Two-Way Parikh Automata

Emmanuel Filiot
Université libre de Bruxelles, Belgium

Shibashis Guha
Université libre de Bruxelles, Belgium

Nicolas Mazzocchi
Université libre de Bruxelles, Belgium

Abstract

Parikh automata extend automata with counters whose values can only be tested at the end of the computation, with respect to membership into a semi-linear set. Parikh automata have found several applications, for instance in transducer theory, as they enjoy a decidable emptiness problem.

In this paper, we study two-way Parikh automata. We show that emptiness becomes undecidable in the non-deterministic case. However, it is PSPACE-C when the number of visits to any input position is bounded and the semi-linear set is given as an existential Presburger formula. We also give tight complexity bounds for the inclusion, equivalence and universality problems. Finally, we characterise precisely the complexity of those problems when the semi-linear constraint is given by an arbitrary Presburger formula.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Parikh automata, two-way automata, Presburger arithmetic

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.40

Related Version https://arxiv.org/abs/1907.09362

Funding *Emmanuel Filiot*: Research associate of F.R.S.-FNRS. He was supported by the ARC Project Transform Fédération Wallonie-Bruxelles, the FNRS CDR project J013116F and "Synapse" FNRS MIS project J013116F.
Shibashis Guha: Supported by the ARC project "Non-Zero Sum Game Graphs: Applications to Reactive Synthesis and Beyond" (Fédération Wallonie-Bruxelles).
Nicolas Mazzocchi: PhD student funded by a FRIA fellowship from the F.R.S.-FNRS.

1 Introduction

Parikh automata, introduced in [18], extend finite automata with counters in $\mathbb{Z}$ which can be incremented and decremented, but the counters can only be tested at the end of the computation, for membership in a semi-linear set (represented for instance as an existential Presburger formula). More precisely, transitions are of the form $(q, \sigma, \vec{v}, q')$ where q, q' are states, σ is an input symbol and $\vec{v} \in \mathbb{Z}^d$ is a vector of dimension d. A word w is accepted if there exists a run ρ on w reaching an accepting state and whose final vector (the component-wise sum of all vectors along ρ) belongs to a given semi-linear set. Parikh automata strictly extend the expressive power of finite automata. For example, the context-free language of words of the form $a^n b^n$ is definable by a deterministic Parikh automaton which checks membership in a^*b^*, counts the number of occurrences of a and b, and at the end tests for equality of the counters, i.e. membership in the linear set $\{(n, n) \mid n \in \mathbb{N}\}$. They still enjoy decidable, NP-C, non-emptiness problem [9].

Parikh automata (PA) have found applications for instance in *transducer* theory, in particular to the equivalence problem of functional transducers on words, and to check structural properties of transducers [10], as well as in answering queries in graph databases [9].

39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019).
Editors: Arkadev Chattopadhyay and Paul Gastin; Article No. 40; pp. 40:1–40:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Extensions of Parikh automata with a pushdown stack have been considered in [17] with positive decidability results with respect to emptiness. Two-way Parikh automata with a visibly pushdown stack have been considered in [6] with applications to tree transducers.

In this paper, our objective is to study *two-way Parikh automata* (2PA), the extension of PA with a two-way input head, where the semi-linear set is given by an existential Presburger formula. For 2PA as well as subclasses such as deterministic 2PA (2DPA), we aim at characterizing the precise complexity of their decision problems (membership, emptiness, inclusion, equivalence), and analysing their expressiveness and closure properties.

Contributions. Since semi-linear sets are closed under all Boolean operations, it is easily seen that deterministic Parikh automata (DPA) are closed under all Boolean operations. More interestingly, it is also known that, while they strictly extend the expressive power of DPA, *unambiguous* PA (UPA) are (non-trivially) closed under complement (as well as union and intersection) [2]. We give here a simple explanation to these good closure properties: UPA *effectively* correspond to 2DPA. Closure of 2DPA under Boolean operations indeed holds straightforwardly due to determinism. The conversion of UPA to 2DPA is however non-trivial, but is obtained by the very same result on word transducers: it is known that unambiguous finite transducers are equivalent to two-way deterministic finite transducers [21], based on a construction by Aho, Hopcroft and Ullman [1], recently improved by one exponential in [7]. Parikh automata can be seen as transducers producing sequences of vectors (the vectors occurring on their transitions), hence yielding the result. The conversion of 2DPA to UPA is a standard construction based on *crossing sections*, which however needs to be carefully analysed for complexity purposes.

The effective equivalence between 2DPA and UPA indeed entails decidability of the non-emptiness problem for 2DPA. However, given that non-emptiness of PA is known to be NP-C [9], and the conversion of 2DPA to UPA is exponential, this leads to NExpTime complexity. By a careful analysis of this conversion and small witnesses properties of Presburger formulas, we show that emptiness of 2DPA, and even *bounded-visit* 2PA, is actually PSpace-C. Bounded-visit 2PA are non-deterministic 2PA such that for some natural number k, each position of an input word w is visited at most k times by any accepting computation on w. In particular, 2DPA are always n-visit for n the number of states. If the number k of visits is a fixed constant, non-emptiness is then NP-C, which is consistent with the complexity result of [9] for (one-way) PA (by taking $k = 1$). We show that dropping the bounded-visit restriction however leads to undecidability.

Thanks to the closure properties of 2DPA, we show that the inclusion, universality and equivalence problems are all coNExpTime-C. Those problems are known to be undecidable for PA [18]. The membership problem of 2PA turns out to be NP-C, just as for (one-way) PA. The coNExpTime lower bound holds for one-way deterministic Parikh automata, a result which is also new, to the best of our knowledge.

Finally, we study the extension of two-way Parikh automata with a semi-linear set defined by a Σ_i-Presburger formula, i.e. a formula with a *fixed* number i of unbounded blocks of quantifiers where the consecutive blocks alternate $i-1$ times between existential and universal blocks, and the first block is existential. We characterise tightly the complexity of the non-emptiness problem for bounded-visit Σ_i-2PA, as well as the universality, inclusion and equivalence problems for Σ_i-2DPA, in the weak exponential hierarchy [13]. For $i > 1$, we find that the complexity of these problems is dominated by the complexity of checking satisfiability or validity of Σ_i-Presburger formulas. This is unlike the case $i = 1$: the non-emptiness problem for bounded-visit 2PA is PSpace-C while satisfiability of Σ_1-formulas is NP-C.

Related work. Parikh automata are known to be equivalent to reversal-bounded multicounter machines (RBCM) [16] in the sense that they describe the same class of languages [2]. Two-way RBCM (2RBCM), even deterministic, are known to have undecidable emptiness problem [16]. Using diophantine equations as in [16], we show that emptiness of 2PA is undecidable. However our decidability result for 2DPA contrasts with the undecidabilty of deterministic 2RBCM emptiness. The difference is that 2RBCM can test their counters at any moment during a computation, and not only at the end. Based on the fact that the number of reversals is bounded, deferring the tests at the end of the computation is always possible [16] but non-determinism is needed. Unlike 2DPA, deterministic 2RBCM are not necessarily bounded-visit. A 2DPA can be seen as a deterministic 2RBCM whose tests on counters are only done at the end of a computation.

