Star Complexity of Parikh Images of Languages over Infinite Alphabets

Yoav Danieli
Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel

Abstract

It has been conjectured that the Parikh (commutative) image of every language over an infinite alphabet recognized by an automaton with registers is defined by a rational expression. This conjecture is known to hold for all languages recognized by one-register automata. We refine this result by proving that the star-height of the Parikh image of any language recognized by a one-register automaton is universally bounded by two. Furthermore, we show that one-register context-free languages have rational commutative images of arbitrarily high star height. We then disprove the conjecture for multiple registers, as well as disprove the equivalence of commutative expressive power between context-free grammars and automata over infinite alphabets. In other words, we show that Parikh's theorem fails for infinite alphabets.

2012 ACM Subject Classification Theory of computation → Grammars and context-free languages; Theory of computation → Automata over infinite objects

Keywords and phrases infinite alphabets, Parikh image, rational sets, star-height

Digital Object Identifier 10.4230/LIPIcs.LICS.2026.35

Related Version *Full Version*: https://arxiv.org/abs/2605.09435 [7]

1 Introduction

Finite-memory automata [16] generalize classical Rabin-Scott finite-state automata [25] to *infinite* alphabets. By equipping the automata with a finite set of registers, which store letters from the infinite alphabet during computation and restricting the power of the automaton to comparing the input letters with register contents and copying input letters to registers only, the automaton retains a fixed finite memory of input letters. Consequently, the languages accepted by finite-memory automata possess properties similar to regular languages and are, thus, termed *quasi-regular* languages.

Automata over infinite alphabets have gained increasing importance in computer science as they provide formal models for analyzing systems that operate over unbounded data domains. Such models are essential for specifying and verifying properties of XML documents, database queries, and programs with variables over infinite domains. The ability to reason about infinite alphabets while maintaining decidability of key properties makes these models particularly valuable for the formal verification of data-aware systems.

Over the years, many models of automata over infinite alphabets have been proposed (see surveys in [26, 18, 4]), though most of these models are incomparable. In the absence of a definitive model for managing infinite alphabets, evaluating a formalism requires consideration of its desirable properties, including: expressive power, closure and regular properties, decidability and complexity of classical problems, and applicability of the model.

Finite-memory automata provide a structured way to reason about such systems by focusing on patterns and repetitions of data values rather than their specific identities. This perspective is particularly useful in applications where the exact values of data are less important than their relative behavior over time. In particular, quasi-regular languages

form a subclass of nominal languages (also known as sets with atoms or Fraenkel-Mostowski sets) [1], and finite-memory automata themselves are expressively equivalent to nominal automata (also known as orbit-finite automata) [2, 3].

Notions of context-free grammars and pushdown-automata were also extended to infinite alphabets by equipping them with registers [5, 3]. They are known to be equivalent in expressive power, which is strictly greater than that of quasi-regular languages.

Over finite alphabets, Parikh's theorem [24] states that the commutative image of any context-free language coincides with that of some regular language and is, in particular, a semi-linear set. For example, the language $\{a^n b^n : n \geq 0\}$ is known for being context-free but not regular, however, its commutative image is identical to the commutative image of the regular language $\{(ab)^n : n \geq 0\}$.

A recent line of work seeks to generalize Parikh's theorem to infinite alphabets [11, 14]. Hofman et al. [15] introduced a natural extension of rational expressions onto infinite alphabets, that differs from the classical one only by allowing *orbit-finite unions*. These rational expressions are associated with rational data languages and rational data vector sets. The authors then continue to establish that all rational data languages are recognized by finite-memory automata. They proposed a program for proving an analogue of Parikh's theorem, i.e., to show that context-free grammars have Parikh images which admit rational expressions. However, the approach in [15] encounters substantial obstacles. Namely, it is only shown that one-register finite-memory automata and *binary* branching one-register context-free grammars have rational Parikh images. Later, in [20], this result has been extended to a richer model, called *hierarchical register automata*, a disciplined subclass of finite-memory automata.

In this paper, we extend the above objective and resolve the remaining open problems. In particular, our contribution is as follows:

1. We refine the methods of [15] to obtain a tight universal bound 2 on the star-height of the Parikh image of any quasi-regular language recognized by a one-register finite-memory automaton.
2. These refinements also allow us to establish the rationality of Parikh images for one-register context-free grammars of arbitrary branching degree. At the same time, we construct a family of languages generated by one-register context-free grammars whose Parikh images have arbitrarily large star-height, demonstrating an essential divergence from the automata case.
3. We refute the conjecture in [15, 20] stating that all quasi-regular languages have rational Parikh images, by exhibiting a language recognized by a three-register finite-memory automaton whose Parikh image is not rational.
4. Finally, we show that the infinite alphabet counterpart of Parikh's theorem fails by constructing a three-register context-free grammar generating a language whose Parikh image does not coincide with the Parikh image of any quasi-regular language.

2 Preliminaries

Throughout this paper, we employ the following conventions.

- Atoms denotes an infinite set whose elements, called *atoms*, are denoted by a, b, c sometimes indexed or primed.
- Infinite alphabets are denoted by uppercase Greek letters, Σ, Γ, Θ and finite alphabets are denoted by uppercase Latin letters D, H, K.

- Words over infinite alphabets are written in bold lowercase Greek letters $\boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\theta}$, etc., also sometimes indexed or primed, and range over Σ, Γ, Θ, respectively.
- Symbols occurring in a word denoted by a boldface letter are written using the same *non-boldface* letter with an appropriate subscript. For example, the letters that occur in $\boldsymbol{\sigma}'$ are denoted by σ'_i.
- For a word $\boldsymbol{\sigma} = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$, we write $[\boldsymbol{\sigma}]$ for the set of all letters occurring in $\boldsymbol{\sigma}$:

  $$[\boldsymbol{\sigma}] = \{\sigma_1, \sigma_2, \ldots, \sigma_n\},$$

  and refer to this set as the *contents* of $\boldsymbol{\sigma}$.
- For a subset $A \subseteq$ Atoms and a positive integer r, we write $A^{r\neq}$ for the set of all r-tuples of pairwise distinct elements of A and $A^{(r)} = \binom{A}{r}$ for the set of all r-elements subsets of A.
- Variables x, y, z, sometimes indexed or primed, range over Atoms.
- The formula asserting that variables $x_1, x_2, \ldots, x_n$ are pairwise distinct is written $\neq(x_1, x_2, \ldots, x_n)$, and abbreviates $\bigwedge_{i<j} x_i \neq x_j$. Similarly, for a set of variables Y, the formula $x \notin Y$ abbreviates $\bigwedge_{y\in Y} x \neq y$. For finite sets of variables X and Y, the expression $X \cap Y = \emptyset$ is interpreted as $\bigwedge_{x\in X} x \notin Y$.
- The length of a word $\boldsymbol{\sigma} \in \Sigma^*$ is denoted by $|\boldsymbol{\sigma}|$ and the cardinality of a finite set X is denoted by $\| X \|$.

