Parikh Images of Grammars: Complexity and Applications

Eryk Kopczyński
Institute of Informatics, Warsaw University

Anthony Widjaja To
School of Informatics, University of Edinburgh

***Abstract*—Parikh's Theorem states that semilinear sets are effectively equivalent with the Parikh images of regular languages and those of context-free languages. In this paper, we study the complexity of Parikh's Theorem over any fixed alphabet size d. We prove various normal form theorems in the case of NFAs and CFGs. In particular, the normal form theorems ensure that a union of linear sets with d generators suffice to express such Parikh images, which in the case of NFAs can further be computed in polynomial time. We then apply apply our results to derive: (1) optimal complexity for decision problems concerning Parikh images (e.g. membership, universality, equivalence, and disjointness), (2) a new polynomial fragment of integer programming, (3) an answer to an open question about PAC-learnability of semilinear sets, and (4) an optimal algorithm for verifying LTL over discrete-timed reversal-bounded counter systems.**

***Keywords*-Parikh Images, Automata, Grammars, Normal Form, Algorithms**

I. Introduction

A *semilinear set* is any subset of $\mathbb{N}^d$ that can be described as a finite union of *linear sets* over $\mathbb{N}^d$ of the form $\{v_0 + \sum_{i=1}^{m} \alpha_i v_i : \alpha_1, \ldots, \alpha_m \in \mathbb{N}\}$ for some *offset* $v_0 \in \mathbb{N}^d$ and *generators* $v_1, \ldots, v_m \in \mathbb{N}^d$. Parikh's Theorem — one of the most celebrated theorems in automata theory — states that semilinear sets are effectively equivalent with the sets of letter-counts (a.k.a. Parikh images) of regular languages and those of context-free languages [Par66].

In this paper, we study the complexity of Parikh's Theorem adopting nondeterministic finite automata (NFA) and context-free grammars (CFG) as representations of regular languages and context-free languages (respectively). Our motivations originated from the multitude of applications of Parikh's Theorem in automata theory (e.g. decision problems for semilinear sets and Parikh images of regular/context-free languages [Esp97], [Huy80], [Huy84], [Huy85], [Huy86] such as membership, universality and inclusion), the verification of well-known subclasses of Minsky counter machines [DIBKS00], [Esp97], [GMT09], [GI81], [Iba78], [TL10], [Yen96], automata and logics over unranked trees with counting [BM99], [SSM07], and equational Horn clauses [VSS05], among many others. While there is a simple polynomial-time translation from a given semilinear set S (where numbers are given in unary) to an NFA or a CFG whose Parikh image represents S, the reverse (more important) direction is not yet fully understood. All known translations from NFA and CFG to semilinear sets (e.g. see [Esp97], [Koz97], [Par66], [VSS05] and the references therein) yield at least *exponentially many* linear sets. It was not clear whether (and perhaps, to what extent) such an exponential blow-up can be avoided.

Contributions

The main contributions of this paper are various "normal form theorems" for Parikh images of NFAs and Parikh images of CFGs, along with their applications for obtaining optimal algorithms for dealing with Parikh images of NFAs and CFGs, and computational problems in many different areas of theoretical computer science. For an underlying alphabet Σ, we show that linear sets with at most $|\Sigma|$ generators always suffice to compose such Parikh images. This is a fact that was known only for $|\Sigma| = 1$ [Ram05] and for $|\Sigma| = 2$ [Abe95], but was open for larger alphabet size (see [Abe95]).

In the case of NFAs with n states and fixed alphabet size, we give a polynomial-time algorithm (exponential in $|\Sigma|$) for computing such Parikh images, each of whose linear sets is of the form $S = \{v_0 + \sum_{i=1}^{k} \alpha_i v_i : \alpha_1, \ldots, \alpha_k \in \mathbb{N}\}$ with $k \leq |\Sigma|$ satisfying the linear independence property: if $v \in S$, then the witness $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$ for this is unique. This fact is already known in the case of $|\Sigma| = 1$ as Chrobak-Martinez Theorem[1] [Chr03], [Mart02], but was open in the case of $|\Sigma| > 1$. Notice that the linear independence property allows us to check whether a vector v is a member of the linear set by using the standard Gaussian-elimination algorithm, which can be implemented in polynomial-time. Our result in fact holds also for linear grammars, which are a more expressive formalism than regular languages and to which NFAs can be translated in polynomial time. We also consider a somewhat unusual extension of these automata model with negative inputs, for which we show that our results for NFA still hold. Such an extension is, in fact, essential for deriving several of our desired applications. On the other hand, we show that the running time of algorithms for computing Parikh images of NFAs *must* have $|\Sigma|$ as an exponent. This is shown by establishing an infinite family $\{A_n\}$ of NFAs, where A_n has n states and its Parikh image contains $\Omega(n^{|\Sigma|-1})$ linear sets.

This extended abstract is a merge of [Kop10] and [To10]

[1]Unfortunately, there was a subtle error in their proofs, which were recently fixed in [To09]

We give several applications of our normal form theorem for NFAs: (1) precise complexity of membership, inclusion, universality, and disjointness for Parikh images of regular languages and linear context-free languages in the case of fixed alphabet size ranging from P to coNP, (2) a new polynomial fragment of integer programming, (3) an answer to an open question posed by Abe [Abe95] about polynomial PAC-learnability of semilinear sets, and (4) an optimal algorithm for LTL model checking over discrete-timed reversal-bounded counter systems. In most of these cases, the best previously known complexity was one exponential higher than the complexity that we derive.

What about the case of general CFGs? In this case, we show that their Parikh images could have exponentially many linear sets even over unary alphabet, i.e., $|\Sigma| = 1$. Despite this, in the case of $d := |\Sigma| \in \{1, 2\}$, we show that the Parikh image of a CFG G can be expressed as finite union of linear sets with d generators (whose magnitude is exponential in $\|G\|$), which can be further grouped into polynomially many "bundles" such that each linear set in each bundle shares the same generator(s) (we call these bundles *A, B-frames*). In the case of $d > 2$, such a normal form is shown to be impossible even for $d = 3$. In spite of this, we show that for any fixed $d > 2$ we can divide $\mathbb{N}^d$ into smaller *regions*, in each of which the number of A, B-frames is polynomial.

As an application of our normal form theorems for CFGs, we derive precise complexity for membership, universality, inclusion, and disjointness for Parikh images of CFGs over fixed alphabet ranging from NP to the second level Π_2^P of the polynomial hierarchy (previously, their known complexity was one exponential higher). These results can, in turn, be applied for deriving an optimal complexity for reachability equivalence for communication-free Petri nets in the case of fixed number of places.

Related work

It is of course well-known that semilinear sets also coincide with subsets of $\mathbb{N}^d$ expressible in Presburger arithmetic [GS66] (i.e., first-order logic over $(\mathbb{N}, +)$). Such an alternative representation has indeed been fruitfully exploited. In fact, using a technique developed by Esparza [Esp97], Verma *et al.* [VSS05] has given a linear translation from context-free grammars (equivalently, pushdown automata) to the NP-complete existential fragment of Presburger arithmetic expressing their Parikh images. Such a translation has been used to derive optimal algorithms for equational Horn clauses [VSS05] and can be used to provide an optimal complexity for membership for Parikh images of NFAs and CFGs over non-fixed alphabet size. For applications requiring Parikh's Theorem over fixed alphabet size, the translation cannot be immediately applied for producing optimal complexity. For example, even when the input is an NFA over a unary alphabet, the translation does not produce a formula in a known polynomial fragment of Presburger arithmetic.