Two-way Parikh automata on *nested words* have been studied in [6] where it is shown that under the *single-use* restriction (a generalisation of the bounded-visit restriction to nested words), they have NExpTime-C non-emptiness problem. Bounded-visit 2PA are a particular case of those Parikh automata operating on (non-nested) words. Applying the result of [6] to 2PA would yield a non-optimal NExpTime complexity for the non-emptiness problem, as it first goes through an explicit but exponential transformation into a one-way machine with known NP-C non-emptiness problem. Here instead, we rely on a small witness property, whose proof uses a transformation into one-way Parikh automaton, and then we apply a PSpace algorithm performing on-the-fly the one-way transformation up to some bounded length.

Finally, the emptiness problem for the intersection of n PA was shown to be PSpace-C in [9]. Our PSpace-C result on 2PA emptiness generalises this result, as the intersection of n PA can be simulated trivially by a (sweeping) n-bounded 2PA. The main lines of our proof are similar to those in [9], but in addition, it needs a one-way transformation on top of the proof in [9], and a careful analysis of its complexity.

2 Two-way Parikh automata

Two-way Parikh automata are two-way automata extended with weight vectors and a semi-linear acceptance condition. In this section, we first define two-way automata, semi-linear sets and then two-way Parikh automata.

Two-way Automata. A *two-way finite automaton* (2FA for short) A over an alphabet Σ is a tuple $(Q, Q^{\mathsf{L}}, Q^{\mathsf{R}}, Q_I, Q_H, Q_F, \Delta)$ whose components are defined as follows. We let $\vdash$ and $\dashv$ be two delimiters not in Σ, intended to represent the beginning and the end of the word respectively. The set Q is a non-empty finite set of states partitioned into the set of right-reading states Q^{R} and the set of left-reading states Q^{L}. Then, $Q_I \subseteq Q^{\mathsf{R}}$ is the set of initial states, $Q_H \subseteq Q$ is the set of halting states, and $Q_F \subseteq Q_H$ is the set of accepting states. The states belonging to $Q_H \setminus Q_F$ are said to be *rejecting*. Finally, $\Delta \subseteq Q \times (\Sigma \cup \{\vdash, \dashv\}) \times Q$ is the set of transitions. Intuitively, the reading head of A is always placed in between input positions, a transition from $q \in Q^{\mathsf{R}}$ (resp. $q \in Q^{\mathsf{L}}$) reads the input letter on the right (resp. left) of the head and moves the head one step to the right (resp. left). Also, we have the following restrictions on the behaviour of the head to keep it in between the boundaries $\vdash$ and $\dashv$ and to ensure the following properties on the initial and the halting states:

1. no outgoing transition from a halting state:
   $(Q_H \times (\Sigma \cup \{\vdash, \dashv\}) \times Q) \cap \Delta = \varnothing$

2. the head cannot move left (resp. right) when it is to the left of $\vdash$ (resp. right of $\dashv$): $(Q^{\mathsf{L}} \times \{\vdash\} \times Q^{\mathsf{L}}) \cap \Delta = \varnothing$ (resp. $(Q^{\mathsf{R}} \times \{\dashv\} \times (Q^{\mathsf{R}} \setminus Q_F)) \cap \Delta = \varnothing$)
3. all transitions leading to a halting state q_H read the delimiter $\dashv$: $((q, a, q_H) \in \Delta \wedge q_H \in Q_H) \implies (q \in Q^{\mathsf{R}} \wedge a = \dashv)$

A configuration $(u^{\mathsf{L}}, p, u^{\mathsf{R}})$ of A on a word $u \in \Sigma^*$ consists of a state p and two words $u^{\mathsf{L}}, u^{\mathsf{R}} \in (\Sigma \cup \{\vdash, \dashv\})^*$ such that $u^{\mathsf{L}}u^{\mathsf{R}} = \vdash u \dashv$. A *run* ρ on a word $u \in \Sigma^*$ is a sequence $\rho = (u_0^{\mathsf{L}}, q_0, u_0^{\mathsf{R}})a_1(u_1^{\mathsf{L}}, q_1, u_1^{\mathsf{R}}) \dots a_n(u_n^{\mathsf{L}}, q_n, u_n^{\mathsf{R}})$ alternating between configurations on u and letters in $\Sigma \cup \{\vdash, \dashv\}$ such that for all $1 \leq i \leq n$, we have $(q_{i-1}, a_i, q_i) \in \Delta$, and for all $s \in \{\mathsf{L}, \mathsf{R}\}$, if $q_{i-1} \in Q^s$ then $|u_i^s| = |u_{i-1}^s| - 1$. The length of the run ρ, denoted $|\rho|$ is the number of letters appearing in ρ. Here $|\rho| = n$. The run ρ is *halting* if $q_n \in Q_H$ (and hence $u_n^{\mathsf{R}} = \varepsilon$ by condition 3), *initial* if $u_0^{\mathsf{L}} = \varepsilon$ and $q_0 \in Q_I$, *accepting* if it is both initial and halting, and $q_n \in Q_F$; otherwise the run is *rejecting*. A word u is accepted by A if there exists an accepting run of A on u, and the language $L(A)$ of A is defined as the set of words it accepts.

An automaton A is said to be *one-way* (FA) if Q^{L} is empty. A run ρ is said to be *k-visit* if every input position is visited at most k times in the run ρ, i.e. for $\rho = (u_0^{\mathsf{L}}, q_0, u_0^{\mathsf{R}}) \dots (u_n^{\mathsf{L}}, q_n, u_n^{\mathsf{R}})$, we have $\max\{|P| \mid P \subseteq \{0, \dots, n\} \wedge \forall i, j \in P,\ u_i^{\mathsf{L}} = u_j^{\mathsf{L}}\} \leq k$. The automaton A is said to be k-visit if all its accepting runs are k-visit, *fixed-visit* if it is k-visit for some fixed k and *bounded-visit* if it is k-visit for some unfixed k. Also, A is said to be *deterministic* if for all $p \in Q$ and all $a \in \Sigma \cup \{\vdash, \dashv\}$ there exists at most one $q \in Q$ such that $(p, a, q) \in \Delta$. Finally, it is *unambiguous* (denoted by the class 2UFA or UFA depending on whether it is two-way or one-way) if for every input word there exists at most one accepting run. The following proposition is trivial but useful:

▶ **Proposition 2.1.** *Any bounded-visit 2FA with n states is k-visit for some $k \leq n$.*

Semi-linear Sets. Let $d \in \mathbb{N}_{\neq 0}$. A set $L \subseteq \mathbb{Z}^d$ of dimension d is *linear* if there exist $\vec{v}_0, \dots, \vec{v}_k \in \mathbb{Z}^d$ such that $L = \{\vec{v}_0 + \sum_{i=1}^k x_i\vec{v}_i \mid x_1, \dots, x_n \in \mathbb{N}\}$. The vectors $(\vec{v}_i)_{1 \leq i \leq k}$ are the *periods* and $\vec{v}_0$ is called the *base*, forming what we call a *period-base representation* of L, whose size is $d \cdot (k+1) \cdot \log_2(\mu + 1)$ where μ is the maximal absolute integer appearing on the vectors. A set is *semi-linear* if it is a finite union of linear sets. A period-base representation of a semi-linear set is given by a period-base representation for each of the linear sets it is composed of, and its size is the sum of the sizes of all those representations.