2.1 Orbit-finite sets

In this section, we provide the necessary definitions and propositions regarding orbit-finite sets needed for the definition of rational sets. The notations we use are mainly from [15]. For a comprehensive presentation of sets with atoms, we refer the reader to the 'atom book' [1].

Informally, a *set with atoms* is a set whose elements may be atoms or other sets with atoms. Formally, we construct the universe of sets with atoms using an adapted cumulative hierarchy by transfinite recursion: the only set of *rank* 0 is the empty set, and for an ordinal γ, sets of rank γ contain atoms as well as sets of smaller rank. In particular, any nonempty subset $X \subseteq$ Atoms has rank one.[1]

Let Perm be the group of all permutations of Atoms. Each permutation $\pi :$ Atoms $\to$ Atoms acts on sets with atoms by consistently renaming their elements. More precisely, by another recursion, we define $\pi(X) = \{\pi(x) : x \in X\}$.

For a finite set of atoms $S \subseteq$ Atoms, an *S-permutation* is a permutation $\pi \in$ Perm fixing S, i.e., $\pi(s) = s$, for all $s \in S$. The set of all S-permutations is denoted by Perm_S. The set S is a *support* of a set X if $\pi(X) = X$, for every S-permutation π.

Supports are closed under intersection; hence, every set X has a unique *least support*, denoted $\text{supp}(X)$ and called *the support* of X. Sets supported by $\emptyset$ (i.e., invariant under all permutations) are called *equivariant*.

In this paper, we consider only *hereditarily finitely supported* sets, which are sets with finite support such that all elements of their transitive closure also have finite support (not necessarily the same).

The S-orbit of an element $x \in X$ is the set of all elements $\pi(x)$, which can be obtained by applying some S-permutation π to x: $orbit_S(x) = \{\pi(x) : \pi \in \text{Perm}_S\}$. Clearly, $x, y \in X$ are in the same S-orbit if and only if there is $\pi \in \text{Perm}_S$ such that $x = \pi(y)$.

A set X is *orbit-finite* if it is a finite union of S-orbits, for some finite subset S of Atoms.

[1] In fact, for the purpose of this paper, the ordinary recursion on natural numbers suffices.

▶ **Example 1.** Examples of orbit-finite sets are: the set of all atoms, $\textsc{Atoms}$ is a single equivariant orbit; for any atom a, the set $\textsc{Atoms} \setminus \{a\}$ is an $\{a\}$-orbit; the set of all pairs of atoms $\textsc{Atoms}^2$ has two equivariant orbits, diagonal $\{(a,a) : a \in \textsc{Atoms}\}$ and non-diagonal $\{(a,b) : a,b \in \textsc{Atoms},\ a \neq b\}$;[2] for any nonnegative integer k, the set $\textsc{Atoms}^k$ has finitely many equivariant orbits, corresponding to the equality types of a k-tuple. In contrast, the set $\textsc{Atoms}^*$ is not orbit-finite, because words of different lengths cannot lie on the same orbit.

In general, increasing the support S may refine the orbit partition of X, but the finiteness of the number of orbits is preserved.

▶ **Lemma 2** ([1, Lemma 3.16]). *A finite union of S-orbits is also a finite union of S'-orbits for every $S \subseteq S'$.*

For instance, $\textsc{Atoms}^{2\neq}$ (the set of all pairs of distinct atoms) is a single equivariant orbit that, under $S = \{a\}$, splits into three orbits $\{(a,b) : b \in \textsc{Atoms} \setminus \{a\}\}, \{(b,a) : b \in \textsc{Atoms} \setminus \{a\}\}$, and $\{(b,c) : b,c \in \textsc{Atoms} \setminus \{a\},\ b \neq c\}$.

▶ **Proposition 3** ([1, Lemma 3.24]). *Orbit-finite sets are closed under union, intersection, Cartesian product, and projection.*

A function $f : X \to Y$ is supported by S if its graph $\{(x, f(x)) : x \in X\}$ is supported by S, equivalently, for every $\pi \in \textsc{Perm}_S$, $f(\pi \cdot x) = \pi \cdot f(x)$ for all $x \in X$. We then say f is *finitely supported*. Furthermore, if X is an orbit-finite set, the union $\bigcup_{x \in X} f(x)$ is called an *orbit-finite union*.

2.2 Data words and vectors

From now on, all alphabets under consideration are orbit-finite sets, which will be called orbit-finite alphabets. Let Σ be an orbit-finite alphabet. Words $w \in \Sigma^*$ are traditionally called *data words* and languages over Σ are called *data languages*.

A *data vector* v of Σ is a function $v : \Sigma \to \mathbb{N}$ such that $v(\sigma) = 0$ for all $\sigma \in \Sigma$ except finitely many. We define:

- the *domain* of v as $\textsc{dom}(v) = \{\sigma \in \Sigma : v(\sigma) > 0\}$,
- the *size* of v as $|v| = \sum_{\sigma \in \textsc{dom}(v)} v(\sigma)$.

That is, for a data vector v and a letter $\sigma \in \Sigma$, $v(\sigma)$ is the *value* of v at σ and if v is clear from context, just the value of σ.

We represent a data vector v as a formal sum

$$v_1\sigma_1 + v_2\sigma_2 + \cdots + v_n\sigma_n\,,$$

where $\textsc{dom}(v) = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, and $v_i = v(\sigma_i)$ for $i = 1, 2, \ldots, n$. For $v_i = 1$, we write σ_i for $1\sigma_i$. Sometimes, we group atoms into disjoint subsets and write

$$v_1K_1 + v_2K_2 + \cdots + v_nK_n\,,$$

where $\textsc{dom}(v) = \bigcup_{i=1}^n K_i$ is a disjoint union and v is constant on each K_i, i.e., $v(\sigma) = v_i$ for all $i = 1, 2, \ldots, n$ and $\sigma \in K_i$.

The *Parikh vector* (or *commutative vector*) of a word $\boldsymbol{\sigma} \in \Sigma^*$ is the data vector $\textsc{Par}(\boldsymbol{\sigma}) : \Sigma \to \mathbb{N}$, where $\textsc{Par}(\boldsymbol{\sigma})(\sigma)$ is the number of appearances of a letter $\sigma \in \Sigma$ in $\boldsymbol{\sigma}$. Since we write $|\boldsymbol{\sigma}|$ for the *length* of $\boldsymbol{\sigma}$, by definition, $|\textsc{Par}(\boldsymbol{\sigma})| = |\boldsymbol{\sigma}|$.

