terminates (or potentially terminates) when a common state, signifying the boundary between ρ_1 and ρ_2 is reached from both ends.

The state space of such an NFA will need pairs of states from Q, to maintain the current state reached by the forward and backward simulations. Since, only one letter of the input can be read in each move, we will also need two moves to simulate a matched increment and decrement and will need states of the form $Q \times Q \times \Sigma$ for the intermediate state that lies between the two moves.

Unfortunately, such a naive simulation would not work if the run had more *reversals*. For then the ith increment in the simulation from the left need not necessarily correspond to the ith decrement in the reverse simulation from the right. In this case, the run ρ can be written as follows:

$$\begin{aligned}(p, 0)\rho_1(p_1, c) &\xrightarrow{\tau_1} (p'_1, c + 1)\rho_3(p'_2, c + 1) \\ &\xrightarrow{\tau_2} (p_2, c)\rho_4(q_1, c)\rho_5(q, 0)\end{aligned}$$

where, the increment τ_1 corresponds to the decrement τ_2 and all the increments in ρ_1 are exactly matched by decrements in ρ_5. Notice that the increments in the run ρ_3 are exactly matched by the decrements in that run and similarly for ρ_4. Thus, to simulate such a well-matched run from p to q, after simulating ρ_1 and ρ_5^{rev} simultaneously matching corresponding increments and decrements, and reaching the state p_1 on the left and q_1 on the right, we can choose to now simulate matching runs from p_1 to p_2 and from p_2 to q_1 (for some p_2). Our idea is to choose one of these pairs and simulate it first, storing the other in a stack. We call such pairs *obligations*. The simulation of the chosen obligation may produce further such obligations which are also stored in the stack. The simulation of an obligation succeeds when the state reached from the left and right simulations are identical, and at this point we we may choose to close this simulation and pick up the next obligation from the stack or continue simulating the current pair further. The entire simulation terminates when no obligations are left. Thus, to go from a single reversal case to the general case, we have introduced a stack into which states of the NFA used for the single reversal case are stored. This can be formalized to show that the resulting PDA is Parikh-equivalent to $\mathcal{A}$.

We also add that the order in which the obligations are verified is not important, however, the use of a stack to do this simplifies the arguments. Observe that in this construction each obligation inserted into the stack corresponds to a reversal in the run being simulated, as a matter of fact, it will correspond to a reversal from decrements to increments. Thus it is quite easy to see that the stack height of the simulating run can be bounded by the number of reversals in the original run.

But a little more analysis shows that there is a simulating run where the height of the stack is bounded by $log(R)$ where R is the number of reversals in the original run. Thus, to simulate all runs of $\mathcal{A}$ with at most R reversals, we may bound the stack height of the PDA by $log(R)$.

We show that if the stack height is h then we can choose to simulate only runs with at most $2^{log(R)-h}$ reversals for the obligation on hand. Once we show this, notice that when $h = log(R)$ we only need to simulate runs with 1 reversal which can be done without any further obligations being generated. Thus, the overall height of the stack is bounded by $log(R)$. Now, we explain why the claim made above holds. Clearly it holds initially when $h = 0$. Inductively, whenever we split an obligation, we choose the obligation with fewer reversals to simulate first, pushing the other obligation onto the stack. Notice that this obligation with fewer reversals is guaranteed to contain at most half the number of reversals of the current obligation (which is being split). Thus, whenever the stack height increases by 1, the number of reversals to be explored in the current obligation falls at least by half as required. On the other hand, an obligation (p, q) that lies in the stack at position h from the bottom, was placed there while executing (earlier) an obligation (p', q') that only required 2^{k-h+1} reversals. Since the obligation (p, q) contributes only a part of the obligation (p', q'), its number of reversals is also bounded by 2^{k-h+1}. And when (p, q) is removed from the stack for simulation, the stack height is $h - 1$. Thus, the invariant is maintained.

We now describe the formal construction of the automaton establish its correctness now. We establish the result directly for a pushdown system. A pushdown system is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \bot, \delta, s, F)$ where Γ is the stack alphabet and $\bot$ is a special bottom of stack symbol. The transitions in δ are of the form $(q, a, push(x), q')$ denoting a move where the letter $x \in \Gamma$ is pushed on the stack while reading $a \in \Sigma_\varepsilon$, or $(q, a, pop(x), q')$ denoting a move where the letter $x \in \Gamma$ is popped from the stack while reading $a \in \Sigma_\varepsilon$ or (q, a, i, q') where the stack is ignored while readin $a \in \Sigma_\varepsilon$. A configuration of such a pushdown is a pair (q, γ) with $q \in Q$ and $\gamma = \Gamma^*\bot$. The notion of move $(q, \gamma) \xrightarrow{\tau} (q', \gamma')$ using some $\tau \in \delta$ and $(q, \gamma) \xrightarrow{a} (q', \gamma')$ where $a \in \Sigma_\varepsilon$ are defined as expected and we omit the details here.

Observe first of all that if Γ is a singleton we have exactly a simple OCA. The push moves correspond to increments, pop moves to decrements and there are no *emptiness tests* here as there are no zero tests in simple OCAs, and the correspondence between configurations is obvious. We remark that as far as PDAs go, the lack of an emptiness test is not a real restriction as we can push a special symbol right at the beginning of the run and subsequently simulate an emptiness test by popping and pushing this symbol back on to the stack. Thus, we lose no generality either. Having said this, we use emptiness test in the PDA we construct as it simplifies the presentation (while omitting it from the one given as input w.l.o.g.)