Merging note

This extended abstract is a merge of [Kop10] and [To10]. Both authors independently discovered the normal form theorem for Parikh images of NFAs and semiliner sets with different techniques. An extension of this theorem to incorporate negative inputs and semilinear sets over all integers was proven in [To10], which was also observed in [Kop10]. We are grateful to the referee who pointed out that our results directly extend to linear grammars. In addition, the author of [Kop10] derived the two normal form theorems for CFGs. In this extended abstract, we follow the presentation of [Kop10]. To make our presentation coherent, we extend the proof of [Kop10] to incorporate an extension of the first normal form theorem to automata models with negative inputs. [For another proof of this fact, see [To10].] In addition, both authors independently discovered the application of the first normal form theorem for deciding membership of Parikh images of NFAs with fixed alphabet size, along with the lower bound for the case of unfixed alphabet size and CFGs with singleton alphabet. Applications to the problem of universality, inclusion, and disjointness as well as extesions to CFGs with fixed alphabet size were derived in [Kop10]. Other applications (integer programming, etc.) were given in [To10].

Organization

The paper is organized as follows. In Section II, we begin with some basic results concerning geometry of multisets. Section III provides some basic results concerning commutative grammars. In Section IV, we establish our normal form theorem for linear grammars. In Section V, we provide our normal form for CFGs of alphabet of size 2, which we show does not extend to alphabet of size 3. Section VI gives our normal form for CFGs of any fixed alphabet size. In Section VII, we give applications of our results.

II. Geometry of Multisets

We denote the set of non-negative integers by $\mathbb{N}$, all integers by $\mathbb{Z}$, non-negative rationals by $\mathbb{P}$, all rationals by $\mathbb{Q}$, and real numbers by $\mathbb{R}$. We use $\mathbb{P}$ for non-negative rationals instead of the more standard $\mathbb{Q}^+$ to avoid double upper indexing, as in $(\mathbb{Q}^+)^X$. We use the notation $[K..L]$ for the set of integers from K to L, and $[K; L]$ for the set of rationals from K to L.

For $F \subseteq \mathbb{R}$, F^X is the set of vectors with coordinates indexed by elements of X and coefficients from F. In particular, the elements of $\mathbb{N}^X$ are interpreted as multisets of elements of X. For $v \in F^X$, $||v||_1 = \sum_{x \in X} |v_x|$, $||v||_\infty = \max_{x \in X} |v_x|$. We say $u \geq v$ iff $u_x \geq v_x$ for each x.

By F_X^X we denote the set of matrices with coefficients in F and coordinates indexed with elements of X. For a matrix $M \in F_X^X$, M^i, the i-th column of the matrix, is a vector in F^X. For $M \in \mathbb{R}_X^X$ and $v \in \mathbb{R}^X$, Mv is a vector given by $(Mv)_j = \sum_i M_j^i v_i$, and $||M||_\infty = \max_{x\in X,\ y\in X} |M_y^x|$. Thus, for example, $[-K..K]_X^X$ is the set of matrices M with integer coefficients such that $||M||_\infty \le K$.

We can add or multiply sets of scalars, vectors, or matrices, in the usual way. For example, $U + V = \{u + v : u \in U, v \in V\}$, and $M\mathbb{N}^X$ for $M \in \mathbb{Z}_X^X$ is the set of vectors which can be obtained as a linear combination of columns of M with coefficients from $\mathbb{N}$.

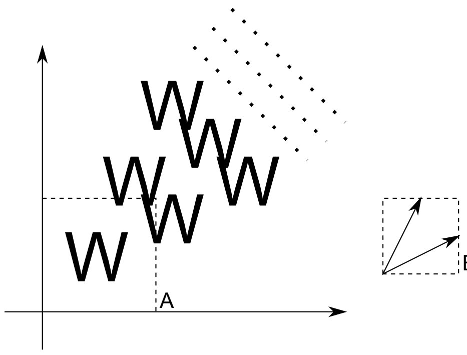
By $B^{\oplus A}$ we denote $\{\sum_{b\in B} \alpha_b b : \forall b \in B\ \alpha_b \in A\}$. A set of form $w + V^{\oplus\mathbb{N}}$, where $w \in \mathbb{Z}^\Sigma$ and V is a finite subset of $\mathbb{Z}^\Sigma$, is called a *linear set*.

Lemma 1 *Let M be a non-degenerate matrix in $\mathbb{Z}_\Sigma^\Sigma$, and let $v \in \mathbb{Z}^\Sigma$. Then $(\det M)v \in M\mathbb{Z}^\Sigma$.*

Proof: Immediate from the well known Cramer's rule. ■

Definition 2 *An A, B***-frame** *is a set of form $W + M\mathbb{N}^\Sigma$, where $W \subseteq [-A..A]^\Sigma$, and $M \in [-B..B]_\Sigma^\Sigma$.*

The following picture shows geometrically what an A, B-frame is for $|\Sigma| = 2$: a set W sitting inside of a box of size A (drawn as the letter W) is copied by shifting it in $|\Sigma| = 2$ directions. The vectors by which we are shifting are bounded by B.



Our normal form theorems will present Parikh images of grammars as unions of a small number of A, B-frames. Although the A parameter is sometimes still large, such presentation has better properties than the usual semilinear set presentation; see Lemma 17 below for an example.

III. Commutative Grammars

Since in this paper we don't care about the order of symbols in strings generated by our grammars, we define our grammars commutatively: a state (nonterminal) produces a multiset of letters and states, not a string.

Derivation trees are defined for commutative grammars similarly as for the usual ones; we omit this definition. However, we usually also abstract from derivation trees, by considering our runs as multisets rather than trees: we don't care where in the tree each transition (production) has been used, we just count the total number of occurences. We show that there is a simple condition which checks whether our multiset corresponds to some full derivation, or a „cyclic" derivation. (A similar algebraic definition of cycles is used by the algebraic topologists.)

We will also allow productions in our commutative grammars to produce negative quantities of terminal letters. While this does not make much sense for non-commutative grammars, it is very natural for commutative ones. Allowing negative productions does not increase the complexity of some of our proofs; indeed, some constructions can be made simpler when using them. Disjointness of images of two commutative regular grammars can be reduced to checking membership of 0 when negative outputs are allowed. The counterexample from Section V was also first constructed with productions with negative outputs (actually, the original construction even had productions with negative quantities of states (non-terminals) on the right side, which in general does not have a natural meaning; in this particular case, negative quantities of both terminals and states were easy to „normalize").

A **commutative grammar** is a tuple $G = (\Sigma, S, s_0, \delta)$, where Σ is a finite **alphabet**, S is a finite set of **states**, $s_0 \in S$ is an initial state, and $\delta \subseteq S \times \mathbb{Z}^\Sigma \times \mathbb{N}^S$ is a set of **transitions**. We will write transitions (s, a, t) as $s \xrightarrow{a} t$; in terms of derivations, each transition consumes the state s and produces each letter from a and each state from t. For a transition $\tau = s \xrightarrow{a} t$, $\text{source}(\tau) = s$, $\text{target}(\tau) = t$, and $\mathcal{P}(\tau) = a$.