[2] As usual, an ordered pair (x, y) is the set $\{\{x\}, \{x, y\}\}$.

The zero data vector $\mathbf{0}$ satisfies $\mathbf{0}(\sigma) = 0$ for all $\sigma \in \Sigma$. A singleton, denoted $\{\sigma\}$, maps $\sigma \mapsto 1$ and all other letters to 0, written simply as 1σ or σ when it is clear from the context that it is a data vector. The set of all singleton data vectors, $\bigcup_{\sigma\in\Sigma} \sigma$, is an orbit-finite union, because Σ is so, and this set is, naturally, denoted Σ.

Addition of data vectors is point-wise: $(v + v')(\sigma) = v(\sigma) + v'(\sigma)$ for every $\sigma \in \Sigma$. The order on data vectors is also point-wise, i.e., $v_1 \leq v_2$ if and only if for every $\sigma \in \Sigma$, $v_1(\sigma) \leq v_2(\sigma)$. Subtraction is defined whenever $v' \leq v$ by $(v - v')(\sigma) = v(\sigma) - v'(\sigma)$. If $v' \leq v$ and $\sigma \in \Sigma$ is a letter such that $v'(\sigma) = v(\sigma)$, we say that σ is *saturated* in the pair $v' \leq v$.

Orbit-finiteness of sets of data vectors coincides exactly with boundness of sizes:

▶ **Lemma 4** ([15, Lemma 1]). *A set X of data vectors is orbit-finite if and only if $\{|v| : v \in X\} \subseteq \mathbb{N}$ is bounded.*

The *Parikh image* (or *commutative image*) $\textsc{Par}(L)$ of a language $L \subseteq \Sigma^*$ is $\textsc{Par}(L) = \{\textsc{Par}(w) : w \in L\}$. Two languages $L, L' \subseteq \Sigma^*$ are *Parikh-equivalent* if they have the same Parikh images: $\textsc{Par}(L) = \textsc{Par}(L')$. For a set of data vectors X, define $\textsc{Par}^{-1}(X) = \{w \in \Sigma^* : \textsc{Par}(w) \in X\}$. Then $\textsc{Par}\left(\textsc{Par}^{-1}(X)\right) = X$, but in general $\textsc{Par}^{-1}(\textsc{Par}(L)) \neq L$, because the Parikh map is not injective.

2.3 Rational sets of data vectors

We consider sets of data vectors over a fixed orbit-finite alphabet Σ. For two sets X, Y of data vectors, their *Minkowski sum* (or addition) is

$$X + Y = \{x + y : x \in X, y \in Y\}.$$

For a set of data vectors X, its *Kleene star* is

$$X^* = \{x_1 + \ldots + x_n : n \geq 0, x_1, \ldots, x_n \in X\},$$

i.e., X^* contains all finite sums of elements of X.

We define *rational sets* of data vectors as the smallest class of sets of data vectors that contains zero $\mathbf{0}$, all singletons $\{\sigma\}$, and is closed under addition, the Kleene star, and orbit-finite unions. In particular, the empty set, all finite sets, and all orbit-finite sets of data vectors are rational. Furthermore, for an orbit-finite set of data vectors P, the set P^* is rational.

The *star-height* of a rational set of data vectors is the smallest nesting depth of the Kleene star in any rational expression that defines it. A rational set of star-height at most 1 is called a *semi-linear* set.

▶ Remark 5. Over finite alphabets, every rational set has star-height at most 1, and hence rationality coincides with semi-linearity [10]. This equivalence fails for infinite alphabets [15, Lemma 10].

3 Finite-memory models and statements of our main theorems

3.1 Finite-memory automata

We recall the definition of finite-memory automata from [16] in which a fixed finite number of distinct letters can be stored in the automaton memory during a computation. Each transition is equipped with a constraint describing the relationship between the current

register contents, the input symbol, and the register contents after the transition. Throughout this paper, input alphabets are of the form $\Sigma = H \times \text{Atoms}$, where H is a finite set, called the *labels* set. A letter $\sigma \in \Sigma$ is written $\langle h, a\rangle$, where $h \in H$ is the *label* of σ and $a \in \text{Atoms}$ is the *atom* component.

▶ **Definition 6.** *An* r-register finite-memory automaton *is a system* $\boldsymbol{A} = \langle S, s_0, F, H, D, \boldsymbol{r}_0, \mu\rangle$ *whose components are as follows.*

- $S, s_0 \in S$, *and* $F \subseteq S$ *are the finite set of* states, *the* initial state, *and the subset of* accepting states, *respectively.*
- H *is a finite set of* labels.
- $D \subset \text{Atoms}$ *is a finite set of* distinguished *atoms (constants).*
- $\boldsymbol{r}_0 = (r_{0,1}, r_{0,2}, \ldots, r_{0,r}) \in \text{Atoms}^{r\neq}$ *specifies the initial register assignment, where* $r_{0,i}$ *is the initial value of register* i*, for* $i = 1, 2, \ldots, r$*.*
- μ *is a finite set of transition rules of the form*

$$s \xrightarrow{h,\varphi} s' , \tag{1}$$

where $s, s' \in S$*,* $h \in H$*, and the* transition constraint φ *is a Boolean combination of equalities involving the variables*

$$x_1, x_2, \ldots, x_r, \quad y, \quad x'_1, x'_2, \ldots, x'_r,$$

and symbols from D*.*

Intuitively, φ must be satisfied by the current register contents $x_1, x_2, \ldots, x_r$, the input atom y, and the next register contents $x'_1, x'_2, \ldots, x'_r$.

The set $\mathcal{R} = \text{Atoms}^{r\neq}$ contains all possible register assignments, i.e., all r-tuples of pairwise distinct atoms.

A *configuration* $[s, \boldsymbol{r}] \in S \times \mathcal{R}$ of $\boldsymbol{A}$, consists of a state $s \in S$ and register values $\boldsymbol{r} = (a_1, a_2, \ldots, a_r) \in \mathcal{R}$, that means that the letter stored in the i-th register is a_i, for $i = 1, 2, \ldots, r$. If $s \in F$, it is called an *accepting* configuration. The configuration $[s_0, \boldsymbol{r}_0]$ is called the initial configuration.