Given a PDA $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \bot, s, F)$ we construct a new PDA $\mathcal{A}_P$ which simulates runs of $\mathcal{A}$, upto Parikh-images, and does so using runs where the stack height is bounded by $log(R)$ where R is the number of reversals in the run of $\mathcal{A}$ being simulated. $\mathcal{A}_P = (\Gamma_P \cup \{s_P, t_P\}, \Sigma, \Gamma_P, \delta_P, s_P, t_P)$ is defined as follows. The set of Γ_P is given by $(Q \times Q) \cup (Q \times Q \times \Sigma)$. States of the form (p, q) are charged with simulating a well matched run from $(p, \bot)$ to $(q, \bot)$. While carrying out a matched push from the left and a pop from the right, as we are only allowed read one letter of Σ in a single move, we are forced to have an intermediary state to allow for the reading of the letters corresponding to both the transitions being simulated. The states of the form (p, q, a), $a \in \Sigma$, are used for this purpose. The transition relation δ_P is described below:

1. $(s_P, \varepsilon, i, (s, t)) \in \delta_P$. Initialize the start and target states.
2. $((p, q), a, i, (p', q)) \in \delta_P$ whenever $(p, a, i, p') \in \delta$. Simulate an internal move from the left.
3. $((p, q), a, i, (p, q')) \in \delta_P$ whenever $(q', a, i, q) \in \delta$. Simulate an internal move from the right.
4. $((p, q), a, i, (p', q', b)) \in \delta_P$ whenever
$$(p, a, push(x), p'), (q', b, pop(x), q) \in \delta_P$$
for some $x \in \Gamma$. Simulate a pair of matched moves, a push from the source and the corresponding pop from the target, first part.
5. $((p, q, b), b, i, (p, q)) \in \delta_P$ whenever $b \in \Sigma$. Second part of the move described in previous item.
6. $((p, q), \varepsilon, push((q', q)), (p, q')) \in \delta_P$ for every state $q' \in Q$. Guess a intermediary state where a pop to push reversal occurs. Simulate first half first and push the second as an obligation on the stack.
7. $((p, q), \varepsilon, push((p, q')), (q', q)) \in \delta_P$ for every state $q' \in Q$. Guess a intermediary state where a pop to push reversal occurs. Simulate second half first and push the first as an obligation on the stack.
8. $((p, p), \varepsilon, pop((p', q')), (p', q')) \in \delta_P$. Current obligation completed, load next one from stack.
9. $((p, p), \varepsilon, \bot?, t_P) \in \delta_P$. All segments completed successfully, so accept.

The following Lemma shows that every run of $\mathcal{A}_P$ simulates some run of $\mathcal{A}$ upto Parikh-image. In what follows, we say that a run ρ is a γ-run for some $\gamma \in \Gamma^*\bot$ if γ is a suffix of the stack contents in every configuration in ρ.

Lemma 93. *Let $\beta \in \Gamma_P^*\bot$. Let $((p, q), \beta) \xrightarrow{w} ((r, r), \beta)$ be a β-run in $\mathcal{A}_P$, for some p, q and r in Q. Then, for every $\gamma \in \Gamma^*\bot$ there is a run $(p, \gamma) \xrightarrow{w'} (q, \gamma)$ in $\mathcal{A}$ such that $\psi(w') = \psi(w)$. Thus, if $w \in \mathcal{L}(\mathcal{A}_P)$ then there is a w' in $\mathcal{L}(\mathcal{A})$ with $\psi(w) = \psi(w')$.*

Proof. For purpose of the proof, we will prove the following claim.

Claim 94. *If there is a run of the form $((p, q), \beta) \xrightarrow{v} ((p', q'), \beta)$ in $\mathcal{A}_P$, then for every $\gamma \in \Gamma^*$, there are runs of the form $(p, \gamma\bot) \xrightarrow{v_1} (p', \alpha\gamma\bot)$ and $(q', \alpha\gamma\bot) \xrightarrow{v_2} (q, \gamma\bot)$, such that $\psi(v) = \psi(v_1.v_2)$.*

Proof. We will now prove this by inducting on stack height reached and on length of the run. Suppose the stack was never used (always remained β), then the proof is easy to see.

Let us assume that stack was indeed used, then the run $((p, q), \beta) \xrightarrow{v} ((p', q'), \beta)$ can be split as

$$((p,q),\beta) \xrightarrow{v_1} ((p_1,q_1),\beta) \rightarrow ((p_2,q_2),(t_1,t_2)\beta) \xrightarrow{v_2} ((r_1,r_1),(t_1,t_2)\beta) \rightarrow ((t_1,t_2),\beta) \xrightarrow{v_3} ((p',q'),\beta)$$

We have two cases to consider, either $q_1 = t_2$ or $p_1 = t_1$. We will consider the case where $q_1 = t_2$, the other case is analogous. In this case, clearly $p_2 = p_1$ and $t_1 = q_2$. Hence the run is of the form

$$((p,q),\beta) \xrightarrow{v_1} ((p_1,q_1),\beta) \rightarrow ((p_1,q_2),(q_2,q_1)\beta) \xrightarrow{v_2} ((r_1,r_1),(q_2,q_1)\beta) \rightarrow ((q_2,q_1),\beta) \xrightarrow{v_3} ((p',q'),\beta)$$

Now consider the sub-run of the form

$$((p,q),\beta) \xrightarrow{v_1} ((p_1,q_1),\beta)$$

clearly such a run is shorter and hence by induction we have a corresponding runs of the form $(p,\gamma\bot) \xrightarrow{v_1'} (p_1,\alpha''\gamma\bot)$ and $(q_1,\alpha''\gamma\bot) \xrightarrow{v_1''} (q,\gamma\bot)$, for some $\alpha'' \in \Gamma^*$ and such that $\psi(v_1) = \psi(v_1'.v_1'')$.

Consider the sub-run of the form

$$((p_1,q_2),(q_2,q_1)\beta) \xrightarrow{v_2} ((r_1,r_1),(q_2,q_1)\beta)$$

clearly stack height of such a run is shorter by 1. Hence by induction, we have a corresponding runs of the form, $(p_1,\alpha\gamma\bot) \xrightarrow{v_2'} (r_1,\alpha'\alpha\gamma\bot)$ and $(r_1,\alpha'\alpha\gamma\bot) \xrightarrow{v_2''} (q_1,\alpha\gamma\bot)$ for some $\alpha' \in \Gamma^*$, such that $\psi(v_2) = \psi(v_2'.v_2'')$.

consider the sub-run of the form

$$((q_2,q_1),\beta) \xrightarrow{v_3} ((p',q'),\beta)$$

clearly such a run is shorter in length, hence by induction, we have corresponding runs $(q_2,\gamma\bot) \xrightarrow{v_3'} (p',\alpha\gamma\bot)$ and $(q',\alpha\gamma\bot) \xrightarrow{v_3''} (q_1,\gamma\bot)$, for some $\alpha \in \Gamma^*$ and such that $\psi(v_3) = \psi(v_3'.v_3'')$.