Alternatively, a semi-linear set of dimension d can be represented as the set of models of a Presburger formula with d free variables. A Presburger formula is a first-order formula built over terms t on the signature $\{0, 1, +, \times_2\} \cup X$, where X is a countable set of variables and $\times_2$ denotes the doubling (unary) function[1]. In particular, Presburger formulas obey the following syntax:

$$\Phi \stackrel{\text{def}}{=} t \leq t \mid \exists x\ \Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi$$

The class of formulas of the form $\exists\vec{x}_1, \forall\vec{x}_2 \dots, \Omega_i\vec{x}_i\ [\varphi]$ where φ is quantifier free and $\Omega \in \{\forall, \exists\}$ is denoted by $\boldsymbol{\Sigma}_i$. In particular, $\boldsymbol{\Sigma}_1$ is the set of existential Presburger formulas. The size $|\Psi|$ of a formula is its number of symbols. We denote by $\vec{v} \models \varphi$ the fact that a vector $\vec{v}$ of dimension d satisfies a formula φ with d free variables, and say that φ is satisfiable if there exists such a $\vec{v}$. The formula φ is said to be valid if it is satisfied by any $\vec{v}$. It is well-known [12] that a set $S \subseteq \mathbb{Z}^d$ is semi-linear iff there exists an existential Presburger formula ψ with d free variables such that $S = \{\vec{v} \mid \vec{v} \models \psi\}$.

[1] The function $\times_2$ is syntactic sugar allowing us to have simpler binary encoding of values.

Let $\Sigma = \{a_1, \ldots, a_n\}$ be an alphabet (assumed to be ordered), and $u \in \Sigma^*$, the Parikh image of u is defined as the vector $\mathfrak{P}(u) = (|u|_{a_1}, \ldots, |u|_{a_n})$ where $|u|_a$ denotes the number of times a occurs in u. The Parikh image of language $L \subseteq \Sigma^*$ is $\mathfrak{P}(L) = \{\mathfrak{P}(u) | u \in L\}$. Parikh's theorem states that the Parikh image of any context-free language is semi-linear.

Two-way Parikh automata. A two-way Parikh automaton (2PA) of dimension $d \in \mathbb{N}$ over Σ is a tuple $P = (A, \lambda, \psi)$ where $A = (Q, Q^{\mathsf{L}}, Q^{\mathsf{R}}, Q_I, Q_H, Q_F, \Delta)$ is a 2FA over Σ, $\lambda\colon \Delta \to \mathbb{Z}^d$ maps transitions to vectors, and ψ is an *existential* Presburger formula with d free variables, and is called the *acceptance constraint*. The *value* $V(\rho)$ of a run ρ of A is the sum of the vectors occurring on its transitions, with $V(\rho) = 0_{\mathbb{Z}^d}$ if $|\rho| = 0$. A word is accepted by P if it is accepted by some accepting run ρ of A and $V(\rho) \models \psi$. The language $L(P)$ of P is the set of words it accepts. The automaton P is said to be *one-way*, *two-way*, *k-visit*, *unambiguous* and *deterministic* if its underlying automaton A is so. We define the representation size[2] of P as $|P| = |Q| + |\psi| + |\mathrm{range}(\lambda)|\big(d \log_2(\mu + 1) + |Q|^2\big)$ where $\mathrm{range}(\lambda) = \{\lambda(t) \mid t \in \Delta\}$ and μ is the maximal absolute entries appearing in weight vectors of P. Finally two 2PA are *equivalent* if they accept the same language.

Examples. Let $\Sigma = \{a, b, c, \#\}$ and for all $n \in \mathbb{N}$, let $L_n = \{a^k\#u \mid u \in \{b, c\}^* \wedge k = |\{i \mid 1 \le i \le |u| - n \wedge u[i] \ne u[i+n]\}|\}$, i.e. k is the number of positions i in u such that the ith letter $u[i]$ mismatches with $u[i+n]$. For all n, L_n is accepted by the 2DPA of Fig. 1 which has $O(n)$ states, tagged with R or L to indicate whether they are right- or left-reading respectively. On a word w, the automaton starts by reading a^k and increments its counter to store the value k (state q^a). Then, for the first $|u| - n$ positions i of u, the automaton checks whether $u[i] \ne u[i+n]$ in which case the counter is decremented. To do so, it stores $\sigma = u[i]$ in its state, moves $n+1$ times to the right (states $q_0, q_1^\sigma, \ldots, q_n^\sigma$), checks whether $u[i+n] \ne u[i]$ (transitions q_n^σ to p_1) and decrements the counter accordingly. Then, it moves n times to the left (states p_1 to p_n). Whenever it reads $\dashv$ from states q_j^σ, p_j or q_0, it moves to state q_F and accepts if the counter is zero.

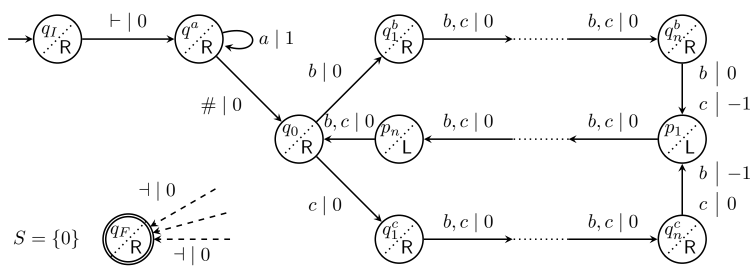


Figure 1 A 2DPA recognising $L_n = \{a^k\#u \mid u \in \{b, c\}^* \wedge k = |\{i \mid 1 \le i \le |u| - n \wedge u[i] \ne u[i+n]\}|\}$.

Our second example shows how to encode multiplication. The language $\{a^n\#a^m\#a^{n\times m} \mid n, m \in \mathbb{N}\}$ is indeed definable by the 2PA of Figure 2 which has dimension 2. When reading a word of the form $a^n\#a^m\#a^\ell$, every accepting run makes k passes over a^n where k is chosen non-deterministically by the choice made on state q_1 on reading $\#$. Along those k passes, the automaton increments the first dimension whenever a is read in a right-to-left pass. It

[2] Note that weight vectors are not memorized on transitions but into a table and transitions only carry a key of this table to refer the corresponding weight vectors.

also counts the number of passes in the second dimension. Thus, when entering state q_2, the sum of the vectors so far is (nk, k). Then, on a^m, it decrements the second dimension and on a^ℓ, it decrements the first dimension, and eventually checks that both the counters are equal to zero, which implies that $k = m$ and $\ell = nk = nm$. Note that this automaton is not bounded-visit as its number of visits to any position of a^n is arbitrary.

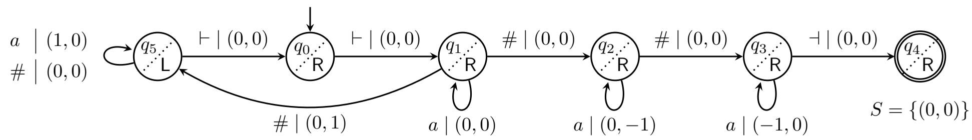


Figure 2 A 2PA recognising $\{a^n \# a^m \# a^{n\times m} \mid n, m \in \mathbb{N}\}$.