We will assume that each state is a source of some transition. Since we want to limit things produced by the grammar in term of $|S|$, we will also assume that for each $s \xrightarrow{a} t \in \delta$, $||a||_1 \le 1$ and $||t||_1 \le 2$. (Grammars not satisfying these conditions can be easily transformed by adding additional states.) A commutative grammar satisfying $||t||_1 \le 1$ is called a **regular commutative grammar**; regular grammars are equivalent to non-deterministic finite automata, with initial state s_0 and transitions with $\text{target}(\tau) = 0$ as transitions to the final state—we prefer to speak about regular grammars rather than NFAs for the sake of uniformity. Observe also that linear grammars treated commutatively are in fact regular commutative grammars. A commutative grammar satisfying $a \ge 0$ (that is, never producing negative quantities of terminal letters) is called **positive**.

For a $D \in \mathbb{N}^\delta$, $\text{source}(D) \in \mathbb{N}^S$ counts how many each state appears as source of a transition: $\text{source}(D)(s) = \sum_{\tau:\text{source}(\tau)=s} D(\tau)$, and $\mathcal{P}(D)$ and $\text{target}(D)$ counts how many each letter and each state, respectively, is produced: $\mathcal{P}(D) = \sum_\tau D(\tau)\mathcal{P}(\tau)$, $\text{target}(D) = \sum_\tau D(\tau)\text{target}(\tau)$. The **support** of D, $\text{supp}(D) = \{s \in S : s \in \text{source}(D)\}$. We say that D is **connected from** $s \in S$ if for each

$t \in \mathrm{supp}(D)$ there is a path from s to t in D, i.e., a sequence $\tau_1, \ldots, \tau_m$ such that $\tau_i \in D$, $s = \mathrm{source}(\tau_1)$, $\mathrm{source}(\tau_{i+1}) \in \mathrm{target}(\tau_i)$, $t \in \mathrm{target}(\tau_m)$. We say that D is a **cycle from** $s \in S$ iff it is connected from s and it satisfies the **Euler condition**: $\mathrm{source}(D) = \mathrm{target}(D)$ (in terms of derivations, each state is consumed as many times as it is produced). We say that D is **run** iff it is connected from s_0 and $\mathrm{source}(D) = \mathrm{target}(D) + \{s_0\}$ (each state is consumed as many times as it is produced, except s_0 which is consumed one time more).

For a commutative grammar G, $\mathcal{P}(G) = \{\mathcal{P}(D) : D \text{ is a run in } G\}$.

This definition of a Parikh image of grammar, based on our algebraic definitions of runs and cycles, which are in turn based on connectedness and Euler condition, gives the same set as the usual one, based on derivation trees. The theorem below is a corollary of Theorem 3.1 from [Esp97]; a proof can also be found in [Kop10]. For regular grammars, it is equivalent to the classic theorems of Euler characterizing graphs with Eulerian paths and cycles.

Proposition 3 *Let G be a commutative grammar. Then:*

- *D is a run iff there is a derivation tree from s_0 where each transition τ appears $D(\tau)$ times, and all the branches are closed (i.e., they end with terminals).*
- *D is a cycle from s iff there is a derivation tree from s where each transition τ appears $D(\tau)$ times, and all the branches are closed except one with the state s at its end (we call such derivation tree* ***cyclic****).*

A cycle is called a **simple cycle** iff it cannot be decomposed as a sum of smaller non-zero cycles, and a run D is called a **skeleton run** if it cannot be decomposed as a sum of a run D_1 and a non-zero cycle C, where $\mathrm{supp}(D_1) = \mathrm{supp}(D)$. For each state $s \in S$, let $\mathcal{C}_s$ be the set of simple cycles from s, and $\mathcal{C}_T = \bigcup_{s\in T} \mathcal{C}_s$, for $T \subseteq S$. Also, let $\mathcal{Y}_s = \mathcal{P}(\mathcal{C}_s)$, and $\mathcal{Y}_T = \bigcup_{s\in T} \mathcal{Y}_s$ (cycle outputs).

Lemma 4 *Let G be a commutative grammar over Σ. If D is a simple cycle or a skeleton run, then $\mathcal{P}(D) = O\left(2^{|S|^{O(1)}}\right)$. If G is a regular grammar, then $||\mathcal{P}(D)||_1 \leq |S|$ for simple cycles, and $||\mathcal{P}(D)||_\infty \leq O(|S|^2)$ for skeleton runs.*

Proof: We start with the cycle case. We consider its cyclic derivation tree from Proposition 3.

If somewhere on the branch leading to s (the main branch) we had another s, we can easily split our cycle into two cycles (by splitting the derivation tree). A similar thing can be done if we had some state t in two places on the main branch.

A similar operation can be done when we find the same state twice on the side part of a branch (i.e. the part disjoint with the main branch).

Since we can use each state at most twice on each branch (once on the main part and once on the side part), this limits the size of a simple cycle to exponential in size of G.

The construction for skeletons is similar. Indeed, consider a skeleton run. If a state s appears $|S| + 1$ times on a branch of the production tree, it means that there exist two successive appearances of s on this branch such that the part of the tree between them can be cut off without removing any state from the support of this skeleton (otherwise each such state would have to be different and we would have $|S| + 1$ states in total).

In case of regular grammars, we use the same methods, except that there are no side branches, and thus we obtain better bounds. ■

[Kop10] contains a generalization of the part regarding regular grammars, which has been obtained purely using commutative methods (that is, the Euler condition itself). This generalization allows obtaining Theorem 8 below in a simpler way for alphabets of size 2.

IV. Normal Form for commutative regular grammars

In this section we derive our normal form theorems for semilinear sets and regular grammars.

Lemma 5 *Let $V \subseteq [-K..K]^\Sigma$ be a linearly dependent set of vectors. Then for some $\alpha \in \mathbb{Z}^V$ we have $\sum_{v\in V} \alpha_v v = 0$, where $||\alpha||_\infty \leq K^{|\Sigma|}|\Sigma|^{|\Sigma|/2}$, and $\alpha_v > 0$ for some v.*

Proof: Without loss of generality we can assume that V is a minimal linearly dependent set. Thus, we get $\sum_{v\in V} \beta_v v = 0$ for some rational coefficients $\beta \in \mathbb{Q}^V$. Let u be such that $|\beta_u| \geq |\beta_v|$ for each v. Let $M \in \mathbb{Z}^\Sigma_\Sigma$ be a non-degenerate matrix whose $|V| - 1$ columns are $V - \{u\}$ (we obtain a non-degenerate matrix since V was a minimal linearly dependent set; if $|V| < |\Sigma| + 1$, we fill up the remaining columns with independent unit vectors). From Lemma 1 we get that $|\det M|u = Mw$ for some $w \in \mathbb{Z}^\Sigma$. Let $\alpha_u = -|\det M|$, $\alpha_v = w_i$ where $v = M^i$, and $\alpha_v = 0$ for remaining vectors. We have that $\sum \alpha_v v = 0$. Moreover, we have that for some $q \in \mathbb{Q}$ we have $\beta_v = q\alpha_v$ for each v (for a minimal linearly dependent set, (β_v) is unique up to a constant); thus, $|\alpha_v| < |\alpha_u| = |\det M|$ for each v. From the classical Hadamard's bound we know that $|\det M| \leq |\Sigma|^{|\Sigma|/2}$. ■

Theorem 6 *Let $V \subseteq [-K..K]^\Sigma$. Let $i(V)$ be the set of all linearly independent subsets of V. Let $M = K^{|\Sigma|}|\Sigma|^{|\Sigma|/2}$. Then $V^{\oplus\mathbb{N}} = \sum_{W\in i(V)}(V^{\oplus[0..M]} + W^{\oplus\mathbb{N}})$. In particular, a linear subset of $\mathbb{Z}^\Sigma$, where $|\Sigma|$ is fixed, is a union of a polynomial number of A, B-frames, where both A and B are polynomial.*

Proof: We will prove that $V^{\oplus\mathbb{N}} \subseteq \sum_{W\in i(V)}(V^{\oplus[0..M]} + W^{\oplus\mathbb{N}})$; the other inclusion is obvious. Let $w = \sum_{v\in V}\gamma_v v$, where $\gamma_v \in \mathbb{N}$.