Now combining these sub-runs, we get the required run.

□

It is easy to see that the proof of Lemma follows directly once this claim is in place.

□

In the other direction, we show that every run of $\mathcal{A}$ is simulated upto Parikh-image by $\mathcal{A}_P$ with a stack height that is logarithmic in the number of reversals. Let

$$(p,\alpha) = (p_0,\alpha_0) \xrightarrow{\tau_1} (p_2,\alpha_1) \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} = (p_n,\alpha_n) = (q,\alpha)$$

be a run in $\mathcal{A}$. A reversal in such a run is a sequence of the from

$$(p_i,\alpha_i) \xrightarrow{a_{i+1},pop(x_i)} (p_{i+1},\alpha_{i+1}) \xrightarrow{\tau_{i+2}\ldots\tau_{j-1}} (p_{j-1},\alpha_{j-1}) \xrightarrow{a_j,push(x_j)} (p_j,\alpha_j)$$

where none of the transitions $\tau_{i+2}\ldots\tau_{j-1}$ are push or pop moves. The next lemma shows how $\mathcal{A}_R$ simulates runs of $\mathcal{A}$ and provides bounds on stack size in terms of the number of reversals of the run in $\mathcal{A}$.

Lemma 95. *Let $(p,\alpha) \xrightarrow{w} (q,\alpha)$ be a α-run of $\mathcal{A}$ with R reversals with $\alpha \in \Gamma^*.\bot$. Then, for any $\gamma \in \Gamma_P^*\bot$, there is a γ-run $((p,q),\gamma) \xrightarrow{w'} ((r,r),\gamma)$ with $\psi(w) = \psi(w')$. Further for any configuration along this run the height of the stack is no more than $|\gamma| + log(R+1)$.*

Proof. The proceeds by a double induction, first on the number of reversals and then on the length of the run.

For the base case, suppose $R = 0$. If the length of the run is 0 then the result follows trivially. Otherwise, let the α-run ρ, $\alpha \in \Gamma^*\bot$ be of the form:

$$(p,\alpha) = (p_0,\alpha_0) \xrightarrow{\tau_1} (p_1,\alpha_1) \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} (p_n,\alpha_n) = (q,\alpha)$$

If τ_1 is an internal move (p_0,a_1,i,p_1) then $((p_0,p_n),a_1,i,(p_1,p_n))$ is a transition δ_P (of type 2). Thus

$$((p_0,p_n),\gamma) \xrightarrow{a_1} ((p_1,p_n),\gamma)$$

is a valid move in $\mathcal{A}_P$. Let $w = a_1w_1$. Then, by induction hypothesis, there is a γ-run

$$((p_1,p_n),\gamma) \xrightarrow{w_1'} ((r,r),\gamma))$$

with $\psi(w_1') = \psi(w_1)$, whose stack height is bounded by $|\gamma|$. Putting these two together we get a γ-run

$$((p_0,p_n),\gamma) \xrightarrow{a_1.w_1'} ((r,r),\gamma)$$

with $\psi(w) = \psi(a_1.w_1')$ whose stack height is bounded by $|\gamma|$ as required.

If τ_n is an internal transition (p_{n-1},a_n,i,p_n) then

$$((p_0,p_n),a_n,i,(p_0,p_{n-1})) \in \delta_P$$

is a transition of of type 3. Thus, $((p_0,p_n),\gamma) \xrightarrow{a_n} ((p_0,p_{n-1}),\gamma)$ is a move in $\mathcal{A}_P$. Further, by the induction hypothesis, there is a word w_2 with $w = w_2.a_n$ and a γ-run $((p_0,p_{n-1}),\gamma) \xrightarrow{w_2'} ((r,r),\gamma)$ with $\psi(w_2) = \psi(w_2')$. Then, since $\psi(a_n.w_2') = \psi(w_2.a_n)$, we can put these two together to get the requisite run. Once again the stack height is bounded by $|\gamma|$.

Since the given run is a α-run, the only other case left to be considered is when τ_1 is a push move and τ_n is a pop move. Thus, let $\tau_1 = (p_0,a_1,push(x_1),p_1)$ and $\tau_n = (p_{n-1},a_n,pop(x_n),p_n)$. We claim that $x_1 = x_n$ and as a matter fact the value x_1 pushed by τ_1 remains in the stack all the way till end of this run and is popped by τ_n. If the x_1 was popped earlier in the run than the last step, then the stack height would have necessarily reached $|\alpha|$ at this pop, and therefore there will necessarily be a subsequent push of x_n. But this contradicts the fact that $R = 0$. Thus, we have the following moves in $\mathcal{A}_P$.

$$((p_0,p_n),\gamma) \xrightarrow{((p_0,p_n),a_1,i,(p_1,p_{n-1},a_n))} ((p_0,p_{n-1},a_n),\gamma) \xrightarrow{((p_1,p_{n-1},a_n),a_n,i,(p_1,p_{n-1})} ((p_1,p_{n-1}),\gamma)$$

Let $w = a_1w_3a_n$. Then applying the induction hypothesis we get a γ-run $((p_1,p_{n-1}),\gamma) \xrightarrow{w_3'} ((r,r),\gamma)$ where the stack height is never more than $|\gamma|$. Combining these two gives us a γ-run $((p_0,p_n),\gamma) \xrightarrow{a_1a_nw_3'} ((r,r),\gamma)$ where the stack height is never more than $|\gamma|$. Observing that $\psi(a_1a_nw_3') = \psi(a_1w_3a_n)$ gives us the desired result.

Now we examine runs with $R \geq 1$. And once again we proceed by induction on the length l of runs with R reversals. For $R \geq 1$ there are no runs of length $l = 0$ and so the basis holds trivially. As usual, let

$$(p,\alpha) = (p_0,\alpha_0) \xrightarrow{\tau_1} (p_1,\alpha_1) \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} = (p_n,\alpha_n) = (q,\alpha)$$

be an α-run with R reversals. If either τ_1 or τ_n is an internal move then the proof can proceed by induction on l exactly along the same lines as above and the details are omitted. Otherwise, since this is a α-run, τ_1 is a push move and τ_n is a pop move. Let $\tau_1 = (p_0, a_1, push(x_1), p_1)$ and $\tau_n = (p_{n-1}, a_n, pop(x_n), p_n)$. Now we have two possibilities.