A transition rule (1) and atoms $a_1, a_2, \ldots, a_r$, b, and $a'_1, a'_2, \ldots, a'_r$ that satisfy φ induce a transition

$$[s, (a_1, a_2, \ldots, a_r)] \xrightarrow{\langle h,b\rangle} [s', (a'_1, a'_2, \ldots, a'_r)] . \tag{2}$$

A *run* of $\boldsymbol{A}$ on a word $\boldsymbol{\sigma} = \langle h_1, b_1\rangle\langle h_2, b_2\rangle \cdots \langle h_n, b_n\rangle \in \Sigma^*$ is any sequence of configurations

$$[s_0, \boldsymbol{r}_0] \xrightarrow{\langle h_1,b_1\rangle} [s_1, \boldsymbol{r}_1] \xrightarrow{\langle h_2,b_2\rangle} \cdots \xrightarrow{\langle h_n,b_n\rangle} [s_n, \boldsymbol{r}_n]. \tag{3}$$

In such a case, we shall also write $[s_0, \boldsymbol{r}_0] \xrightarrow{\boldsymbol{\sigma}}{}^* [s_n, \boldsymbol{r}_n]$.

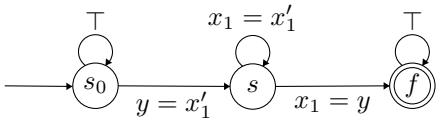
A run is accepting if it ends in an accepting configuration. In such case, we say that $\boldsymbol{\sigma}$ is accepted by $\boldsymbol{A}$. The set of all accepted words is denoted by $L(\boldsymbol{A})$ and is called the language of $\boldsymbol{A}$. Languages of finite-memory automata are called *quasi-regular languages.*

For configurations c, c', we write $L_{c,c'}(\boldsymbol{A})$ for the set of all words admitting a run starting in c and ending in c'. In particular,

$$L(\boldsymbol{A}) = \bigcup_{s\in F, \boldsymbol{r}\in\mathcal{R}} L_{[s_0,\boldsymbol{r}_0],[s,\boldsymbol{r}]}(\boldsymbol{A}) . \tag{4}$$

▶ **Proposition 7** (Invariance of FMA). *Let α be a D-permutation of* Atoms*, $\boldsymbol{\sigma} \in \Sigma^*$, and let c and c' be configurations of $\boldsymbol{A}$ such that $c \xrightarrow{\boldsymbol{\sigma}}{}^* c'$. Then $\alpha(c) \xrightarrow{\alpha(\boldsymbol{\sigma})}{}^* \alpha(c')$.*

▶ **Example 8** ([16, Example 1]). Consider a one-register finite-memory automaton $\boldsymbol{A}$, over an input alphabet of the atoms $\Sigma =$ Atoms (the set of labels H is treated as a singleton), a self-explanatory diagram of which is shown below, where the symbol $\top$ denotes the truth formula.



It is straightforward to verify that the language of $\boldsymbol{A}$ consists precisely of all words over Σ in which some letter appears more than once:

$$L(\boldsymbol{A}) = \{\sigma_1\sigma_2\cdots\sigma_n : \text{ there exist } 1 \le i < j \le n \text{ such that } \sigma_i = \sigma_j\}\,.$$

Its Parikh image, $\textsc{Par}\,(L(\boldsymbol{A}))$, consists of all data vectors which have a value of at least 2 for some atom:

$$\textsc{Par}\,(L(\boldsymbol{A})) = \{v : \text{there is an atom } a \text{ such that } v(a) \ge 2\}\,,$$

it is also equivalent to the rational expression

$$\textsc{Par}\,(L(\boldsymbol{A})) = \bigcup_{a\in\textsc{Atoms}} 2a + \textsc{Atoms}^*\,.$$

▶ **Example 9.** The language

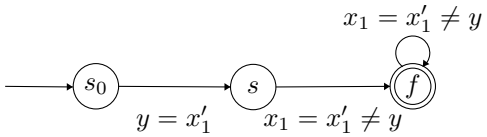
$$L_{\text{first}} = \{\sigma_1\sigma_2\cdots\sigma_n : n \ge 2, \text{ and for all } 1 < i \le n,\ \sigma_1 \ne \sigma_i\}\,,$$

consists of all words whose first letter does not repeat. Its reversal is the language that consists of all words in which the last letter does not appear earlier, namely,

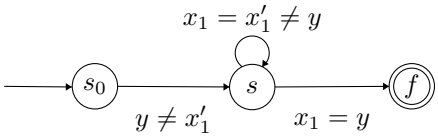
$$L_{\text{last}} = \{\sigma_1\sigma_2\cdots\sigma_n : n \ge 2, \text{ and for all } 1 \le i < n,\ \sigma_n \ne \sigma_i\}\,.$$

These languages are accepted by automata $\boldsymbol{B}$ and $\boldsymbol{C}$ shown below. Furthermore, these languages are Parikh-equivalent; their Parikh image consists of all data vectors v such that for some atom $a \in$ Atoms, $v(a) = 1$,

$$\textsc{Par}\,(L_{\text{first}}) = \textsc{Par}\,(L_{\text{last}}) = \bigcup_{a\in\textsc{Atoms}} a + (\textsc{Atoms}\setminus\{a\})^*\,.$$



(a) Automaton $\boldsymbol{B}$.



(b) Automaton $\boldsymbol{C}$.

▶ Remark 10. By Definition 6, an automaton may reassign register values non-deterministically. Such transitions, often called *guesses*, strictly increase the expressive power of finite-memory automata. For instance, the language L_{last} from Example 9 cannot be recognized without guessing (see [17, 5]).

▶ Remark 11. Definition 6 additionally stipulates that registers have predetermined initial values, that there is a single initial state, and that at every configuration all registers store distinct atoms.

These design choices are not unique; several alternative formulations appear in the literature. For instance, register values may be initialized non-deterministically, or may start empty until assigned during a run, or they may be erased and rewritten [22]. Some variants allow the same atom to appear in multiple registers simultaneously [16, 23]. All these variants have the same expressive power.

One may question the need to include both H (the finite set of labels) and D, the finite set of constants in the automaton description.

On the one hand, any data word $\langle h_1, a_1\rangle\langle h_2, a_2\rangle \cdots \langle h_n, a_n\rangle$ can be equivalently viewed as a word over atoms augmented with constants from H, namely $h_1a_1h_2a_2 \cdots h_na_n$. On the other hand, the constants from D can be stored and maintained in registers via the initial register assignment.

Nevertheless, in order to remain consistent with previous works [16, 23, 15] and to preserve the generality and robustness of the model for future applications, we adopt the most general definition.

3.2 Block finite-memory automata

In this section, we introduce a new model that generalizes finite-memory automata by allowing transitions to be labeled by finite strings (blocks) rather than single letters. Over finite alphabets, this model is known as a *generalized automaton* [9, Chapter 7, Section 10], [21, Definition 2.2.1].