Let $P \subseteq V$ be set of vectors v for which $\gamma_v \geq M$. If $P \in i(V)$, we are done—we have already expressed w in the required form. Otherwise, by Lemma 5, $\sum_{v\in P}\alpha_v v = 0$ for some (γ_v), $|\gamma_v| \leq M$, and $\gamma_u > 0$ for some $u \in P$. This allows us to transfer multiplicites between different cycles: if we take $\gamma'_v = \gamma_v - m\alpha_v$ for $v \in P$, and $\gamma'_v = \gamma_v$ for $v \notin P$, we have $\sum_v \gamma_v v = \sum_v \gamma'_v v$. We perform this transferring operation for the largest m which does not drop any γ' below 0. Since for some v we have $\gamma_v - (m+1)\alpha_v = \gamma'_v - \alpha_v < 0$, and $\alpha_v \leq M$, we have $\gamma'_v < M$. Thus, after replacing γ with γ', the set P loses one element. We repeat this operation until $P \in i(V)$. ■

As a corollary, we get the following normal form for Parikh images of regular grammars:

Theorem 7 *Let G be a regular commutative grammar, and $K \in \mathbb{N}^\Sigma$. Then $K \in \mathcal{P}(G)$ iff there exists a run D in G, $||D||_\infty = O(|S|^{2|\Sigma|}|\Sigma|^{|\Sigma|/2})$, and simple cycles $C_1,\ldots,C_m$, $C_i \in \mathcal{C}_{\mathrm{supp}(D)}$, such that C_i are linearly independent and $K = \mathcal{P}(D) + \sum_i \alpha^i\mathcal{P}(C_i)$ for some $\alpha^1,\ldots,\alpha^m$.*

Proof:

Let D_0 be a run in G such that $K = \mathcal{P}(D_0)$. We decompose the run D_0 into a sum of simpler runs (on the same support) and simple cycles, until we get $D_0 = D_2 + \sum_{C\in\mathcal{C}_{\mathrm{supp}(D_0)}}\gamma_C C$, where D_2 is a skeleton. From Lemma 4 we get that $||D_2||_\infty \leq |\delta|$. By taking $\mathcal{P}$'s, we get $K = \mathcal{P}(D_2) + \sum_{Y\in\mathcal{Y}_{\mathrm{supp}(D_0)}}\gamma_Y Y$, where $\gamma_Y \in \mathbb{N}$ for each Y.

Let $P \subseteq \mathcal{Y}_{\mathrm{supp}(D_0)}$ be the set of such cycle outputs Y that $\gamma_Y \geq L$ for $L = |S|^{|\Sigma|}|\Sigma|^{|\Sigma|/2}$. From Theorem 6 we get that K can be decomposed in a way such that P is linearly independent.

Let $D_1 = D_2 + \sum_{Y\in\mathcal{Y}_{\mathrm{supp}(D_1)}-P}\gamma_Y Y$. Since $||D_2||_\infty = O(|S|^2)$, $\gamma_Y < L$, and there are at most $O(|S|^{|\Sigma|})$ distinct simple cycle images in $\mathcal{Y}_{\mathrm{supp}(D_0)}$ (since they are bounded by $O(|S|)$), we get that $||D_1||_\infty \leq O(L\ |S|^{|\Sigma|})$. Now, $K = D_1 + \sum_{Y\in P}\gamma_Y Y$. ■

Theorem 8 *Let G be a regular commutative grammar. Then $\mathcal{P}(G)$ is a union of a polynomial number of A,B-frames, where $A = ||D||_\infty$ and $B = |S|$. Moreover, we can compute these A,B-frames in polynomial time.*

Proof: Let a *short cycle* be a cycle C such that $||C||_1 \leq |S|$. Let $\mathcal{C}'_s$ be the set of all short cycles for the given grammar G, and $\mathcal{Y}'_s = \mathcal{P}(\mathcal{C}'_s)$. The set $\mathcal{Y}'_s$ for each state s can be calculated using simple dynamic programming. Indeed, let $A(s,t,i)$ be the set of outputs of paths from s to t of length i. For $i = 0$ we have $A(s,t,0) = \{0\}$ for $s = t$, and an empty set otherwise. We also have $A(s,u,i+1) = \sum_{t\xrightarrow{a}u\in\delta}A(s,t,i) + a$. We use these formulas to calculate sets $A(s,t,i)$ for each i by induction; these sets are bounded polynomially. $\mathcal{Y}'_s$ is then a union of $A(s,s,i)$ for $i \leq |S|$.

For each $T \subseteq S$ of size at most $|\Sigma|$, let W_T be the set of Parikh images of runs D such that $T \subseteq \mathrm{supp}(D)$, and $||D||_\infty$ satisfies the limit from Theorem 7. Again, we can calculate W_T using a similar dynamic programming algorithm (we try all possible orderings of elements of T on the path; for each ordering, we take the sum of the appropriate $A(s,t,i)$ sets).

For each $T \subseteq S$ of size at most $|\Sigma|$, and each sequence $Y_1\ldots Y_m$ of linearly independent elements of $\mathcal{Y}'_T$, take the A,B-frame given by W_T and a matrix whose columns are $Y_1\ldots Y_m$ (if $m < |\Sigma|$, we fill the remaining columns with zeros). Let U be the union of these A,B-frames.

It is easy to show that $U \subseteq \mathcal{P}(G)$. The other inclusion, $\mathcal{P}(G) \subseteq U$, follows from Theorem 7 and $\mathcal{Y}_s \subseteq \mathcal{Y}'_s$. Note that we could not use the sets $\mathcal{Y}_s$ directly, since calculating them is NP hard (by reduction of the Hamiltonian circuit problem). ■

It turns out that Theorem 8 does not hold for NFAs over unbounded alphabet size and CFGs over unary alphabet, even in descriptional complexity sense.

Proposition 9 *For each fixed positive integer d, there exists a sequence $\{A_{n,d}\}_{n=2}^\infty$ of deterministic finite automata $A_{n,d}$ with $n+1$ states over an alphabet of size d whose Parikh image contains at least $\Omega(n^{d-1})$ linear sets.*

The automaton $A_{n,d}$ recognizes the language $\Pi_{i=1}^n\left(\bigcup_{i=1}^d a_i\right)$, where here Π stands for language concatenation. We can then show that $\mathcal{P}(L(A_{n,d}))$ has $\Omega(n^{d-1})$ linear sets. For more details, see [To10].