Case 1: The value x_1 pushed in τ_1 is popped only by τ_n. This is again easy, as we can apply the same argument as in the case $R = 0$ to conclude that,

$$((p_0, p_n), \gamma) \xrightarrow{((p_0,p_n),a_1,i,(p_1,p_{n-1},a_n))} ((p_0, p_{n-1}, a_n), \gamma)$$
$$\xrightarrow{((p_1,p_{n-1},a_n),a_n,i,(p_1,p_{n-1}))} ((p_1, p_{n-1}), \gamma)$$

Again, with $w = a_1 w_3 a_2$, and applying the induction hypothesis to the shorter run $(p_1, \alpha_1) \xrightarrow{w_3} (p_{n-1}, \alpha_{n-1})$ with exactly R reversals, we obtain a γ-run

$$((p_1, p_{n-1}), \gamma) \xrightarrow{w_3'} ((r, r), \gamma)$$

in which the height of the stack is bounded by $|\gamma| + log(R+1)$. Combining these gives us the γ-run with stack height bounded by $|\gamma| + log(R+1)$, $((p_0, p_n), \gamma) \xrightarrow{a_1 a_n w_3'} ((r, r), \gamma)$ as required.

Case 2: The value x_1 pushed in τ_1 is popped by some τ_j with $j < n$. Then we break the run into two α-runs, $\rho_1 = (p_0, \alpha_0) \xrightarrow{a_1 \ldots a_j} (p_j, \alpha_j)$ and $\rho_2 = (p_j, \alpha_j) \xrightarrow{a_{j+1} \cdots a_n} (p_n, \alpha_n)$. Note that $\alpha = \alpha_0 = \alpha_j = \alpha_n$. Let $a_1 \ldots a_j = w_1$ and $a_{j+1} \ldots a_n = w_2$. Let the number of reversals of ρ_1 and ρ_2 be R_1 and R_2 respectively. First of all, we observe that $R_1 + R_2 + 1 = R$. Thus $R_1, R_2 < R$ and further either $R_1 \leq R/2$ or $R_2 \leq R/2$.

Suppose $R_1 \leq R/2$. Then, by the induction hypothesis, there is an $((p_j, p_n)\gamma)$-run

$$\rho_1' = (((p_0, p_j), (p_j, p_n).\gamma) \xrightarrow{w_1'} ((r', r'), (p_j, p_n).\gamma))$$

with $\psi(w_1) = \psi(w_1')$ and whose stack height is bounded by

$$\begin{aligned} |(p_j, p_n).\gamma| + log(R_1 + 1) &= |\gamma| + 1 + log(R_1 + 1) \\ &\leq |\gamma| + 1 + log(R + 1) - 1 \\ &= |\gamma| + log(R + 1) \end{aligned}$$

Similarly, by the induction hypothesis, there is an γ-run $\rho_2' = ((p_j, p_n), \gamma) \xrightarrow{w_2'} ((r, r), \gamma)$ whose number of reversals is bounded by $|\gamma| + log(R_2 + 1) \leq |\gamma| + log(R+1)$ and for which $\psi(w_2') = \psi(w_2)$.

We have everything in place now. We construct the desired run by first using a transition of type 6, following by ρ_1', followed by a transition of type 8, followed by a simulation of ρ_2' to obtain the following:

$$((p_0, p_n), \gamma) \xrightarrow{((p_0,p_n),\varepsilon,push((p_j,p_n)),(p_0,p_j))} ((p_0, p_j), (p_j, p_n).\gamma)$$
$$\xrightarrow{w_1'} ((r', r'), (p_j, p_n)\gamma)$$
$$\xrightarrow{((r',r'),\varepsilon,pop((p_j,p_n)),(p_j,p_n))} ((p_j, p_n), \gamma)$$
$$\xrightarrow{w_2'} ((r, r), \gamma)$$

This runs satisfies all the desired properties. The case where $R_2 \leq R/2$ is handled similarly using moves of type 7 instead of type 6 and using the fact the $\psi(w_2'.w_1') = \psi(w_1'.w_2')$. This completes the proof of the Lemma. ☐

As we did for OCAs we let $\mathcal{L}_R(\mathcal{A})$ refer to the language of words accepted by $\mathcal{A}$ along runs with atmost R reversals. Now, for a given R, we can simulate runs of $\mathcal{A}_P$ where stack height is bounded by $log(R)$, using an NFA by keeping the stack as part of the state. The size of such an NFA is $O(|Q_P||\Gamma_P|^{O(log(R))}) = O(|\Sigma||Q|^{O(log(R))})$. Let $\mathcal{A}_R$ be such an NFA. Then by Lemma 93, we have $\psi(\mathcal{L}(\mathcal{A}_R)) \subseteq \psi(\mathcal{L}(\mathcal{A}))$ and by Lemma 95 we also have $\psi(\mathcal{L}_R(\mathcal{A})) \subseteq \psi(\mathcal{L}(\mathcal{A}_R))$. By keeping track of the reversal count in the state, we may construct an $\mathcal{A}'$ with state space size $O(R.|Q|)$ such that that $\mathcal{L}(\mathcal{A}') = \mathcal{L}_R(\mathcal{A}') = \mathcal{L}_R(\mathcal{A})$. Thus, we have

Lemma 96. *There is a procedure that takes a simple OCA $\mathcal{A}$ with K states and whose alphabet is Σ, and a number $R \in \mathbb{N}$ and returns an NFA Parikh-equivalent to $\mathcal{L}_R(\mathcal{A})$ of size $O(|\Sigma|.(RK)^{O(log(R))})$.*

D.3 Completeness result

In the proofs, we will use the following fact, which is easy to see.