▶ **Definition 12.** *An* r-register k-block finite-memory automaton *is a system* $\boldsymbol{A} = \langle S, s_0, F, H, D, \boldsymbol{r}_0, \mu\rangle$ *where the components are as in Definition 6 except that transition rules in* μ *are of the following form,*

$$s \xrightarrow{h_1h_2\cdots h_p,\varphi} s' \,, \tag{5}$$

where $s, s' \in S$*,* p *is a non-negative integer less than or equal to* k*,* $h_1, h_2, \ldots, h_p \in H$*, and the transition constraint* φ *is a Boolean combination of equalities involving the variables* $x_1, x_2, \ldots, x_r$ *(current register values),* $y_1, y_2, \ldots, y_p$ *(input atoms in the block),* $x'_1, x'_2, \ldots, x'_r$ *(next register values), and symbols from* D*.*

The class of r-register block finite-memory automata includes all r-register k-block finite-memory automata, for every positive integer k.

A configuration of $\boldsymbol{A}$ is a pair $[s, \boldsymbol{r}]$ as in standard finite-memory automata. For atoms $a_1 \ldots, a_r$, $b_1, b_2, \ldots, b_p$, $a'_1, \ldots, a'_r$ such that

$$(a_1 \ldots, a_r, b_1, b_2, \ldots, b_p, a'_1, \ldots, a'_r) \models \varphi \,,$$

rule (5) induces a transition from the automaton configuration $[s, (a_1, a_2, \ldots a_r)]$ over the string $\langle h_1, b_1\rangle\langle h_2, b_2\rangle \cdots \langle h_p, b_p\rangle$ to configuration $[s', (a'_1, a'_2, \ldots a'_r)]$:

$$[s, (a_1, a_2, \ldots a_r)] \xrightarrow{\langle h_1,b_1\rangle\langle h_2,b_2\rangle\cdots\langle h_p,b_p\rangle} [s', (a'_1, a'_2, \ldots a'_r)] \,. \tag{6}$$

If $p = 0$, (6) is an ε-transition.

A *run* of $\boldsymbol{A}$ over a word $\boldsymbol{\sigma} \in \Sigma^*$ is a sequence of configurations $c_0, c_1, \ldots, c_m$ over a decomposition of $\boldsymbol{\sigma}$, $\boldsymbol{\sigma} = \boldsymbol{\tau}_1 \boldsymbol{\tau}_2 \cdots \boldsymbol{\tau}_m$, such that

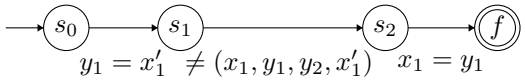
$$c_0 \xrightarrow{\boldsymbol{\tau}_1} c_1 \xrightarrow{\boldsymbol{\tau}_2} \cdots \xrightarrow{\boldsymbol{\tau}_m} c_m. \tag{7}$$

Accepting runs, the language of the automaton, and the language between configurations are defined exactly as for standard finite-memory automata.

▶ **Example 13.** Block transitions allow simultaneous comparison of several values. For instance, the language

$$L_{\text{four}} = \{abcd \in \textsc{Atoms}^4 : \neq (a, b, c, d)\}$$

is not accepted by a one-register finite-memory automaton, but is accepted by a one-register two-block finite-memory automaton, depicted below.



▶ Remark 14. Every r-register finite-memory automaton can be seen as an r-register 1-block finite-memory automaton. However, by Example 13, the class of languages recognized by one-register block finite-memory automata is strictly larger than the class recognized by one-register finite-memory automata.

In general, block comparisons can be simulated by standard letter-by-letter transitions using additional intermediate registers. More specifically, each block transition depends on the atoms of the block (at most k atoms) and the atoms in the registers before and after the transition (at most $2r$ distinct atoms). Consequently, block finite-memory automata do not increase expressive power; they recognize exactly the class of quasi-regular languages.

3.3 Finite-memory context-free grammars

Context-free grammars over infinite alphabets were introduced in [5], where they were shown to be expressively equivalent to finite-memory pushdown automata.

▶ **Definition 15.** *An* r-register finite-memory context-free grammar *is a system* $\boldsymbol{G} = \langle V, H, K, P, S, \boldsymbol{r}_0 \rangle$*, where*

- V *is a finite set of* variables *(nonterminals).*
- H *is a finite set of terminals, also called* labels*, and* $V \cap H = \emptyset$*.*
- $K \subset \Sigma$ *is a finite set of distinguished symbols.*
- P *is a finite set of production rules of the form*

$$A(X) \xrightarrow{\varphi} B_1(Y_1) B_2(Y_2) \cdots B_m(Y_m)\,, \tag{8}$$

where $A \in V$*,* m *is a non-negative integer, and* $B_1, B_2, \ldots, B_m \in H \cup K \cup V$*. The transition constraint* φ *is a Boolean combination of equalities involving:*

- *the register variables* $X = \{x_1, x_2, \ldots, x_r\}$ *(the register contents before the production),*
- *for each* i *such that* $B_i \in H$*, a single variable* $Y_i = \{y_i\}$ *representing the atom produced,*

- *for each i such that $B_i \in V$, a set of r variables $Y_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,r}\}$ representing the registers contents associated with the nonterminal B_i after the application of the production,*[3]
- $\boldsymbol{r}_0 \in \mathcal{R}$ *is the initial register assignment.*
- $S \in V$ *is the start symbol.*

The *branching degree* (or *arity*) of finite-memory context-free grammar $\boldsymbol{G}$ is the maximal integer m that appears in production rules in P, that is, the maximal size of productions. A grammar of branching degree 2 is also called a *binary* grammar.

A *variable configuration* of $\boldsymbol{G}$ is a pair $[A, \boldsymbol{r}] \in V \times \mathcal{R}$, consisting of a variable $A \in V$ and a register valuation $\boldsymbol{r} \in \mathcal{R}$. The variable configuration $[S, \boldsymbol{r}_0]$ is called the initial configuration.

A product rule of the form (8), atoms $a_1, a_2, \ldots, a_r$, and b_i for each $B_i \in H$ and $c_{i,1}, c_{i,2}, \ldots, c_{i,r}$ for each $B_i \in V$, that satisfy φ, induces a one-step derivation

$$[A, \boldsymbol{r}] \Longrightarrow X_1 X_2 \cdots X_m \,, \tag{9}$$

such that $X_i = \langle B_i, b_i \rangle \in \Sigma$ if $B_i \in H$, $X_i = B_i \in \Sigma$ if $B_i \in K$, and $X_i = [B_i, (c_{i,1}, c_{i,2}, \ldots, c_{i,r})]$ if $B_i \in V$.