Proposition 10 *There exists a small positive constant c and a sequence $\{G_n\}_{n=1}^\infty$ of CFGs G_n of size at most cn over the alphabet $\Sigma := \{a\}$ whose Parikh image contains precisely 2^n linear sets.*

This is done by a modification of the technique from [PSW02] of succinct encoding large numbers. Our CFG G_n contains nonterminals $S, \{A_i\}_{i=0}^{n-1}$, and $\{B_i\}_{i=0}^{n-1}$, and consists precisely of the following rules:

$$\begin{array}{rcl} S & \to & A_0\ldots A_{n-1} \\ A_i & \to & \epsilon \quad \text{for each } 0 \leq i < n \\ A_i & \to & B_i \quad \text{for each } 0 \leq i < n \\ B_i & \to & B_{i-1}B_{i-1} \quad \text{for each } 0 < i < n \\ B_0 & \to & a \end{array}$$

The initial nonterminal is declared to be S. It is not hard to show that this gives the desired CFG.

V. Normal form for commutative grammars over binary alphabet

Theorem 11 *Let $\Sigma = \{a, b\}$, and G be a positive commutative grammar over Σ. Then $\mathcal{P}(G) = \bigcup_{i \in I} Z_i$, where $|I| = O(|S|^2)$, and Z_i are A, B-frames, where $A, B = O\left(2^{|S|^{O(1)}}\right)$.*

Proof:
Let D be a run of G, and $T = \mathrm{supp}(D)$. For $Y \in \mathcal{Y}_T$, let $b(Y) = Y_b/||Y||_1$; let $M(T)$ be the matrix whose columns M^a and M^b are the elements of $\mathcal{Y}_T$ with the smallest and largest $b(Y)$, respectively. We have $||M||_\infty = O\left(2^{|S|^{O(1)}}\right)$ from Lemma 4. There are at most $|S|^2$ possible matrices $M(T)$. Let $R(M)$ be the set of runs D such that $M(\mathrm{supp}(D)) = M$. We will show that $\mathcal{P}(R(M))$ is of form $W_M + M\mathbb{N}^\Sigma$.

We decompose D as $D_2 + \sum_{C \in \mathcal{C}_T} \alpha_C C$, where D_2 is a skeleton. Thus, $\mathcal{P}(D)$ is decomposed as $\mathcal{P}(D_2) + \sum_{Y \in \mathcal{Y}_T} \alpha_Y Y + M^a\beta_a + M^b\beta_b$, where $\beta_a \in \mathbb{N}^\Sigma$, $\alpha_Y \in \mathbb{N}$. We can assume that each $\alpha_Y < \det M$, because otherwise we can replace $(\det M)Y$ by $q_aY^a + q_bY^b$, where $q_a, q_b \in \mathbb{N}$ (the coefficients are integers from Lemma 1 and non-negative since Y^a and Y^b are extreme cycles). Each Y and $\mathcal{P}(D_2)$ is $O\left(2^{|S|^{O(1)}}\right)$ from Lemma 4, and there are $O\left(2^{|S|^{O(1)}}\right)$ possible Y's, thus $D_3 = \mathcal{P}(D_2) + \sum_{Y \in \mathcal{Y}_T} \alpha_Y Y$ satisfies $||D_3||_\infty = O\left(2^{|S|^{O(1)}}\right)$.

By taking for W_M the sets of possible D_3 for all runs from $R(M)$, we get the required conclusion. ■

In this proof, we have used some properties which hold only for two letter alphabets (i.e., existence of extreme cycles). This was necessary – Theorem 11 does not generalize straightforwardly to larger alphabets.

Theorem 12 *There exists a sequence of context-free grammar (G_n) over $\{x, y, z\}$ such that G_n is of size $O(n)$, and $\mathcal{P}(G_n)$ is a union of $\Omega(2^n)$ A, B-frames.*

The construction and its proof of correctness are non-trivial; see [Kop10]. We will present the general idea of the construction. Consider the following grammar.

$$\begin{array}{lcl|l}
S & \to & 0 & SABCDEz \\
A & \to & B^2 & C^2D^2E^2xy \\
B & \to & C^2 & D^2E^2x^2y \\
C & \to & D^2 & E^2x^4y \\
D & \to & E^2 & x^8y \\
E & \to & 0 & x^{16}y
\end{array}$$

The state S generates any number of z's together with the same number of $ABCDE$'s. $ABCDE$ generates a convex 32-gon on the surface $\mathbb{N}^{\{x,y\}}$ (we get 32 corners by deciding which transition always to use for each of five states A, B, C, D, E; they are points with coordinates $(\frac{y(y+1)}{2}, y)$ for $y \in [0..31]$). Since we generate z^n together with $(ABCDE)^n$, $\mathcal{P}(G)$ is a cone (i.e., a unbounded pyramid) with 32 edges (each edge is the line $\{(\frac{zy(y+1)}{2}, zy, z\}) : z \in \mathbb{R}\}$ for some y), and hence we need more than 16 three-dimensional A, B-frames to cover $\mathcal{P}(G)$. This example generalizes to any number of states (bigger examples are constructed using the same simple rule as the example above) — we need more than 2^{n-2} A, B-frames for a grammar with n states and two transitions for each state. Note that it is quite easy to write this grammar in the limited form we use in this paper (i.e., for each derivation $s \xrightarrow{a} t$, $||a||_1 \leq 1$, $||t||_1 \leq 2$) using $O(n)$ states.

VI. Normal form for commutative grammars over larger alphabet

The proof of the generalization Theorem 18 to alphabets of larger (but still fixed) size is very long and technical. We had to omit most proofs for space reasons.

In this proof, we will require lots of constants; some of them are dependant on other. To keep our constants ordered, and make sure that there is no circular reference between them, we will name them consistently $C_1, C_2, C_3, \ldots$ through the whole section; each constant will be defined in such a way that it will depend single exponentially on the size of the grammar and/or polynomially on the lower numbered constants. (C_2 and C_{10} appear only in detailed proofs and thus do not appear in this paper; we use the same naming for consistence with the full version.) By induction, all numbered constants depend single exponentially on the size of the grammar. As usual, when we say *C_i is polynomial in C_j*, we assume that the size of alphabet Σ is fixed. (If $|\Sigma|$ is not fixed, then $C_i = O(C_j^{p(|\Sigma|)})$, where p is a polynomial.)

Let $J_\Sigma = \{x \in \mathbb{R}^\Sigma : x \geq 0, ||x||_1 = 1\}$.

Let $F_{C_1} \subseteq [0..C_5]^{\frac{1}{\Sigma}}$ (i.e., a set of some linear functions over $\mathbb{N}^\Sigma$ with integer coefficients up to C_5) be such that for each set of $|\Sigma| - 1$ vertices $V \subseteq [0..C_1]^\Sigma$, there exists a non-zero $f \in F_{C_1}$ such that $fV = 0$. This can be done with C_5 polynomial in C_1.

Let $L_{C_3} = \{0, C_3\}^\Sigma$ be the set of vertices of the hypercube of dimension $|\Sigma|$ and edge length C_3.

Let $\mathcal{R}(C_1, C_3)$ be the set of functions from $F_{C_1} \times L_{C_3}$ to $\{-1, 0, 1\}$.