Lemma 97. *Let $\mathcal{A}$ be an NFA of size n over Σ and $\sigma\colon \Sigma \to \mathcal{P}(\Gamma^*)$ be a substitution of size m. Then there is an NFA for $\sigma(\mathcal{L}(\mathcal{A}))$ of size $n^2 \cdot m$.*

Proof of Lemma 25. A *phase* of $\mathcal{A}$ is a walk in which no reversal occurs, i.e. which is contained in δ_+^* or in δ_-^*, where

$$\begin{aligned} \delta_+ &= \{(p, a, s, q) \in \delta \mid s \in \{0, 1\}\}, \\ \delta_- &= \{(p, a, s, q) \in \delta \mid s \in \{0, -1\}\}. \end{aligned}$$

Observe that since $\mathcal{A}$ is r-reversal-bounded, every accepting run decomposes into at most $r + 1$ phases. As a first step, we rearrange phases to achieve a certain normal form. We call two phases u and v *equivalent* if (i) $\psi(u) = \psi(v)$, and (ii) they begin in the same state and end in the same state.

If $u = (p_0, a_1, s_1, p_1)(p_1, a_2, s_2, p_2) \cdots (p_{m-1}, a_m, s_m, p_m)$ is a phase, then we write $\Theta(u)$ for the set of all phases

$$(p_0, a_1, s_1, p_1)v_1(p_1, a_2, s_2, p_2) \cdots v_{m-1}(p_{m-1}, a_m, s_m, p_m) \tag{3}$$

where for each $1 \leq i \leq m$, we have $v_i = w_{i,1} \cdots w_{i,k_i}$ for some simple p_i-cycles $w_{i,1}, \ldots, w_{i,k_i}$.

Claim: For each phase v, there is a phase u with $|u| \leq B := 2n^2 + n$ and a phase $v' \in \Theta(u)$ such that v' is equivalent to v.

Observe that it suffices to show that starting from v, it is possible to successively delete factors that form simple cycles such that (i) the set of visited states is preserved and (ii) the resulting phase u has length $\leq B$: If we collect the deleted simple cycles and insert them like the $w_{i,j}$ from 3 into u, we obtain a phase $v' \in \Theta(u)$, which must be equivalent to v.

We provide an algorithm that performs such a successive deletion in v. During its execution, we can mark positions of the current phase with states, i.e. each position can be marked by at most one state. We maintain the following invariants. Let $M \subseteq Q$ be the set of states for which there is a marked position. Then (i) deleting all marked positions results in a walk (hence a phase), (ii) If $p \in M$, then there is some position marked with p that visits p, (iii) for each state p, at most n positions are marked by p, and (iv) the unmarked positions in v form at most $|M| + 1$ contiguous blocks.

The algorithm works as follows. In the beginning, no position in v is marked. Consider the phase $\bar{v}$ consisting of the unmarked positions in v. If $|\bar{v}| \leq n(n+1)$, the algorithm terminates. Suppose $|\bar{v}| > n(n+1)$. Since the unmarked positions in v form at most $n + 1$ contiguous blocks, there has to be a contiguous block w of unmarked positions with $|w| > n$. Then then w contains a simple p-cycle f as a factor. Note that f is also a factor of v. We distinguish two cases:

- If deleting f from v does not reduce the set of visited states, we delete f.
- If there is a state p visited only in f. Then, $p \notin M$: Otherwise, by invariant (ii), there would be a position that visits p and is marked by p and hence lies outside of f. Therefore, we can mark all positions in f by p.

These actions clearly preserve our invariants.

Each iteration of the algorithm reduces the number of unmarked positions, which guarantees termination. Upon termination, we have $|\bar{v}| \leq n(n+1)$, meaning there are at most $n(n+1)$ unmarked positions in v. Furthermore, by invariant (iii), we have at most n^2 marked positions. Thus, v has length $\leq B = 2n^2 + n$ and has the same set of visited states as the initial phase. This proves our claim.

We are ready to describe the OCA $\mathcal{B}$. It has states $Q' = Q \times [0, B] \times [0, r]$. For each $j \in [1, r]$, let $\delta_j = \delta_+$ is j is even and $\delta_j = \delta_-$ if j is odd. The initial state is $(q_0, 0, 0)$ and (q, B, r) is final, where f is the final state of $\mathcal{A}$. For each $(i, j) \in [0, B-1] \times [0, r]$, and each transition $(p, a, s, q) \in \delta_j$, we add the transitions

$$((p, i, j), a, s, (q, i+1, j)), \tag{4}$$

$$((p, i, j), \varepsilon, 0, (p, i+1, j)). \tag{5}$$

Moreover, for each $p \in Q$, $i \in [0, B]$, and $j \in [0, r-1]$, we include

$$((p, i, j), \varepsilon, 0, (q, 0, j+1)). \tag{6}$$

The input alphabet Σ' of $\mathcal{B}$ consists of the old symbols Σ and the following fresh symbols. For each $p \in Q$ and $z \in [-n, n]$, we include a new symbol $a_{p,z}$. Moreover, for each $p \in Q$ and $k \in [0, n]$, we add a loop transition

$$((p, i, j), a_{p,s\cdot k}, s \cdot k, (p, i, j)), \tag{7}$$

where $s = (-1)^{j+1}$. In other words, we add a loop in (p, i, j) that reads $a_{p,s\cdot k}$ and adds $s \cdot k$ to the counter, where the sign s depends on which phase we are simulating. Let us estimate the size of $\mathcal{B}$. It has $n \cdot B \cdot (r+1)$ states. Moreover, for each $k \in [1, n]$, it has $2 \cdot n \cdot B \cdot (r+1)$ transitions with an absolute counter value of k. This means, $\mathcal{B}$ is of size

$$\begin{aligned} & nB(r+1) + \sum_{k=1}^{n} (k-1) \cdot 2 \cdot nB(r+1) \\ = \ & nB(r+1) \cdot (1 + 2 \cdot \frac{1}{2} n(n-1)) \\ = \ & n(2n^2+n)(r+1) \cdot (1 + n(n-1)) \\ \leq \ & 3n^3(r+1) \cdot 2n^2 = 6n^5(r+1) \end{aligned}$$

Furthermore, $\mathcal{B}$ is an RBA: If we set $(p, i, j) < (q, \ell, m)$ iff (i) $j < m$ or (ii) $j = m$ and $i < \ell$, then this is clearly a strict order on the states of $\mathcal{B}$ and every transition of type (4), (5) or (6) is increasing with respect to this order. Hence, a cycle cannot contain a transition of type (4), (5) or (6), and all other transitions are loops. Thus, $\mathcal{B}$ is an RBA.