For words $\boldsymbol{X}$ and $\boldsymbol{Y}$ over $(\Sigma \cup (V \times \mathcal{R}))^*$, we write $\boldsymbol{X} \Longrightarrow \boldsymbol{Y}$ if there exist words $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$ over the same alphabet and a configuration $[A, \boldsymbol{r}] \in V \times \mathcal{R}$ such that

$$\boldsymbol{X} = \boldsymbol{X}_1 [A, \boldsymbol{r}] \boldsymbol{X}_3, \qquad \boldsymbol{Y} = \boldsymbol{X}_1 \boldsymbol{X}_2 \boldsymbol{X}_3 \,,$$

and $[A, \boldsymbol{r}] \Longrightarrow \boldsymbol{X}_2$.

As usual, the reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$. The language $L(\boldsymbol{G})$ generated by $\boldsymbol{G}$ is defined by

$$L(\boldsymbol{G}) = \{\boldsymbol{\sigma} \in \Sigma^* : [S, \boldsymbol{r}_0] \Longrightarrow^* \boldsymbol{\sigma}\} \,.$$

Any such language is called a *quasi-context-free language*.

▶ **Example 16.** There is a direct translation from a block finite-memory automaton to a *linear* finite-memory grammar with the set of variables being the set of states of the automaton and the following productions,

$$s \xrightarrow{h_1 h_2 \cdots h_p, \varphi} s' \mapsto s(X) \xrightarrow{\varphi} h_1(y_1) h_2(y_2) \cdots h_p(y_p) s'(X') \,.$$

For simplicity, we sometimes use a smaller set of variables whenever possible. For instance, the production rule $A(x) \to B(x)C(y)D(y)$ is an abbreviation for the production rule

$$A(x_1) \xrightarrow{x_1 = y_1, y_2 = y_3} B(y_1)C(y_2)D(y_3) \,.$$

▶ **Example 17** ([15, Example 3])**.** Consider the alphabet $\Sigma = \{l, r\} \times \textsc{Atoms}$. Let $\boldsymbol{G}_{\text{mirror}}$ be a one-register context-free grammar with $V = \{S\}$ and production rules

$$P = \{S(x) \to l(x)S(y)r(x), \quad S(x) \to l(x)r(x)\} \,.$$

Then $L(\boldsymbol{G}_{\text{mirror}})$ consists of all words of the following form,

$$\langle l, a_1 \rangle \langle l, a_2 \rangle \cdots \langle l, a_n \rangle \langle r, a_n \rangle \langle r, a_{n-1} \rangle \cdots \langle r, a_1 \rangle \,,$$

for atoms $a_1, a_2, \ldots, a_n \in \textsc{Atoms}$.

[3] When $B_i \in K$, Y_i is irrelevant. We set it $\emptyset$, to keep uniformity of notation.

Observe that this language is Parikh-equivalent to the language $L_{\text{mirror.com}}$ which consists of all words of the form,

$$\langle l, a_1\rangle\langle r, a_1\rangle\langle l, a_2\rangle\langle r, a_2\rangle \cdots \langle l, a_n\rangle\langle r, a_n\rangle \,.$$

The latter is recognizable by a finite-memory automaton.

Namely, the Parikh image of both languages is the following rational set,

$$\left(\bigcup_{a \in \text{Atoms}} \langle l, a\rangle + \langle r, a\rangle \right)^* .$$

3.4 Parikh's theorem and main results

Recall Parikh's theorem [24] stating that the commutative image of every context-free language over a *finite* alphabet coincides with the commutative image of some regular language. Over finite alphabets, rational sets of data vectors coincide with semi-linear sets (cf. Remark 5). Parikh's proof shows that commutative images of context-free languages are exactly the semi-linear sets and that regular languages are expressive enough to capture every semi-linear set.

Guided by examples such as Example 17, it is natural to conjecture an analogue of Parikh's theorem for infinite alphabets. One might hope to identify an appropriate notion of semi-linear sets for languages over infinite alphabets, show that commutative images of quasi-context-free languages satisfy this notion, and then prove that quasi-regular languages are expressive enough to capture every such set.

This program was initiated in [15], where the authors introduced *rational sets of data vectors*. They showed that every rational set of data vectors is the commutative image of a rational data language which is necessarily quasi-regular. Thus, to obtain a Parikh-type correspondence, it would suffice to show that the commutative image of every quasi-context-free language is rational. However, even for finite-memory automata, this rationality was nontrivial, and the authors established it only for one-register automata; later, in [20] this result was extended to the more expressive model of *hierarchy register automata*, which still recognize a proper subclass of the quasi-regular languages.

In retrospect, this strategy was doomed. We show that finite-memory context-free grammars with three registers can generate commutative images that do not arise from *any* finite-memory automaton.

▶ **Theorem 18.** *Commutative images of quasi-context-free languages form a strictly larger class than the class of commutative images of quasi-regular languages.*

Moreover, the paradigm itself breaks down: even within the class of quasi-regular languages, commutative images need not be rational, even for automata with three registers.

▶ **Theorem 19.** *Commutative images of quasi-regular languages are not always rational.*

So, the prospect of a Parikh-type theorem for infinite alphabets collapses in general. Nevertheless, for the important restricted case of one register, the situation is more favorable.

The restriction to a single register, while still allowing a finite, unbounded set of distinguished symbols, is significant. One-register automata are known to enjoy much better algorithmic and structural properties than their multi-register counterparts.

For instance, consider finite-memory automata without guessing, where universality is decidable for one-register automata but becomes undecidable already for two registers; the alternating one-register model has decidable nonemptiness [8, 13, 12]; the latter is

tightly connected to linear temporal logic with the freeze quantifier [8]. Furthermore, for nondeterministic one-register automata, determinisation is decidable [6], and over ordered alphabets, the intersection of nondeterministic and co-nondeterministic one-register languages lies within the deterministic class [19].

Regarding Parikh images, it is known that the commutative image of any one-register quasi-regular language is rational and, therefore, of finite star-height [15, Theorem 6]. We strengthen this result by establishing a tight universal upper bound,

▶ **Theorem 20.** *Commutative images of one-register quasi-regular languages are of star-height at most two.*

This bound is optimal: [15, Lemma 10] exhibits a one-register quasi-regular language whose commutative image has star-height at least two. Moreover, we extend this result to the strictly larger class of languages recognized by one-register block finite-memory automata (cf. Remark 14).

▶ **Theorem 21.** *Commutative images of one-register block finite-memory automata are of star-height at most two.*

For one-register context-free grammars, commutative images are known to be rational for binary grammars [15, Theorem 7]. Using one-register block finite-memory automata, we generalize this result to one-register finite-memory context-free grammars of arbitrary branching degree.