For a $r \in \mathcal{R}$, let

$$\begin{aligned}
\mathrm{reg}(r) &= \left\{ v \in \mathbb{N}^\Sigma : \begin{array}{c} \forall f \in F_{C_1} \forall l \in L_{C_3} \\ \mathrm{sgn}(fv - fl) = r_{f,l} \end{array} \right\}, \\
\mathrm{Reg}(r) &= \left\{ v \in \mathbb{R}^\Sigma : \begin{array}{c} \forall f \in F_{C_1} \forall l \in L_{C_3} \\ \mathrm{sgn}(fv - fl) \in \{0, r_{f,l}\} \end{array} \right\}, \\
\tau(r) &= \left\{ x \in \mathbb{R}^\Sigma : \begin{array}{c} x \geq 0, x \neq 0, \\ \forall f \in F_{C_1} \forall l \in L_{C_3} \\ \mathrm{sgn}(fx) \in \{0, r_{f,l}\} \end{array} \right\}.
\end{aligned}$$

The following picture (Figure A) shows what J_Σ and $\tau(r) \cap J_\Sigma$ look like for $C_1 = 2$ and $|\Sigma| = 3$. (If $t \in \tau(r)$, then also $xt \in \tau(r)$ for $x > 0$; thus, a cross of $\tau(r)$ and J_Σ gives us information about the whole $\tau(r)$.)

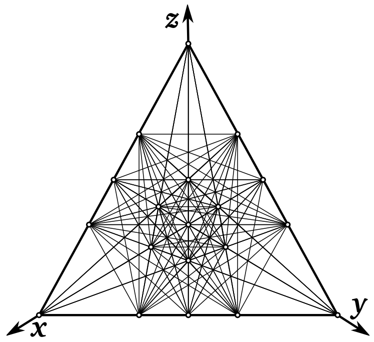
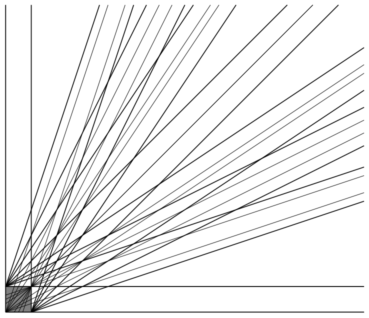


Figure A

The big equilateral triangle is J_Σ. The 19 small white circles are points $v/||v||_1$ for $v \in [0..C_1]^\Sigma$. We connect each pair of points with a line; these lines correspond to elements of F_{C_1}.

For each r, $\tau(r) \cap J_\Sigma$ is a part of the triangle defined by their relationship with each line (above, below, or on the line). Thus, each $\tau(r)$ is either an empty set, or one of the points where lines cross (including the 19 circles), or a line segment between two consecutive points where the lines cross, or a polygon bounded by lines.

The next picture shows the subdivision of $\mathbb{R}^\Sigma$ into regions (reg and Reg) for $|\Sigma| = 2$ and $C_1 = 3$; C_3 is the size of the dark square in the bottom left corner. Lines shown on the picture are boundaries between regions. They are grouped in bundles—each bundle corresponds to different element of F_{C_1}. Lines in each bundle correspond to different elements of L_{C_3}.



There are several types of regions: bounded ones, 8 unbounded regions in angular shapes (between two consective bundles of lines — there are 9 bundles of lines because vectors in $[0..3]^\Sigma$ go in 9 directions), and unbounded strip-shaped regions between semi-lines in the same bundle. It can be easily calculated that each point where bundles of lines going in different directions cross has its coordinates bounded polynomially (by $O(AB^2)$), and thus for region $\mathrm{Reg}(r)$ we can find a polynomially bounded $v' \in \mathrm{Reg}(r)$. Moreover, for each $v \in \mathrm{reg}(r)$ and each C_4 we can find a polynomially bounded $v' \in \mathrm{reg}(r)$ such that $v - v' \in C_4\mathbb{Z}^\Sigma$.

This observation is generalized in the following lemma, whose proof is very technical.

Lemma 13 *Let $C_1, C_3, C_4 \in \mathbb{N}$. There exist constants C_7 polynomial in C_1, and C_8 polynomial in C_1, C_3 and C_4, such that for each $r \in \mathcal{R}(C_1, C_3)$, for each $v \in \mathrm{reg}(r)$, if $||v||_\infty \geq C_8$, then $v = v_0 + C_4 t$, where $t \in \tau(r) \cap [0..C_7]^\Sigma$ and $v_0 \in \mathrm{reg}(r)$. Also, for each $v \in \mathrm{reg}(r)$ there is a $v' \in \mathrm{reg}(r)$ such that $||v'||_\infty < C_8$ and $v - v' \in C_4\mathbb{Z}^\Sigma$.*

Another fact we need is a polynomially bounded separation property for disjoint convex sets in $\mathbb{R}^\Sigma$. It is a well known property of convex sets that two disjoint closed convex polygons in affine space can be separated strictly with a hyperplane. In our case, the space is J_Σ, the polygons are its intersections with $P^{\oplus\mathbb{P}}$ and $Q^{\oplus\mathbb{P}}$, and we need a polynomial bound for coefficients of the hyperplane.

Lemma 14 *Let $C_7 \in \mathbb{N}$. Then there exists C_9 polynomial in C_7 such that:*

Let $P, Q \subseteq [0..C_7]^\Sigma$ such that $P^{\oplus\mathbb{P}} \cap Q^{\oplus\mathbb{P}} = \{0\}$, and $0 \notin P, Q$. Then there is a $\Phi \in \mathbb{N}^1_\Sigma$ (i.e., a linear function over $\mathbb{N}^\Sigma$ with integer coefficients) such that $\Phi P > 0$, $\Phi Q < 0$, and $||\Phi||_\infty \leq C_9$.

We use these two lemmas to show that an intersection of a region and a linear set with bounded offset and periods can be expressed as an intersection of this region and a polynomially bounded A, B-frame.

Lemma 15 *Let $C_1, C_3 \in \mathbb{N}$. Then there exists a constant C_{11} such that:*

Let $S = W + \mathcal{Y}^{\oplus\mathbb{N}}$, where $W \subseteq [0..C_3]^\Sigma$ and $\mathcal{Y} \subseteq [0..C_1]^\Sigma$. Let $r \in \mathcal{R}(C_1, C_3)$. Then there exists a matrix $M \in [0..C_1]^\Sigma_\Sigma$ and $W_1 \subseteq [0..C_{11}]^\Sigma$ such that $S \cap \mathrm{reg}(r) = (W_1 + M\mathbb{N}^\Sigma) \cap \mathrm{reg}(r)$.

After applying Lemma 15 to commutative grammars, we get:

Theorem 16 *Let G be a positive commutative grammar, and let r be a region.*

The intersection of $\mathcal{P}(G) \cap \mathrm{reg}(r)$ is an intersection of $\mathrm{reg}(r)$ and a polynomial union of C_{11}, C_1-frames, where C_{11} and C_1 are single exponential in $|G|$.