The idea is now to substitute each symbol $a_{p,z}$ by the regular language of of p-cycles without reversal that add z to the counter. To this end, we define the NFA $\mathcal{B}_{p,z}$ as follows.

- If $z \geq 0$, then $I_z = [0, z]$ and let $\delta'_z = \delta_+$.
- If $z < 0$, then $I_z = [z, 0]$ and let $\delta'_z = \delta_-$.

$\mathcal{B}_{p,z}$ has states $Q_{p,z} = Q \times I_z$, $(p, 0)$ is its initial state and (p, z) its final state. It has the following transitions: For each transition $(q, a, s, q') \in \delta'_z$, we include $((q, y), a, (q', y+s))$ for each $y \in I_z$ with $y + s \in I_z$. Now indeed, $\mathcal{L}(\mathcal{B}_{p,z})$ is the set of inputs of p-cycles without reversal that add z to the counter. Note that $\mathcal{B}_{p,z}$ has at most $n(n+1)$ states.

Let us now define the substitution σ. For each $a \in \Sigma$, we set $\sigma(a) = \{a\}$. For the new symbols $a_{p,z} \in \Sigma' \setminus \Sigma$, we define $\sigma(a_{p,z}) = \mathcal{L}(\mathcal{B}_{p,z})$. Since each $\mathcal{B}_{p,z}$ has at most $n(n+1)$ states, σ has size at most $n(n+1)$.

It remains to be shown that $\psi(\sigma(\mathcal{L}(\mathcal{B}))) = \psi(\mathcal{L}(\mathcal{A}))$. It is clear from the construction that $\sigma(\mathcal{L}(\mathcal{B})) \subseteq \mathcal{L}(\mathcal{A})$ and in particular $\psi(\sigma(\mathcal{L}(\mathcal{B}))) \subseteq \psi(\mathcal{L}(\mathcal{A}))$. For the other inclusion, we apply our claim. Suppose v is an accepting walk of $\mathcal{A}$. Since $\mathcal{A}$ is r-reversal-bounded, v decomposes into $r+1$ (potentially empty) phases: We have $v = v_0 \cdots v_r$, where each v_j is a phase.

For each v_j, our claim yields phases u_j and $v'_j \in \Theta(u_j)$ such that $|u_j| \leq B$ and v'_j is equivalent to v_j. Now each v'_j gives rise to a walk w_j of $\mathcal{B}$ as follows. First, each u_j induces a run from $(p, 0, j)$ to $(q, |u_j|, j)$, where p and q is the first and last state of u_j, respectively, via transitions (4). Now for each simple p-cycle added to u_j to obtain v'_j, we insert a transition of type (7), whose input can later be replaced with the p-cycle by σ. Now we can connect the walks $w_0, \cdots, w_r$ via the transitions (5) and (6) and thus obtain a walk w of $\mathcal{B}$. Clearly, applying σ to the output of w yields an word that is Parikh-equivalent to the ouput of v. This proves $\psi(\mathcal{L}(\mathcal{A})) \subseteq \psi(\sigma(\mathcal{L}(\mathcal{B})))$. □

Proof of Lemma 26. Assume there is a Dyck sequence $x_1, \ldots, x_n$ for which the statement fails. Furthermore, assume that this is a shortest one. We may assume that $x_i \neq 0$ for all $i \in [1, n]$. Of course we have $n > 2r(2N^2 + N)$: Otherwise, we could choose $I = [1, n]$. We define $s_i = \sum_{j=1}^{i} x_j$ for each $j \in [1, n]$.

Consider

$$s_+ = \sum_{i \in [1,n],\ x_i > 0} x_i, \qquad s_- = \sum_{i \in [1,n],\ x_i < 0} x_i.$$

A contiguous subsequence of $x_1, \ldots, x_n$ is called a *phase* if all its numbers have the same sign. Suppose we had $s_i \leq 2N^2 + N$ for every $i \in [1, n]$. Then every positive phase contains at most $2N^2 + N$ elements. Since we have at most r phases, this means $s_+ \leq r(2N^2 + N)$. However, we have $s_+ + s_- \geq 0$ and thus $|s_-| \leq |s_+| \leq r(2N^2 + N)$. This implies $n \leq 2r(2N^2 + N)$, in contradiction to above.

Hence, we have $s_i > 2N^2 + N$ for some $i \in [1, n]$. Choose $r \in [1, i]$ maximal such that $s_{r+1} \leq N^2 + N$. Then $s_r \geq N^2$. Similarly, choose $t \in [i, n]$ minimal such that $s_{t-1} \leq N^2 + N$. Then $s_t \geq N^2$. Note also that $s_j \geq N^2$ for $j \in [r, t]$. Now we have $\sum_{j=r+2}^{i} x_j \geq N^2$ and $\sum_{j=i+1}^{t-2} x_j \leq -N^2$. Therefore, there is a $u \in [0, N]$ that appears at least N times among $x_{r+2}, \ldots, x_i$ and there is $v \in [-N, 0]$ that appears at least N times among $x_{i+1}, \ldots, x_{t-2}$.

We can remove v-many appearances of the u and u-many appearances of the v. Since $s_j \geq N^2$ for $j \in [r, t]$ and we lower the partial sums by at most N^2, this remains a Dyck sequence. We call it $y_1, \ldots, y_m$. Moreover, it has the same sum as $x_1, \ldots, x_n$ since we removed $v \cdot u$ and added $u \cdot v$. Finally, it is shorter than our original sequence and thus has a removable subset I with at most $2r(2N^2 + N)$ elements. We denote sequence remaining after removing the y_i, $i \in I$, by $z_1, \ldots, z_p$. Note that it has sum 0. Now, we add the removed appearances of u and v back at their old places into $z_1, \ldots, z_p$, we get a Dyck sequence with sum 0 that differs from $x_1, \ldots, x_n$ only in the removed y_i, $i \in I$. Thus, we have found a removable subset with at most $2r(2N^2 + N)$ elements, in contradiction to the assumption. □

Proof of Lemma 27. Suppose $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is an RBA of size n. The idea is to keep the counter effect of non-loop transitions in the state. This is possible since $\mathcal{A}$ is an RBA and thus the accumulated effect of all non-loop transitions is bounded by K. This means, however, that the counter value we simulate might be smaller than the value of $\mathcal{B}$'s counter. This means, if the effect stored in the state is negative and $\mathcal{B}$'s counter is zero, we might simulate quasi runs with a negative counter value. That is why faithfully, we can only smulate runs that start and end at counter value K.