▶ **Theorem 22.** *Commutative images of one-register finite-memory context-free languages are rational.*

However, we show that, unlike in the automata case, there is no universal bound on their star-height. Specifically, we demonstrate the existence of grammars with arbitrarily high star-height.

▶ **Theorem 23.** *For every $n \in \mathbb{N}$, there is a one-register context-free grammar generating a language whose commutative image has star-height n.*

3.5 Organization of the paper

Due to space constraints, most technical proofs of Theorems 18–23 are omitted and can be found in the full version of this article [7]. In this paper, we therefore focus on two canonical results: one positive and one negative. The positive result establishes that a canonical language, called *anti-paths*, has a semi-linear Parikh image, that serves as the key for proving a universal upper bound on star-height. The negative result is the full proof of Theorem 19. The example is likewise canonical and underlies all separation and hardness results in this paper.

Namely, the rest of the paper is organized into two parts.

Separations and lower bounds

We introduce the notion of *linear forms* in Section 4, that serves as the main technical tool. Using this framework, we prove Theorem 19 in Section 5. The proof of Theorem 18 builds on the same ideas.

Universal upper bound

We show a motivating result on star complexity. Section 6 shows that the Parikh image of the language of anti-paths is semi-linear. Extending this argument to multi-dimensional anti-paths is technically involved and leads to the proofs of Theorems 20, 21, and 22.

4 Linear forms of sets of data vectors

We lift the notions of linearity to infinite alphabets. Linear forms will be defined recursively, according to the star-height of the rational expressions describing them.

For star-height $h = 0$, a rational set is necessarily orbit-finite. Such a set is said to be in linear form of height 0 if it is presented as an orbit-finite union of singletons:

$$\bigcup_{i \in I} g_i \,, \tag{10}$$

where g_i is a data vector for $i \in I$.

For star-height $h > 0$, a rational set is in linear form if it is presented as an orbit-finite union of singletons and lower-height linear forms, namely:

$$\bigcup_{i \in I} g_i + P_i^* \,, \tag{11}$$

where g_i is a data vector and P_i is a rational set of data vectors that is already in a linear form of star-height strictly smaller than h, for all $i \in I$.

▶ **Example 24.** Let

$$P_a = \bigcup_{b \in \text{Atoms} \setminus \{a\}} a + 2b, \quad \text{for each } a \in \text{Atoms}\,,$$
$$R = \bigcup_{a \in \text{Atoms}} a + (P_a)^* \,.$$

Intuitively, R consists of all the data vectors which for some atom a, contain a exactly $k+1$ times and contain $2k$ duplicates of atoms distinct from a, for some $k \geq 0$. For instance, $3a + 2b + 2c$ and $3a + 4b$ belong to R. Furthermore, R is in linear form of height one.

▶ **Proposition 25** (Cf. [15, Proposition 9])**.** *Every rational set of data vectors admits a representation in linear form of some finite height.*

▶ **Definition 26.** *Let R be a rational set of data vectors presented in linear form, and let $v \in R$ be a data vector. A* parsing tree *of v is a rooted tree whose vertices are labeled by pairs consisting of a data vector and a finite subset of atoms. Parsing trees are defined recursively according to the star-height of R.*

For star-height $h = 0$, R is of the form (10), therefore, there is $i \in I$ such that $v = g_i$. Let $S_v = \text{SUPP}(f)$ where f is the mapping $x \mapsto g_x$, which is a finitely supported function, because it is an orbit-finite union. The parsing tree T_v consists of a single root vertex labeled by (v, S_v).

For star-height $h > 0$, R is of the form (11). Then there are $i \in I, v_0 = g_i$ and $v_1, \ldots, v_k \in P_i$, such that $v = \sum_{j=0}^{k} v_j$. The root of the parsing tree T_v is labeled by $(v_0, \text{SUPP}(f))$ (where f is the finitely supported function $x \mapsto (g_x, P_x)$), and it has k children whose sub-trees are the parsing tree $\{T_{v_j}\}_{j=1}^{k}$ of the corresponding data vectors in P_i.

Note that the depth of any parsing tree T_v is at most the star-height of the rational set R.

Given a parsing tree T and a sub-tree T' of T (i.e., a node together with all its descendants), we define the *value* of T' as the sum of all data vectors occurring in its vertex labels. In particular, the value of a parsing tree T_v is the vector v itself.

▶ **Example 27.** Continuing Example 24, the data vectors $3a + 2b + 2c$ and $3a + 4b$ both admit parsing trees of depth one, as imposed by the structure of R. Their parsing trees are depicted in Figure 2.

(a) Parsing tree for $3a + 2b + 2c$. **(b)** Parsing tree for $3a + 4b$.

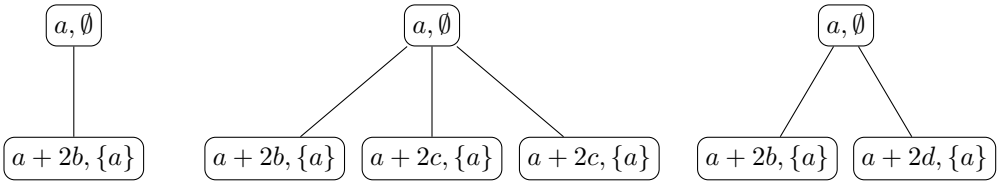
Figure 2 Parsing trees for data vectors in Example 24.

The lemma below immediately follows from Definition 26.

▶ **Lemma 28.** *Let R be a rational set of data vectors in linear form, and let T be a parsing tree of a data vector in R. The following operations on T yield a parsing tree for some vector in R:*

1. *Pruning sub-trees: deleting sub-trees.*
2. *Duplicating sub-trees: adding copies of existing sub-trees at the same node.*
3. *Permuting sub-trees: applying a permutation to the atoms occurring in any sub-tree, provided the permutation preserves the atom-set in the label of its root.*

▶ **Example 29.** Continuing Examples 24 and 27, Figure 3 depict possible operations on the parsing tree of $3a + 2b + 2c$ on the sub-tree of the vertex labeled $(a + 2c, \{a\})$, which creates parsing trees for the data vectors, $2a + 2b$, $4a + 2b + 4c$, and $3a + 2b + 2d$.



(a) Pruning a sub-tree. **(b)** Duplicating a sub-tree. **(c)** Permuting a sub-tree $(c \mapsto d)$.

Figure 3 Parsing tree operations applied to a parsing tree from Example 24.

▶ **Proposition 30.** *Let R be a rational set of data vectors in linear form. There are constants N and M, such that, for every $v \in R$ and every parsing tree T_v of v, every label (u, A) in T_v satisfies $|u| \leq N$ and $\| A \| \leq M$.*

Proof. The proof is by induction on the star-height of R.