3 Relating two-way and one-way Parikh automata

In this section, we provide an algorithm which converts a bounded-visit 2PA into a PA defining the same language, through a *crossing section* construction. This technique is folkloric in the literature (see Section 2.6 of [15]) and has been introduced to convert a 2FA into an equivalent FA. Intuitively, the one-way automaton is constructed such that on each position i of the input word, it guesses a tuple of transitions (called crossing section), triggered by the original two-way automaton at the same position i and additionally checks a local validity between consecutive tuples (called *matching* property). A one-way automaton takes crossing sections as set of states. Furthermore, the matching property is defined to ensure that the sequence of crossing sections which successively satisfy it, correspond to the sequence of crossing sections of an accepting two-way run. Thanks to the commutativity of $+$, the order in which weights are combined by the two-way automaton does not matter and therefore, transitions of the one-way automaton are labelled by summing the weights of transitions of the crossing section. Formally, we define a crossing section as follows:

▶ **Definition 3.1** (crossing section). *Let $k \in \mathbb{N}_{\neq 0}$. Consider a k-visit 2PA $P = (A, \lambda, \psi)$ over Σ and $a \in \Sigma \cup \{\vdash, \dashv\}$. An a-crossing section is a sequence $c = (p_1, a, q_1) \dots (p_\ell, a, q_\ell) \in \Delta^+$ such that $1 \le \ell \le k$, $p_1, q_\ell \in Q^{\mathsf{R}}$ and for all $m \in \{\mathsf{L}, \mathsf{R}\}$, $p_i \in Q^m \implies p_{i+1} \notin Q^m$. We define the value of c as $V(c) = \sum_{i=1}^{\ell} \lambda(p_i, a, q_i)$, and its length $|c| = \ell$. The L-anchorage of c is defined by $p_1 f(q_2, p_3) \dots f(q_{\ell-1}, p_\ell)$ where $f(q_i, p_{i+1}) = \varepsilon$ if $q_i = p_{i+1}$ and $q_i \in Q^{\mathsf{R}}$, otherwise $f(q_i, p_{i+1}) = q_i p_{i+1}$. The R-anchorage of c is defined by $f(q_1 p_2) \dots f(q_{\ell-2} p_{\ell-1}) q_\ell$ where $f(q_i, p_{i+1}) = \varepsilon$ if $q_i = p_{i+1}$ and $q_i \in Q^{\mathsf{L}}$, otherwise $f(q_i, p_{i+1})$ is the identity. Furthermore, c is said to be* initial *if its L-anchorage is $p_1 \in Q_I$. Dually, c is said to be* accepting *if its R-anchorage is $q_\ell \in Q_F$.*

Given a run ρ of a 2PA over u and a position $1 \le i \le |u|$, the *crossing section of ρ at position i* is defined as the sequence of all transitions triggered by ρ when reading the ith letter, taken in the order of appearance in ρ. We also define the *crossing section sequence* $\mathcal{C}(\rho)$ as the sequence of crossing sections of ρ from position 1 to $|u|$. Note that the first crossing section is initial and the last crossing section of ρ is accepting if ρ is accepting.

▶ **Example 3.2.** Figure 3, shows a run over the word $\vdash ab \dashv$. Consider the a-crossing section $c = (p_1, a, q_1)(p_2, a, q_2)(p_2, a, q_3)(p_4, a, q_4)(p_5, a, q_5)$ with $q_1 = p_2, q_2 = p_3$ and $q_4 = p_5$. In particular the run makes on immediate reversal at those states, and exits the a-crossing section from q_3 to q_5. The L-anchorage of c is $p_1 f(q_2, p_3) f(q_4, p_5) = p_1$, the R-anchorage of c is $f(q_1, p_2) f(q_3, p_4) q_5 = q_3 p_4 q_5$ and $V(c) = \vec{v}_2 + \vec{v}_3 + \vec{v}_4 + \vec{v}_{11} + \vec{v}_{12}$. Note that the states of the crossing section do not appear in the anchorage when the run changes its reading direction.

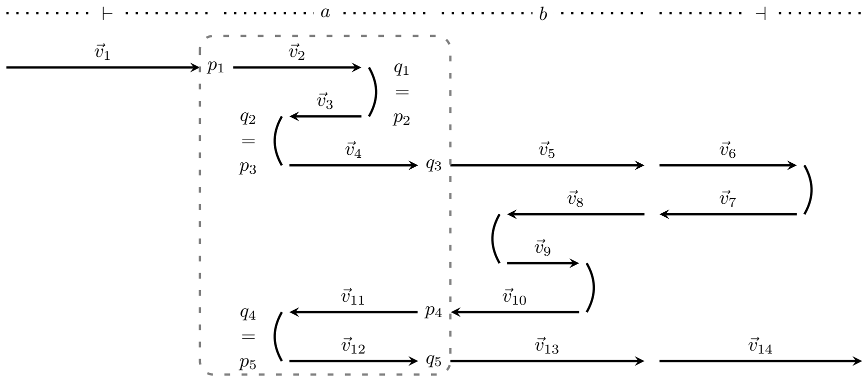


Figure 3 A a-crossing section of a run.

▶ **Definition 3.3** (matching relation). *Consider two crossing sections c_1, c_2 from the same automaton. The matching relation M is defined such that $(c_1, c_2) \in M$ if the* R*-anchorage of c_1 equals the* L*-anchorage of c_2.*

In general, an arbitrary sequence of crossing sections may not correspond to a run of a two-way automaton, that is a crossing section sequence $s = c_1, \dots, c_\ell$ such that $\mathcal{C}(\rho) \neq s$ for all run ρ. Lemma 3.4 shows that the matching property ensures the existence of such a run ρ in the two-way automaton.

▶ **Lemma 3.4.** *Consider $s = c_1, \dots, c_n$ where c_i is an a_i-crossing section such that c_1 is initial, c_n is accepting, and $(c_i, c_{i+1}) \in M$ for all $i \in \{1, \dots, n-1\}$. Then there exists an accepting two-way run ρ over $a_1 \dots a_n$ such that $\mathcal{C}(\rho) = s$. Moreover, $V(\rho) = \sum_{i=1}^{n} V(c_i)$.*

▶ **Theorem 3.5.** *Let $k \in \mathbb{N}_{\neq 0}$. Given a k-visit 2PA P, one can effectively construct an equivalent PA R that is at most exponentially bigger. Furthermore, if P is deterministic then R is unambiguous.*

Proof. Let $P = (A, \lambda, \psi)$ with $A = (Q, Q^{\mathsf{L}}, Q^{\mathsf{R}}, Q_I, Q_H, Q_F, \Delta)$ be a k-visit 2PA of dimension d with $n = |Q|$ states. In this proof we show how to construct $R = (B, \omega, \psi)$ where $B = (V, V^{\mathsf{L}}, V^{\mathsf{R}}, V_I, V_H, V_F, \Gamma)$ is a PA of dimension d having $\mathcal{O}(n^{2k})$ states such that $|\text{range}(\omega)| \leq |\text{range}(\lambda)|^{k+1}$. Note that the formula ψ is the same in both P and R.