We will use the following bounds:

$$N = n^2, \quad M = 2N^2 + N, \quad K = N + M \cdot n.$$

We construct the automaton $\mathcal{B}$ as follows. It has the state set $Q' = Q \times [-K, K] \times [0, M]$. Its initial state is $(q_0, 0, 0)$ and all states $(q, 0, m)$ with $q \in F$ and $m \in [0, M]$ are final. For each non-loop transition $(p, a, s, q) \in \delta$, we include transitions

$$((p, k, m), a, 0, (q, k + s, m)), \tag{8}$$

for all $k \in [-K, K]$ with $k + s \in [-K, K]$ and $m \in [0, M]$. In contrast to non-loop transitions, loop transitions can be simulated in two ways. For each loop transition $t = (p, a, s, p)$, we include the loop transition:

$$((p, k, m), a, s, (p, k, m)) \tag{9}$$

for each $k \in [-K, K]$ and $m \in [0, M]$, but also the transition

$$((p, k, m), a, 0, (p, k + s, m + 1)) \tag{10}$$

for each $k \in [0, K]$ and $m \in [0, M - 1]$ with $k + s \in [-K, K]$.

First, we show that this OCA is in fact acyclic. By assumption, $\mathcal{A}$ is acyclic, so we can equip Q with a partial order $\leq$ such that every non-loop transition of $\mathcal{A}$ is strictly increasing. We define a partial order $\leq'$ on Q' as follows. For $(p, k, m), (p', k', m') \in Q'$, we have $(p, k, m) \leq' (p', k', m')$ if and only if (i) $p < q$ or (ii) $p = q$ and $m < m'$. Then, clearly, all transitions in $\mathcal{B}$ are increasing with respect to $\leq'$.

Note that $\mathcal{B}$ has $n \cdot (2K + 1) \cdot (M + 1)$ states. Moreover, for each of the transitions of $\mathcal{A}$ that have positive weight (of which there are at most n), it introduces $(2K+1) \cdot M$ edges, each of weight at most n. In total, $\mathcal{B}$ has size at most

$$n \cdot (2K + 1) \cdot (M + 1) + n \cdot (2K + 1) \cdot M \cdot n,$$

which is polynomial in n.

It remains to be shown that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}_{(K)}(\mathcal{A})$. We begin with the inclusion $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Consider an accepting walk in $\mathcal{A}$. It can easily be turned into a walk of $\mathcal{B}$ as follows. Instead of a non-loop transition, we take its counterpart of type (8). Instead of a loop-transition, we take its counterpart of type (9). Consider the final configuration $((q, k, m), x)$ reached in $\mathcal{B}$. The sum of k and x is the counter value at the end of walk in $\mathcal{A}$, so $k + x = 0$. Now it could happen that $x > 0$ and $k < 0$, in which case this is not an accepting walk of $\mathcal{B}$. However, we know that k results from executing non-loop transitions and $\mathcal{A}$ is acyclic, which means there are at most n of them in a walk. Hence, we have $x = |k| \leq n^2 = N$.

Consider the loop transitions executed in our walk and let $x_1, \ldots, x_m$ be the counter values they add to $\mathcal{B}$'s counter. Note that $\sum_{i=1}^{m} x_i = x \in [0, N]$. Now we want to *switch* some of the loops, meaning that instead of taking a transition of type (9), we take the corresponding transition of type (10) (including, of course, the necessary updates to the rightmost component of the state). Observe that the loop-transitions contain at most n reversals (the loop-transitions on each state have the same sign in their counter action, otherwise, the automaton would not be reversal-bounded).

According to Lemma 26, there are $\leq 2n(2N^2 + N) = M$ occurences of loops we can switch such that (i) the resulting walk still leaves $\mathcal{B}$'s counter non-negative at all times and (ii) the new walk leaves $\mathcal{B}$'s counter empty in the end. Since we do this with at most M loop-transitions, the rightmost component of the state has enough capacity.

Moreover, we do not exceed the capacity of the middle component: Before the switching, this component assumed values of at most N, because there are at most n non-loop transitions in a run of $\mathcal{A}$. Then, we add at most M times a number of absolute value $\leq n$. Hence, at any point, we have an absolute value of at most $N + M \cdot n = K$. Thus, we have found an accepting walk in $\mathcal{B}$ that accepts the same word. We have therefore shown $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.

The other inclusion, $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}_{(K)}(\mathcal{A})$, it easy to show: Whenever we can go in one step from configuration $((q, k, m), x)$ to $((q', k', m'), x)$ in $\mathcal{B}$ for $q, q' \in Q$, then we can go from $(q, K + x + k)$ to $(q', K + x' + k')$ in $\mathcal{A}$. Note that then, $K + x + y$ and $K + x' + y'$ are both non-negative. This implies that $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}_{(K)}(\mathcal{A})$. ☐

Proof of Theorem 28. Suppose $\mathcal{A}$ is a loop-counting RBA of size n. By possibly adding a state and an ε-transition, we may assume that in $\mathcal{A}$, every run involves at least one transition that is not a loop. After this transformation, $\mathcal{A}$ has size at most $m = n + 1$.