VII. Applications

Decision problem for Parikh images

Our normal form theorems lead to efficient algorithms for the basic problems regarding Parikh images of regular grammars (equivalently, Parikh images of finite automata) and (positive) context free grammars (equivalently, Parikh images of push down automata, or of finite automata on trees) over an alphabet of fixed size. These basic problems include deciding membership (check whether $v \in \mathbb{Z}^\Sigma$, where

v is given in binary, is in $\mathcal{P}(G)$), universality ($\mathcal{P}(G) = \mathbb{Z}^\Sigma$?), inclusion ($\mathcal{P}(G) \subseteq \mathcal{P}(H)$?), equivalence ($\mathcal{P}(G) = \mathcal{P}(H)$?), disjointness (are $\mathcal{P}(G)$ and $\mathcal{P}(H)$ disjoint?). The following table summarizes complexities of these problems. F stands for alphabets of fixed size, while U stands for alphabet of unfixed size.

regular languages				
alphabet size	1	2	F	U
membership	P	P	P	NPc
universality	coNPc	coNPc	coNPc	?
inclusion	coNPc	coNPc	coNPc	?
disjointness	P	P	P	coNPc
context-free languages				
alphabet size	1	2	F	U
membership	NPc	NPc	NPc	NPc
universality	Π_2^P	Π_2^P	Π_2^P	?
inclusion	Π_2^Pc	Π_2^Pc	Π_2^Pc	?
disjointness	coNPc	coNPc	coNPc	?

For example, checking membership for Parikh images of regular grammars boils down to generating the normal form (Theorem 8) and then checking whether any of the semilinear sets in the normal form contain v. Since the normal form uses linearly independent periods for each semilinear set, this can be done by solving a system of equations.

Algorithms for problems regarding context free grammars are based on the following property of unions of a small number of A, B-frames.

Lemma 17 *Let $W_i \subseteq [0..C_3]^\Sigma$, $M_i \subseteq [0..C_1]^\Sigma_\Sigma$ for $i \in I$. Let $Z_i = W_i + M_i\mathbb{N}^\Sigma$. Then for each $v \in \mathbb{N}^\Sigma$ there exists a $v' \in \mathbb{N}^\Sigma$ such that $||v'||_\infty$ is polynomial in C_1 and C_3 for fixed $|\Sigma|$, and, for each i, $v \in Z_i$ iff $v' \in Z_i$.*

Proof: Assume that the matrices M_i are non-degenerate (the case of degenerate matrices can be solved easily by changing the matrices).

Let C_4 be the least common multiple of determinants of matrices M_i, $C = O({C_1}^{O(|I|)})$.

Let $v \in \mathbb{N}^\Sigma$. Let v' be the vector v' from Lemma 13 for our v and C; We will show that it satisfies our conditions.

It is enough to check whether $v \in Z'$ iff $v' \in Z'$ for each Z' of form $w_0 + M_i\mathbb{N}^\Sigma$, where $w_0 \in W_i$.

Since M_i is non-degenerate, for some $\alpha, \alpha' \in \mathbb{Q}^\Sigma$ we have $v = w_0 + M_i\alpha$ and $v' = w_0 + M_i\alpha'$. Since v' is in the same region as v, $\alpha \geq 0$ iff $\alpha' \geq 0$. On the other hand, $v - v' \in C\mathbb{Z}^\Sigma \subseteq (\det M_i)\mathbb{Z}^\Sigma \subseteq M_i\mathbb{Z}^\Sigma$ from Lemma 1. Thus, $v \in \mathbb{N}^\Sigma$ iff $v' \in \mathbb{N}^\Sigma$. ∎

Theorem 18 *Let G_1 and G_2 be two commutative grammars over $\Sigma = \{a, b\}$. Then the problem of deciding $\mathcal{P}(G_1) \subseteq \mathcal{P}(G_2)$ is Π_2^P-complete.*

Proof of Theorem 18: The problem is Π_2^P-hard because we can reduce the problem of semilinear set inclusion [Huy80] to it.

We will first show the proof for $d = 2$.

Using Theorem 11, we can write each $\mathcal{P}(G_k)$ as $\bigcup_{i \in I_k} Z_i^k$, where I_k is a polynomial set of indices and Z_i^k is a A, B-frame, where A and B are $O\left(2^{|S|^{O(1)}}\right)$.

From Lemma 17 we get that it is enough to check inclusion on vectors $v \in \mathbb{N}^\Sigma$ of size $|v| < P = O((A+B)^{O(|I_1|+|I_2|)})$. We call such vectors small vectors.

A witness for membership of v in a grammar G is a run D such that $\mathcal{P}(D) = v$. For each small $v \in \mathcal{P}(G)$ we can find a small witness for its membership, i.e., one that does not contain non-productive cycles (i.e., C such that $\mathcal{P}(C) = 0$; such cycles can be eliminated), and thus it can be described as a string of length polynomial in size of G.

For each small v, and each small witness of membership of v in G_1, we have to find a small witness of membership of v in G_2. This can be done in Π_2^P.

For $d > 2$, we have to apply these methods separately for each region from Theorem 16. ∎

See [Kop10], [To10] for the remaining algorithms and lower bounds.

An immediate corollary of our result on inclusion of Parikh images of CFGs over a fixed alphabet is related to the problem of reachability equivalence for *communication-free Petri nets*, which was shown in [Yen96] to be solvable in double exponential time. In the case of nets with a fixed number of places, our result gives a substantially lower complexity.

Theorem 19 *Reachability equivalence for communication free nets with a fixed number of places is Π_2^P-complete.*

Integer Programming

Integer programming (IP) is the problem of checking whether a given integer program $A\mathbf{x} = b$ ($\mathbf{x} \geq \mathbf{0}$), where A is an k-by-m integer matrix and $b \in \mathbb{Z}^k$, has a solution. It is well-known that this problem is NP-complete. On the other hand, for $k = 1$, it is well-known that there is a pseudopolynomial-time algorithm for this problem (a.k.a. *knapsack problem*), which remains NP-complete under binary representation of input numbers. Furthermore, Papadimitriou [Pap81] has established a pseudopolynomial-time algorithm for solving IP, for any fixed $k \geq 1$. Using Theorem 8, we could show that the problem remains poly-time solvable (for any fixed $k \geq 1$) even if the numbers in b are given in *binary* (and A in unary).

Theorem 20 *Fix an integer $k > 0$. Given a k-by-m integer matrix A, where numbers are represented in unary, and a vector $b \in \mathbb{Z}^k$, where numbers are represented in binary. Then, checking whether the integer program $A\mathbf{x} = b$ ($\mathbf{x} \geq \mathbf{0}$) has a solution can be done in polynomial time.*

Theorem 20 is known to be true when $k = 1$, by solutions to the Frobenius problem, and has such applications as the *coin problem* [Ram05].

Polynomial PAC-learnability of semilinear sets

Valiant's notion of PAC (*Probably Approximately Correct*) learning is a standard model in computational learning theory [AB92]. In this framework, a learning algorithm is required to run in time polynomial in the size of the training sample, and output a hypothesis for an (unknown) target concept that is as *precise* as the "user" desires, given any sufficiently large training sample (but still polynomial in the reciprocals of approximation/confidence parameters).

The issues of PAC-learnability of semilinear sets have been addressed by Abe [Abe95]. In particular, learning semilinear sets of dimension 1 under binary representation of numbers is shown to be as hard as learning boolean formulas in DNF, which is (still) a major open question in learning theory. On the positive side, he shows that semilinear sets of dimension 1 and 2 can be poly-time PAC-learned when the numbers are represented in unary. To this end, he established a *normal form lemma* for semilinear sets of dimension 2 (in unary), which is simply a special case of Theorem 8 for dimension 2. However, his proof makes use of geometric facts that are specific to $\mathbb{R}^2$. For this reason, he leaves open the learnability question of semilinear sets in unary over any fixed dimension $k > 2$ [Abe95, Section 9]. Replacing Abe's normal form lemma with Theorem 8 and following the proof of [Abe95, Theorem 6.1], we can easily deduce the more general theorem (a sketch is given in [To10]).