To do so, we first consider a symbol $\top$ and extend the relation M such that $(c, \top) \in M$ holds for all accepting crossing section c. Then, we define R as follows:

- V is the set of crossing sections of length at most k
- V_I is the set of initial crossing sections and $V_H = V_F = \{\top\}$
- $\Gamma = \{(c_1, a, c_2) \in V \times (\Sigma \cup \{\vdash, \dashv\}) \times V \mid (c_1, c_2) \in M \wedge c_1 \text{ is an } a\text{-crossing section}\}$
- $\omega \colon (c_1, a, c_2) \mapsto V(c_1)$

Similar to the case of 2FA, a word u is accepted by B if there exists an accepting run of B on u, and the language $L(B)$ of B is defined as the set of words it accepts. The inclusion $L(R) \subseteq L(P)$ is a direct consequence of Lemma 3.4, while the other direction is based on the following observation: any accepting two-way run ρ has a sequence of crossing sections $\mathcal{C}(\rho)$, consecutively satisfying the matching relation. Note that, the choice of c_2 in a transition (c_1, a, c_2) is non-deterministic in general; but when P is deterministic at most one such choice of c_2 will correspond to a two-way run ensuring unambiguity. ◀

The previous crossing section construction permits to construct a one-way automaton from a bounded-visit two-way automaton. This construction is exponential in the number of states and in the number of distinct weight vectors. Nevertheless, a close inspection of the proof of Theorem 3.5, reveals that the exponential explosion in the number of distinct weight vectors can be avoided, while preserving the non-emptiness (but not the language).

▶ **Lemma 3.6.** *Let P be a k-visit 2PA. We can effectively construct a PA R with $\mathcal{O}(n^{2k})$ states and such that $L(R) = \varnothing$ iff $L(P) = \varnothing$. Furthermore, R has the same set of weight vectors and the same acceptance constraint as P.*

Proof. The construction is the same as in Theorem 3.5 but each transition of the one-way automaton $t = (c_1, a, c_2)$ is split into the following $|c_1|$ consecutive transitions, using a fresh symbol $\# \notin \Sigma$: $c_1 \xrightarrow{a} (t, 1) \xrightarrow{\#} (t, 2) \xrightarrow{\#} \ldots (t, |c_1| - 2) \xrightarrow{\#} (t, |c_1| - 1) \xrightarrow{\#} c_2$. The vectors of those transitions are defined as follows. If $c_1[i]$ denotes the ith transition of c_1, then the vector of the first R-transition is the vector of the P-transition $c_1[1]$, and the vector of any R-transition from state (t, i) is the vector of the P-transition $c_1[i + 1]$. The two languages are then equal modulo erasing $\#$ symbols. ◀

▶ **Theorem 3.7.** *Unambiguous Parikh automata have the same expressiveness as two-way deterministic (even reversible[3]) Parikh automata i.e. UPA = 2DPA. Furthermore, the transformation from one formalism to the other can be done in* ExpTime.

Proof. We only show here UPA $\subseteq$ 2DPA. The opposite direction is given by Theorem 3.5. Let $P = (A, \lambda, \psi)$ be a UPA of dimension d over Σ. Consider the alphabet $\Lambda \subseteq \mathbb{Z}^d$ as the set of vectors occurring on the transitions of P. We can see the automaton A with the morphism λ as an unambiguous finite transducer T defining a function from Σ^* to Λ^*. It is known that any unambiguous letter-to-letter one-way transducer can be transformed into an equivalent letter-to-letter deterministic two-way transducer. This result is explicitly stated in Theorem 1 of [21] which is based on a general technique introduced by Aho, Hopcroft and Ullman [1][4]. Recently, another approach has been introduced which reduces the complexity of the previous technique by one exponential [7], and allows to show that any unambiguous finite transducer is equivalent to a reversible two-way transducer exponentially bigger, yielding our result. ◀

4 Emptiness Problem

The emptiness problem asks, given a 2PA, whether the language it accepts is empty. We have seen in Example 2 how to encode the multiplication of two natural numbers encoded in unary. We can generalise this to the encoding of solutions of Diophantine equations as languages of 2PA, yielding undecidability:

▶ **Theorem 4.1.** *The emptiness problem for 2PA is undecidable.*

The proof of this theorem relies on the fact that an input position can be visited an arbitrary number of times, due to non-determinism. If instead we forbid this, we recover decidability. To prove it, we proceed in two steps: first, we rely on the result of the previous

[3] An automaton is said to be reversible if it is both deterministic and co-deterministic.
[4] Based on the technique of Aho and Hopcroft and Ullman a similar result was shown in [4] for weighted automata, namely that an unambiguous weighted automata over a semiring can be converted into an equivalent deterministic two-way weighted automata.

section showing that any bounded-visit 2PA can be effectively transformed into some (one-way) PA. This yields decidability of the emptiness problem as this problem is known to be decidable for PA. To get a tight complexity in PSPACE, we analyse this transformation (which is exponential), to get exponential bounds on the size of shortest non-emptiness witnesses. A key lemma is the following, whose proof gathers ideas and arguments that already appeared in [20, 9].

▶ **Lemma 4.2.** *Let P be a one-way Parikh automaton with n states and γ distinct weight vectors. Then, we can construct an existential Presburger formula $\varphi(x) = \bigvee_{i=1}^{m} \varphi_i(x)$ such that for all $\ell \in \mathbb{N}$, $\varphi(\ell)$ holds iff there exists $w \in L(P) \cap \Sigma^\ell$. Furthermore, $\log_2(m)$ and each φ_i are $\mathsf{poly}(|P|, \log n)$, in addition φ can be constructed in time $2^{\mathcal{O}(\gamma^2 \log(\gamma n))}$.*

▶ Remark 4.3. Note that, $\varphi(x)$ is not in *prenex normal form* (PNF) but φ_i are. Since φ is a disjunction of PNF subformulas, it can be in PNF in polynomial time.

Thanks to the lemma above, we are able to show that the non-emptiness problem for bounded-visit 2PA is PSPACE-C, just as the non-emptiness problem for two-way automata. In some sense, adding semi-linear constraints to two-way automata is for free as long as it is bounded-visit.

▶ **Theorem 4.4.** *The non-emptiness problem for bounded-visit 2PA is* PSPACE-C. *It is* NP-C *for k-visit 2PA when k is fixed.*

Proof. Consider a k-visit 2PA $P = (A, \lambda, \psi)$ of dimension d. We start with the PSPACE membership. Intuitively, we first want to apply Lemma 3.6 in order to deal with a one-way automaton, and apply then Lemma 4.2 to reduce the non-emptiness problem of the one-way Parikh automaton to the satisfiability of an existential Presburger formula. Nevertheless, we cannot explicitly transform P into a one-way automaton while keeping polynomial space. So, in the sequel, (i) we highlight an upper bound on the smallest witness of non-emptiness and based on it, (ii) we provide an NPSPACE algorithm which decides if there exists such a witness.

(i) By Lemma 4.2 applied on the PA obtained from Lemma 3.6, there exists an existential Presburger formula $\varphi(\ell) = \bigvee_{i=1}^{m} \varphi_i(\ell)$ where each $|\varphi_i|$ is polynomial in $|P|$. This formula is satisfiable iff there exists $w \in \Sigma^\ell$ such that $w \in L(P)$. By Theorem 6 (A) of [22], there exists N exponential in $|\varphi_i|$ such that φ_i is satisfiable iff $\varphi_i(\ell)$ holds for some $0 \leq \ell \leq N$. Hence, there exists N exponential in $|P|$ such that $\min\{|u| \mid u \in L(P)\} \leq N$.