Let $f \in Q$ be the final state of $\mathcal{A}$. Since $\mathcal{A}$ is acyclic, we may define a partial order on Q as follows. For $p, q \in Q$, we write $p \leq q$ is there is a (possibly empty) walk starting in p and arriving in q. Then indeed, since $\mathcal{A}$ is acyclic, $\leq$ is a partial order. We can therefore sort Q topologically, meaning we can find an injective function $\varphi\colon Q \to [1, m]$ such that $p \leq q$ implies $\varphi(p) \leq \varphi(q)$ and $\varphi(q_0) = 1$ and $\varphi(f) = m$. The function φ will help us map states of $\mathcal{A}$ to states of $\mathcal{H}_{2m}$, but it is not quite enough in its current form: We want to map a state p in $\mathcal{A}$ to a state q in $\mathcal{H}_{2m}$ such that the signs of the counter actions of loops in p and in q coincide. To this end, we have to modify φ slightly.

Consider a state p. Since $\mathcal{A}$ is r-reversal-bounded, either all p-loops are non-incrementing or all p-loops are non-decrementing. Hence, we may define $\tau\colon Q \to \{0, 1\}$ by

$$\tau(p) = \begin{cases} 1 & \text{if all } p\text{-loops are non-decrementing} \\ 0 & \text{if all } p\text{-loops are non-incrementing} \end{cases}$$

Using τ, we can construct our modification $\chi\colon Q \to [1, 2m]$ of φ. For $p \in Q$, let

$$\chi(p) = 2 \cdot \varphi(p) - \tau(p).$$

Note that we may assume that $\tau(q_0) = 1$ (otherwise, we could delete the loops in q_0, they cannot occur in a valid run) and thus $\chi(q_0) = 1$. By the same argument, we have $\tau(f) = 0$ and hence $\chi(f) = 2m$. Moreover, we still have that $p \leq q$ implies $\chi(p) \leq \chi(q)$.

The idea is now to let each symbol $a_{s,k}$ be substituted by all labels of loops in the state $\chi^{-1}(s)$ that change the counter by k. Moreover, we want to substitute $c_{s,t}$ by all labels of transitions from $\chi^{-1}(s)$ to $\chi^{-1}(t)$. However, since the loops in $\mathcal{H}_{2m}$ always modify the counter, those loops on $\chi^{-1}(s)$ (or on $\chi^{-1}(t)$) that do not modify the counter, are generated in the images of $c_{s,t}$.

We turn now to the definition of σ. Let $S \subseteq [1, 2m]$ be the set $\chi(Q)$ of all $\chi(q)$ for $q \in Q$. Then $\chi\colon Q \to S$ is a bijection. We begin by defining subsets $\Gamma_{i,j}$ and $\Omega_{i,j}$ of $\Sigma \cup \{\varepsilon\}$ for $i, j \in [1, 2m]$. Consider $s \in S$ and let $k \in [0, m]$. Note that all counter actions on transitions in $\mathcal{A}$ have an absolute value of $\leq m$. By $\Gamma_{s,k}$ we denote the set of all $a \in \Sigma$ such that there is a loop $(\chi^{-1}(s), a, u, \chi^{-1}(s))$ in $\mathcal{A}$ with $|u| = k$. For all other indices i, j, $\Gamma_{i,j}$ is empty. Furthermore, for $s, t \in S$ with $s < t$, let

$$\Omega_{s,t} = \{a \in \Sigma \mid (\chi^{-1}(s), a, 0, \chi^{-1}(t)) \in \delta\}.$$

Note that if $s \neq t$, all transitions from $\chi^{-1}(s)$ to $\chi^{-1}(t)$ leave the counter unchanged ($\mathcal{A}$ is loop-counting). Again, for all other choices of $i, j \in [1, 2m]$, $\Omega_{i,j}$ is empty.

We define σ as follows. Let

$$\Sigma_{2m} = \{a_{i,j} \mid i, j \in [1, 2m]\} \cup \{c_{i,j} \mid i, j \in [1, 2m],\ i < j\}$$

be the input alphabet of $\mathcal{H}_{2m}$. For $s, k \in [1, 2m]$, let

$$\sigma(a_{i,j}) = \Gamma_{i,j} \tag{11}$$

and for $s, t \in [1, 2m]$, $s < t$, we set

$$\sigma(c_{s,t}) = (\Gamma_{s,0})^* \Omega_{s,t} (\Gamma_{t,0})^*. \tag{12}$$

Now it is easy to verify that $\psi(\sigma(\mathcal{L}(\mathcal{H}_{2n+2}))) = \psi(\mathcal{L}(\mathcal{A}))$ (recall that $m = n + 1$). Note that loops in $\mathcal{A}$ that do not modify the counter are contributed by the images of the $c_{s,t}$. This will generate all inputs because we assumed that in $\mathcal{A}$, every run involves at least one non-loop transition. Observe that since the regular languages (11) and (12) each require at most 2 states, σ has size at most 2. □

Proof of Theorem 24. Suppose for each n, there is a Parikh-equivalent NFA for $\mathcal{H}_n$ of size at most $h(n)$. Our proof strategy is the following. The preceding lemmas each allow us to restrict the class of input automata further. It is therefore convenient to define the following. Let $\mathcal{C}$ be a class of one-counter automata. We say that $\mathcal{C}$ is *polynomial modulo h* if there are polynomials p and q such that for each OCA $\mathcal{A}$ in $\mathcal{C}$, there is a Parikh-equivalent NFA $\mathcal{B}$ of size at most $q(h(p(n)))$. Of course, we want to show that the class of all OCA is polynomial modulo h.

First, in section 5.1, we have seen that there is a polynomial p_1 such that the following holds: if the class $\mathcal{C}_{p_1}$ of all automata $\mathcal{A}$ that are $p_1(|\mathcal{A}|)$-reversal-bounded, is polynomial modulo h, then so is the class of all OCA. Therefore, it remains to be shown that $\mathcal{C}_{p_1}$ is polynomial modulo h.

Next, we apply Lemma 25. Together with Lemma 97, it yields that if the class of RBAsis polynomial modulo h, then so is the class $\mathcal{C}_{p_1}$. Hence, it remains to be shown that the class of RBAs is polynomial modulo h.

Furthermore, Lemma 1 and Lemma 27 together imply that if the class of loop-counting RBAs is polynomial modulo h, then so is the class of all RBAs. Hence, we restrict ourselves to the class of loop-counting RBAs.

Finally, Lemma 28 tells us that the class of loop-counting RBAs is polynomial modulo h. □