Theorem 21 *Semilinear sets in unary representation over any fixed dimension $k \geq 1$ can be polynomially PAC-learned with respect to concept complexity.*

Verifying counter systems

Minsky's counter machines are well-known to be a Turing-powerful model of computation. In verification literature, many decidable subclasses of counter machines have been studied including *reversal-bounded* counter systems [GI81], [Iba78], [TL10], and their extensions with pushdown stacks and discrete clocks [DIBKS00], [TL10]. Intuitively, r-reversal k-counter systems are simply Minsky's counter machines with k counters, each of which can change from an incrementing mode to a decrementing mode (or vice versa) only for a fixed r number of times. Their connection to our result is due to the use of Parikh's Theorem for obtaining decidability (initially used in [Iba78]).

In [TL10], it was shown that model checking Linear Temporal Logic (LTL) over reversal-bounded counter systems with discrete clocks is solvable in double exponential time, even when one of the counters is *free* (not reversal-bounded). More precisely, if t is the number of clocks, n the number of states in the finite control, c the size (in binary) of the maximum number appearing in clock constraints, and $\|\phi\|$ the size of the input LTL formula ϕ, then model checking LTL on such systems is decidable in time exponential in n but double exponential in c, k, r, $\|\phi\|$, and t. This result was derived by carefully analyzing the complexity of effective semilinearity of the Parikh images of languages recognized by reversal-bounded counter machines [Iba78] and replacing the application of Parikh's Theorem in [Iba78] by the linear translation of [VSS05] from CFGs to existential Presburger formulas expressing their Parikh images. By replacing the application of the translation of [VSS05] by Theorem 8 and not allowing a single free counter, we can easily obtain the following upper bound.

Theorem 22 *Model checking LTL over reversal-bounded counter systems with discrete clocks is solvable in time polynomial in n and exponential in c, k, r, $\|\phi\|$, and t.*

It turns out that none of the exponents in the algorithm can be lowered, assuming the usual complexity assumption (see [To10]).

Acknowledgment

We thank anonymous referees for their useful comments. The first author thanks Sławek Lasota for his introduction to these problems, and to everyone on the Automata Scientific Excursion for the great atmosphere of research. The second author thanks Shunichi Amano, Floris Geerts, Stefan Göller, Filip Murlak, Benjamin Rubinstein, and Tony Tan for their helpful comments. The first author is supported by the Polish government grant no. N206 008 32/0810. The second author is supported by EPSRC grant E005039.

References

[Abe95] N. Abe. Characterizing PAC-Learnability of Semilinear Sets. *Inf. Comput.* 116(1):81–102 (1995)

[AB92] M. Anthony and N. Biggs. *Computational learning theory: an introduction.* Cambridge University Press, 1992.

[BHLW10] P. Barcelo, C. Hurtado, L. Libkin and P. Wood. Expressive languages for path queries over graph-structured data. To appear in *PODS'10.*

[BM99] C. Beeri and T. Milo. Schemas for Integration and Translation of Structured and Semi-structured Data. In *ICDT'99*, pages 296–313.

[Chr03] M. Chrobak. Finite automata and unary languages. In *TCS* 302:497–498 (2003)

[DIBKS00] Z. Dang, O. Ibarra, T. Bultan, R. Kemmerer and J. Su. Binary reachability analysis of discrete pushdown timed automata In *CAV'00*, pages 69–84.

[EN94] J. Esparza and M. Nielsen. Decidability issues for Petri nets – a survey. *Bulletin of the EATCS* 52:245–262 (1994)

[Esp97] J. Esparza. Petri nets, Commutative Context-Free Grammars, and Basic Parallel Processes. *Fundam. Inform.* 31(1):13–25 (1997)

[GS66] S. Ginsburg and E. H. Spanier. Semigroups, Presburger Formulas, and Languages. *Pacific Journal of Mathematics* 16(2):285–296 (1966)

[GMT09] S. Göller, R. Mayr and A. W. To. On the Computational Complexity of Verifying One-Counter Processes. In *LICS'09*, pages 235–244.

[GI81] E. Gurari and O. Ibarra. The Complexity of Decision Problems for Finite-Turn Multicounter Machines. *JCSS* 22:220–229 (1981)

[Hac76] M.H.T. Hack. *Decidability Questions for Petri Nets*. PhD Thesis, MIT, 1976.

[Huy80] T. D. Huynh. The Complexity of Semilinear Sets. In *ICALP'80*, pages 324–337.

[Huy84] D. T. Huynh. Deciding the inequivalence of context-free grammars with 1-letter terminal alphabet is Σ_2^P–complete. *TCS* 33:305–326 (1984)

[Huy85] D. T. Huynh. Complexity of equivalence problems for commutative grammars. *Inform. and Control* 66(1/2):103–121 (1985)

[Huy86] D. T. Huynh. A simple proof for the Σ_2^Pupper bound of the inequivalence problem for semilinear sets. *Inform. Process. Cybernet. (EIK)* 22:147–156 (1986)

[Hüt94] H. Hüttel. Undecidable equivalences for basic parallel processes. In *TACS'94*.

[Iba78] O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *JACM* 25:116–133 (1978)

[Kop10] E. Kopczyński. Complexity of Problems for Commutative Grammars. *CoRR abs/1003.4105*: (2010)

[Koz97] *Automata and Computability*. Springer, 1997.

[Kos82] S. R. Kosaraju. Decidability of Reachability in Vector Addition Systems. In *STOC'82*, pages 267–281.

[Mart02] A. Martinez. Efficient computation of regular expressions from unary NFAs. In *DFCS'02*, pages 174–187.

[Pap81] C. H. Papadimitriou. On the complexity of integer programming. *J. ACM* 28(4): 765–768 (1981)

[Par66] R. J. Parikh. On context-free languages. *JACM* 13(4):570–581, 1966.

[Pet81] J. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[PSW02] G. Pighizzini, J. Shallit, and M. Wang. Unary Context-Free Grammars and Pushdown Automata, Descriptional Complexity and Auxiliary Space Lower Bounds. *JCSS* 65(2): 393–414 (2002)

[Ram05] J. L. Ramírez Alfonsín. *The Diophantine Frobenius Problem*. Oxford University Press, 2005.

[Rei85] W. Reisig. *Petri Nets: An Introduction.* Springer, 1985.

[SSM07] H. Seidl, Th. Schwentick, and A. Muscholl. Counting in Trees. *Texts in Logic and Games* 2:575–612 (2007)

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.* 3:1–22 (1977)

[To09] A. W. To. Unary finite automata vs. arithmetic progressions. *IPL* 109(17):1010–1014 (2009)

[To10] A. W. To. Parikh Images of Regular Languages: Complexity and Applications. *CoRR abs/1002.1464*: (2010)

[TL10] A. W. To and L. Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *FoSSaCS'10*, pages 221–236.

[VSS05] K. N. Verma, H. Seidl and T. Schwentick. On the complexity of equational Horn clauses. In *CADE'05*, pages 337–352.

[Yen96] H. C. Yen. On reachability equivalence for BPP-nets. *TCS* 179:301–317 (1996)