(ii) The algorithm guesses a witness u of length at most N on-the-fly and a run on it. It controls its length by using a binary counter: as N is exponential in $|P|$, the memory needed for that counter is polynomial in $|P|$. The transitions of the one-way automaton obtained from Lemma 3.6 can also be computed on-demand in polynomial space. Eventually, it suffices to check that the last state is accepting and the sum $\vec{v} = (v_1, \ldots, v_d)$ of the vectors computed on-the-fly along the run satisfies the Presburger formula $\psi(x_1, \ldots, x_d)$. To do so, our algorithm constructs a closed formula $\psi^{\vec{v}}$ in polynomial time such that $\psi^{\vec{v}}$ is true iff $\vec{v} \models \psi$. It is possible by hardcoding the values of $\vec{v}$ in ψ by substituting each x_i by a term t_{v_i} of size $(\log_2(v_i))^2$ encoding v_i, by using the function symbol $\times_2$ e.g. $t_{13} = \times_2(\times_2(\times_2(1))) + \times_2(\times_2(1)) + 1$. Let us argue that $\psi^{\vec{v}}$ has polynomial size. Let μ be the maximal absolute entry of vectors of P, then $v_i \leq \mu N$, and since N is exponential in $|P|$, t_{v_i} has polynomial size in $|P|$ and $\log_2(\mu)$. Hence $\psi^{\vec{v}}$ has polynomial size, and its satisfiability can be checked in NP [22].

The lower bound is direct as it already holds for the emptiness problem of deterministic two-way automata, by a trivial encoding of the PSPACE-C intersection problem of n DFA [19].

When k is fixed, then the conversion to a one-way automaton (Lemma 3.6) is polynomial. Then, the result follows from the NP-C result for the non-emptiness of PA [9]. ◀

▶ Remark 4.5. In [9], non-emptiness is shown to be polynomial time for PA when the dimension is fixed, the values in the vectors are unary encoded and the semi-linear constraint is period-base represented. As a consequence, for all fixed d, k, the non-emptiness problem for k-visit 2PA with vectors in $\{0,1\}^d$ and a period-base represented semi-linear constraint can be solved in PTime.

5 Closure properties, universality, inclusion and equivalence problems

Since the class of 2DPA is equivalent to the class of UPA that is known to be closed under Boolean operations [3, 18], we get the closure properties of 2DPA for free, although with non-optimal complexity. We show here that they can be realised in linear-time for intersection and union. For the complement however, while the size of the state-space stays linear, the size of the acceptance condition explodes due to the transformation of negated existential Presburger formulas into existential formulas.

▶ **Theorem 5.1** (Boolean closure). *Let P, P_1, P_2 be 2DPA such that $P = (A, \lambda, \psi)$. One can construct a 2DPA $\overline{P} = (A', \lambda', \psi')$ such that $L(\overline{P}) = \overline{L(P)}$ and the size of A' is linear in the size of A. One can construct in linear-time a 2DPA $P_\cup$ (resp. $P_\cap$) such that $L(P_\cup) = L(P_1) \cup L(P_2)$ (resp. $L(P_\cap) = L(P_1) \cap L(P_2)$).*

Proof. Let us start by intersection, assuming $P_i = (A_i, \lambda_i, \psi_i)$ has dimension d_i. The automaton $P_\cap$ is constructed with dimension $d_1 + d_2$. Then $P_\cap$ first simulates P_1 on the first d_1 dimensions (with weight vectors belonging to $\mathbb{Z}^{d_1} \times \{0\}^{d_2}$), and then, if P_1 eventually reaches a halting state, it stops if it is non-accepting and rejects, otherwise it simulates P_2 on the last d_2 dimensions with vectors in $\{0\}^{d_1} \times \mathbb{Z}^{d_2}$, and accepts the word if the word is accepted by P_2 as well. The Presburger acceptance condition is defined as $\psi(\vec{x}_1, \vec{x}_2) = \psi_1(\vec{x}_1) \wedge \psi_2(\vec{x}_2)$. Note that if P_1 never reaches a halting state, then $P_\cap$ won't either, so the word is rejected by both automata. It is also a reason why this construction cannot be used to show closure under union: even if P_1 never reaches a halting state, it could well be the case that P_2 accepts the word, but the simulation of P_2 in that case will never be done. However, assuming that P_1 halts on any input, closure under union works with a similar construction. Additionally, we need to keep in some new counter c the information whether P_1 has reached an accepting state: First $P_\cup$ simulates P_1, if P_1 halts in some accepting state, then c is incremented and $P_\cup$ proceeds with the simulation of P_2. The formula is then $\psi(\vec{x}_1, \vec{x}_2, c) = (c = 1 \wedge \psi_1(\vec{x}_1)) \vee \psi_2(\vec{x}_2)$.

So, we have closure under union in linear-time as long as P_1 halts on every input. This can be used to show closure under complement, using the following observation: $\overline{L(P)} = \overline{L(A)} \cup L(A, \lambda, \neg\psi)$ and moreover, it is known that 2DFA can be complemented in linear-time into a 2DFA which always halts [11]. The formula $\neg\psi$ is universal since ψ is existential. Then, $\neg\psi$ could be converted into an equivalent existential formula using quantifier elimination [5] of doubly exponential size.

For the closure under union, we use the equality $L(P_1) \cup L(P_2) = \overline{\overline{L(P_1)} \cap \overline{L(P_2)}}$. It can be done in linear-time because the formulas for $\overline{P_1}$ and $\overline{P_2}$ are universal, and so is the formula for the 2DPA accepting $\overline{L(P_1)} \cap \overline{L(P_2)}$. By applying again the complement construction, we get an existential formula (without using quantifier eliminations). ◀

Thanks to Theorem 5.1 and decidability of non-emptiness for 2DPA, we easily get the decidability of the universality problem (deciding whether $L(P) = \Sigma^*$), the inclusion problem (deciding whether $L(P_1) \subseteq L(P_2)$), and the equivalence problem (deciding whether

$L(P_1) = L(P_2)$) for 2DPA. The following theorem establishes tight complexity bounds. It is a consequence of a more general result (Theorem 6.4) that we establish for Parikh automata with *arbitrary* Presburger formulas in Section 6.

▶ **Theorem 5.2.** *The universality, inclusion and equivalence problems are* coNExpTime-C *for 2DPA.*

Finally, we study the membership problem which asks given a Parikh automaton P and a word $w \in \Sigma^*$, whether $w \in L(P)$. Hardness was known already for PA [9].

▶ **Theorem 5.3.** *The membership problem for 2PA is* NP-C*.*

6 Parikh automata with arbitrary Presburger acceptance condition

In this section, we consider Parikh automata where the acceptance constraint is given as an arbitrary Presburger formula, that is, not restricted to existential Presburger formula, and we study the complexity of their decision problems. For all $i > 0$, a two-way Σ_i-Parikh automaton (Σ_i-2PA for short) is a tuple $P = (A, \lambda, \Psi)$ where A, λ are defined just as for 2PA and $\Psi \in \Sigma_i$. In particular, a Σ_1-2PA is exactly a 2PA. Similarly, we also define Σ_i-DPA, Σ_i-2DPA, Σ_i-PA as expected, and their Π_i counterpart (when the formula is in Π_i).

The complexity of Presburger arithmetic has been connected to the weak ExpTime hierarchy [14, 13] which resides between NExpTime and ExpSpace. It is defined as $\bigcup_{i \geq 0} \Sigma_i^{\text{Exp}}$ where:

$$\Sigma_0^{\text{P}} \overset{\text{def}}{=} \Pi_0^{\text{P}} \overset{\text{def}}{=} \text{PTime} \qquad \Sigma_{i+1}^{\text{P}} \overset{\text{def}}{=} \text{NP}^{\Sigma_i^{\text{P}}} \qquad \Pi_{i+1}^{\text{P}} \overset{\text{def}}{=} \text{coNP}^{\Sigma_i^{\text{P}}}$$
$$\Sigma_0^{\text{Exp}} \overset{\text{def}}{=} \Pi_0^{\text{Exp}} \overset{\text{def}}{=} \text{ExpTime} \qquad \Sigma_{i+1}^{\text{Exp}} \overset{\text{def}}{=} \text{NExpTime}^{\Sigma_i^{\text{P}}} \qquad \Pi_{i+1}^{\text{Exp}} \overset{\text{def}}{=} \text{coNExpTime}^{\Sigma_i^{\text{P}}}$$

Since Lemma 4.2 uses the acceptance constraint as a black box, we can generalise it as follows.

▶ **Lemma 6.1.** *For any fixed $i \in \mathbb{N}_{\neq 0}$, given a Σ_i-PA P with n states and γ distinct weight vectors, we can construct a Σ_i-formula Φ such that for all $\ell \in \mathbb{N}$ we have that $\Phi(\ell) = \bigvee_{j=1}^{m} \Phi_j(\ell)$ holds iff there exists $w \in L(P) \cap \Sigma^{|\ell|}$. Furthermore, $\log_2(m)$ and the size of each Φ_j are $\mathsf{poly}(|P|, \log(n))$, in addition Φ can be constructed in time $2^{\mathcal{O}(\gamma^2 \log(\gamma n))}$.*

Using Lemma 6.1, we can extend Theorem 4.4 to bounded-visit Σ_{i+1}-2PA. Note that the case of Σ_1-2PA is not covered by the following statement.

▶ **Theorem 6.2.** *For any fixed $i \in \mathbb{N}_{\neq 0}$, the non-emptiness problem for bounded-visit Σ_{i+1}-2PA is Σ_i^{Exp}-C.*

Proof. For the upper-bound, we show that this problem can be solved by an alternating Turing machine in exponential time, which alternates at most i times between sequences of non-deterministic and universal transitions, starting with non-deterministic transitions (called i-alternating machine in the sequel). As shown in [13], the satisfiability of Σ_{i+1}-formulas is complete for Σ_i^{Exp}-C. Hence there is an i-alternating machine $\mathcal{M}$ running in exponential time which checks the satisfiability of such formulas. Now, similar to the case of Σ_1 in Theorem 4.4, from a bounded-visit Σ_{i+1}-2PA P one can construct a Σ_{i+1}-formula which is true iff the automaton has a non-empty language. We can do so by applying Lemma 6.1 on the PA obtained[5] from Lemma 3.6. Hence, non-emptiness of a bounded-visit Σ_{i+1}-2PA

[5] Lemma 3.6 can be trivially adapted to Σ_i-formulas as acceptance condition.

reduces to satisfiability of a Σ_{i+1}-formula $\Phi(\ell) = \bigvee_{j=1}^{m} \Phi_j(\ell)$ such that $\log_2(m)$ and the size of each Φ_j are polynomial in $|P|$ and can be constructed in time $2^{\mathcal{O}(\gamma^2 \log(\gamma n))}$. However we cannot construct explicitly Φ, since its size is exponential in $|P|$. Instead we construct an i-alternating machine $\mathcal{M}'$ that first guesses a disjunct Φ_s and constructs it in exponential time, and then simulates the machine $\mathcal{M}$ on Φ_s. Recall the $\mathcal{M}$ starts with non-deterministic transitions. Thus the machine $\mathcal{M}'$ runs in exponential time, and also performs only i alternations, which provides Σ_i^{Exp} upper bound.

Hardness comes from checking if a Σ_{i+1}-sentence holds true, which is Σ_i^{Exp}-C as shown in [13]. From a Σ_{i+1}-sentence Ψ it suffices to construct a Parikh automaton $P = (A, \lambda, \Psi)$ of dimension 0 such that $L(A) \neq \varnothing$, therefore $L(P) \neq \varnothing$ iff $L(P) = L(A)$ iff Ψ holds. ◀

▶ **Theorem 6.3** (Boolean closure). *Let P, P_1, P_2 be Σ_i-2DPA. One can construct in linear time a Π_i-2DPA $\overline{P}$ and two Σ_i-2DPA $P_\cup, P_\cap$ such that $L(\overline{P}) = \overline{L(P)}$, $L(P_\cup) = L(P_1) \cup L(P_2)$ and $L(P_\cap) = L(P_1) \cap L(P_2)$.*

Proof. The constructions are the same as in the proof of the case $i = 1$ of Theorem 5.1, using closure under disjunction and conjunction of Σ_i and the fact that negating a Σ_i-formula yields a Π_i-formula. ◀

▶ **Theorem 6.4.** *For all fixed $i \in \mathbb{N}_{\neq 0}$, the universality, inclusion and equivalence problems for Σ_i-2DPA are Π_i^{Exp}-C.*

Proof. We first prove the upper bound for the most general problem which is inclusion. Let $P_i = (A_i, \lambda_i, \psi_i)$ be a Σ_i-2DPA. Note that $L(P_1) \subseteq L(P_2)$ iff $L(P_1) \cap \overline{L(P_2)} = \varnothing$. So, using Theorem 6.3 we first construct in linear-time a Π_i-2DPA $\overline{P_2} = (A'_2, \lambda'_2, \Psi'_2)$ such that $L(\overline{P_2}) = \overline{L(P_2)}$ and then $P_\cap = (A, \lambda, \Psi)$ such that $L(P_\cap) = L(P_1) \cap L(\overline{P_2})$. From the construction in Theorem 5.1 generalised to Σ_i-2DPA, recall that the formula Ψ is defined as $\Psi(\vec{x}_1, \vec{x}_2) = \Psi_1(\vec{x}_1) \wedge \Psi'_2(\vec{x}_2)$. Let $\Psi_1(\vec{x}_1) = \exists \vec{y}_1 \forall \vec{y}_2 \ldots \Omega \vec{y}_i \, [\varphi_1(\vec{x}_1, \vec{y}_1, \ldots, \vec{y}_i)]$, and $\Psi'_2(\vec{x}_2) = \forall \vec{z}_1 \exists \vec{z}_2 \ldots \mho \vec{z}_i \, [\varphi_2(\vec{x}_2, \vec{z}_1, \ldots, \vec{z}_i)]$ where $\Omega, \mho \in \{\exists, \forall\}$ such that $\Omega \neq \mho$. Hence Ψ is equivalent to the following Σ_{i+1}-formula.

$$\exists \vec{y}_1 \forall \vec{z}_1 \forall \vec{y}_2 \exists \vec{z}_2 \exists \vec{y}_3 \ldots \Omega \vec{z}_{i-1} \vec{y}_i \mho \vec{z}_i \Big[\varphi_1(\vec{x}_1, \vec{y}_1, \ldots, \vec{y}_i) \wedge \varphi_2(\vec{x}_2, \vec{z}_1, \ldots, \vec{z}_i)\Big]$$

Finally, emptiness of $P_\cap$ can be decided in Π_i^{Exp} by Theorem 6.2.

For the lower bound, we show that the universality problem of Σ_i-DPA is Π_i^{Exp}-hard. This holds even for a fixed number of states and vector values in $\{-1, 0, 1\}$, showing that the complexity comes from the formula part. From a Σ_i-formula Ψ with d free variables, we construct a Parikh automaton $P = (A, \lambda, \Psi)$ of dimension d over alphabet $\Sigma = \{a_i^+, a_i^-\}_{1 \leq i \leq d}$. Any word w over Σ defines a valuation $\mu_w(x_i) = |w|_{a_i^+} - |w|_{a_i^-}$ for all $1 \leq i \leq d$. Conversely, any valuation μ can be encoded as a word over Σ. Hence, Ψ holds for all values iff for all $w \in \Sigma^*$, we have $\mu_w \models \Psi$. We construct a deterministic one-way automaton A such that $L(A) = \Sigma^*$ and for all $w \in \Sigma^*$, the value of the run r over w is μ_w. The automaton A has one accepting and initial state q over which it loops and, when reading a_i^+ (resp. a_i^-) it increases dimension i by 1 (resp. by -1). ◀

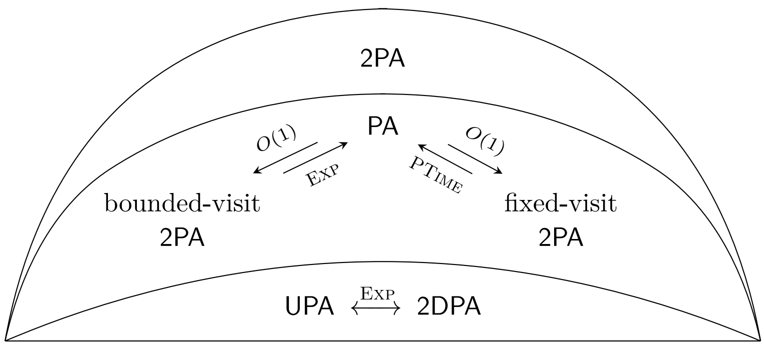
▶ Remark 6.5. Since a 2DPA is a Σ_1-2DPA, and the class coNExpTime is the same as Π_1^{Exp}, we have that Theorem 6.4 for $i = 1$ is exactly the same as Theorem 5.2.

7 Conclusion

In this paper, we have provided tight complexity bounds for the emptiness, inclusion, universality and equivalence problems for various classes of two-way Parikh automata. We have shown that when the semi-linear constraint is given as a Σ_i-formula, for $i > 1$, the complexity of those problems is dominated by the complexity of checking satisfiability or validity of Σ_i-formulas. We have shown that 2DPA (resp. bounded-visit 2PA) have the same expressive power as unambiguous (one-way) PA (resp. non-deterministic PA). Remark that the same techniques apply to show that 2UPA are equivalent to 2DPA, and hence to UPA, exactly as it is done for string transducer in [7, 8].

In terms of succinctness, it is already known that 2DFA are exponentially more succinct than FA, witnessed for instance by the family $D_n = \{uu \mid u \in \{0,1\}^* \wedge |u| = n\}$. However D_n is accepted by a PA with polynomially many states in n and vectors of dimension $2n$ which permit to store each input letters and check equality with the acceptance constraint. We conjecture that 2DPA are exponentially more succinct than PA, witnessed by the language L_n of Section 2. We leave as future work the introduction of techniques allowing to prove such results (pumping lemmas), as the dimension and acceptance constraint size has to be taken into account as well, as shown with D_n.

Finally, we plan to extend the pattern logic of [10], which intensively uses (one-way) Parikh automata for its model-checking algorithm, to reason about structural properties of two-way machines, and use two-way Parikh automata emptiness checking algorithms for model-checking this new logic.



Two-way automata	**Non-emptiness**	**Universality & Inclusion**
2PA	undecidable	undecidable
bounded-visit 2PA	PSpace-C	undecidable
fixed-visit 2PA	NP-C	undecidable
2DPA	NP-C	coNExpTime-C
bounded-visit Σ_i-2PA	$\Sigma_{i-1}^{\text{Exp}}$-C	undecidable
fixed-visit Σ_i-2PA	$\Sigma_{i-1}^{\text{Exp}}$-C	undecidable
Σ_i-2DPA	$\Sigma_{i-1}^{\text{Exp}}$-C	Π_i^{Exp}-C

Figure 4 Summary of expressivenesses and complexities where bounded-visit 2PA (resp. fixed-visit 2PA) holds for k-visit 2PA for some k (resp. for some fixed k).

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. A general theory of translation. *Mathematical Systems Theory*, 3(3):193–221, 1969.

2 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the Expressiveness of Parikh Automata and Related Models. In *NCMA'11 Proceedings*, pages 103–119, 2011.

3 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous Constrained Automata. *Int. J. Found. Comput. Sci.*, 24(7):1099–1116, 2013.

4 Vincent Carnino and Sylvain Lombardy. On Determinism and Unambiguity of Weighted Two-Way Automata. *IJFCS*, 26(8):1127–1146, 2015.

5 David C. Cooper. Theorem proving in arithmetic without multiplication. *Machine Intelligence*, 7(1):91—99, 1972.

6 Luc Dartois, Emmanuel Filiot, and Jean-Marc Talbot. Two-Way Parikh Automata with a Visibly Pushdown Stack. In *FoSSaCS'19 Proceedings*, pages 189–206, 2019.

7 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On Reversible Transducers. In *ICALP'18 Proceedings*, pages 113:1–113:12, 2017.

8 Rodrigo de Souza. Uniformisation of Two-Way Transducers. In *LATA'13 Proceedings*, pages 547–558, 2013.

9 Diego Figueira and Leonid Libkin. Path Logics for Querying Graphs: Combining Expressiveness and Efficiency. In *LICS'15 Proceedings*, pages 329–340, 2015.

10 Emmanuel Filiot, Nicolas Mazzocchi, and Jean-François Raskin. A Pattern Logic for Automata with Outputs. In *DLT'18 Proceedings*, pages 304–317, 2018.

11 Viliam Geffert, Carlo Mereghetti, and Giovanni Pighizzini. Complementing two-way finite automata. *Information and Computation*, 205(8):1173–1187, 2007.

12 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

13 Christoph Haase. Subclasses of presburger arithmetic and the weak EXP hierarchy. In *LICS'14 Proceedings*, pages 47:1–47:10, 2014.

14 Lane A. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39:299–322, 1987.

15 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

16 Oscar H. Ibarra. Reversal-Bounded Multicounter Machines and Their Decision Problems. *Journal ACM*, 25(1):116–133, 1978.

17 Wong Karianto. Parikh Automata with Pushdown Stack, 2004.

18 Felix Klaedtke and Harald Rueß. Monadic Second-order Logics with Cardinalities. In *ICALP'03 Proceedings*, pages 681–696, 2003.

19 Dexter Kozen. Lower bounds for natural proof systems. *Foundations of Computer Science*, pages 254–266, 1977.

20 Anthony Widjaja Lin. *Model checking infinite-state systems: generic and specific approaches.* PhD thesis, University of Edinburg, 2010.

21 Michal P. Chytil and Vojtech Jákl. Serial Composition of 2-Way Finite-State Transducers and Simple Programs on Strings. In *ICALP'77 Proceedings*, pages 135–147, 1977.

22 Bruno Scarpellini. Complexity of subcases of presburger arithmetic. *American Mathematical Society*, 284(1):203–218, 1984.