For star-height $h = 0$, R is of the form (10). Let $N = \max |g_i|$ and $M = \| \operatorname{SUPP}(x \mapsto g_x) \|$. Since I is orbit-finite and all elements in the same orbit are of equal size, these maxima exist and are finite.

For star height $h > 0$, R is of the form (11). Let N_i, M_i be the constants obtained by the induction hypothesis for P_i, for $i \in I$. Define $N_i' = \max\{|g_i|, N_i\}$. Then, let $N = \max N_i'$ and $M = \max M_i$. These maxima are finite because I is orbit-finite, and only finitely many orbits need to be considered. ◀

5 Proof of Theorem 19

We precede the presentation of the separating language with an introduction to the notion of *commutative stability*, that is the key concept for the construction of the separating language.

5.1 Commutative stability

The Parikh map is not injective, hence, the same data vector may be the Parikh vector of multiple different words. Moreover, parsing-tree operations allow us to transform a data vector in the set into other data vectors in the set. To analyze the Parikh image of a language it is useful to understand the structure of witnesses that are the result of a parsing-tree operation. To this end, we ask the following question: given $v \in \textsc{Par}(L)$ and $w \leq v$ such that $v + w \in \textsc{Par}(L)$, under what conditions, the structure of $\textsc{Par}^{-1}(v + w)$ is close to that of $\textsc{Par}^{-1}(v)$, in some sense of proximity.

The examples below illustrate this notion.

▶ **Example 31.** Consider the finite-alphabet language,

$$L_{3\text{Block}} = \{(abc)^{n_1}(cde)^{n_2}(efg)^{n_3} : n_i \geq 1 \text{ for } i = 1, 2, 3\}.$$

In this language, every word is uniquely determined by its Parikh vector. Indeed, the values of a and b are the same and determine the first exponent n_1; the value of d determines the second exponent; and the values of f and g are the same and determine the third exponent n_3.

Let $z = (abc)^{n_1}(cde)^{n_2}(efg)^{n_3} \in L_{3\text{Block}}$ with $v = \textsc{Par}(z)$, and let $w \leq v$ be such that there exists $z' \in L_{3\text{Block}}$ with $\textsc{Par}(z') = v + w$. Since the values of a, d, and g have increased or remained unchanged, we conclude that $z' = (abc)^{m_1}(cde)^{m_2}(efg)^{m_3}$ with $m_i \geq n_i$ for $i = 1, 2, 3$. Thus, the increase of the Parikh vector corresponds to an increase of the exponents.

Therefore, we call $L_{3\text{Block}}$ a commutatively stable language.

This stability phenomenon fails for blocks of size two.

▶ **Example 32.** Consider the finite-alphabet language,

$$L_{2\text{Block}} = \{(ab)^{n_1}(bc)^{n_2}(cd)^{n_3} : n_i \geq 1 \text{ for } i = 1, 2, 3\}.$$

Again, every word is uniquely determined by its Parikh vector: the values of a and d determine the first and third exponents, while the second exponent is determined by the value of either b or c.

However, for $z = (ab)^{n_1}(bc)^{n_2}(cd)^{n_3} \in L_{2\text{Block}}$ with $n_2 \geq 2$, the word $z' = (ab)^{n_1+1}(bc)^{n_2-1}(cd)^{n_3+1}$ also belongs to $L_{2\text{Block}}$. Although $\textsc{Par}(z') \geq \textsc{Par}(z)$, the exponents structure is broken.

In general, we do not need that all the words in the language to be commutatively stable, but a large class of them.

To extend Example 31 for infinite alphabets, we introduce a delimiter symbol # whose value determines the number of blocks and, consequently, bounds the number of distinct atoms in a word of L. Moreover, for infinite alphabets, there is no guarantee that atoms

preserve their relative order (for instance, $(ab)(bc)(cd)$ may be commuted to $(ad)(db)(bc)$). To enforce that relative order is preserved, it suffices to choose block exponents that grow exponentially. In this case, the atoms with the largest multiplicities must belong to the same blocks, which fixes the block order and yields commutative stability.

5.2 The separating language

Let L be the language of all words of the form

$$(\tau_1\tau_2)^{n_0}\#\#(\tau_2\tau_3\tau_4)^{n_1}\#\#(\tau_4\tau_5\tau_6)^{n_2}\#\#(\tau_6\tau_7\tau_8)^{n_3}\cdots \\ \#\#(\tau_{2k}\tau_{2k+1}\tau_{2k+2})^{n_k}\#\#(\tau_{2k+2}\tau_{2k+3})^{n_{k+1}}\,. \tag{12}$$

where

1. $k \geq 0$;
2. $n_i \geq 1$, $i = 0, 1, \ldots, k+1$; and
3. $\tau_i \in \textsc{Atoms} \setminus \{\#\}$, $i = 1, 2, \ldots, 2k+3$.

▶ **Lemma 33.** *L is recognized by a deterministic finite-memory automaton with three registers.*

Proof. The automaton stores the relevant triple of atoms that appear in a block (or the pair of atoms for the first and last blocks) and checks consistency of adjacent blocks. The double separator ## synchronizes the transitions between consecutive blocks. ◀

▶ **Theorem 34.** $\textsc{Par}(L)$ *is irrational.*

Obviously, Theorem 19 follows from Lemma 33 and Theorem 34.

The proof of Theorem 34 is by *reductio ad absurdum*. The remainder of this section is the detailed proof; first we give the proof idea and then develop the required technical lemmas.

5.3 Proof idea

The main idea is to show that the language L contains a sequence of commutatively stable words with an increasing number of blocks. Due to commutative stability, we shall show that in $\textsc{Par}(L)$, each block is generated by a number of star expressions associated with the block exponent. Consequently, these sub-expressions must remember all atoms occurring within the block.

However, the blocks are not independent: the last atom of each block is also the first atom of the next block. Such overlap forces a star sub-expression that generates a block to also remember its immediate neighbors. By iteration, this neighbor constraint propagates across the blocks, requiring distinct atoms from arbitrarily many adjacent blocks to appear in the support of a single star sub-expression.

When the number of blocks grows, the number of atoms that must be remembered by a single star sub-expression grows as well. Hence, the required support becomes unbounded, contradicting the bounded-support property of rational expressions. Hence, the Parikh image of the language is not rational.

5.4 Commutative stability in infinite alphabets

Consider the sub-language $L' \subset L$ containing words in which all symbols τ_i are pairwise distinct. We focus on $\textsc{Par}(L') \subseteq \textsc{Par}(L)$.

▶ **Proposition 35.** *For every* $v \in \textsc{Par}(L)$,

$$|\textsc{dom}(v)| \leq v(\#) + 2\,,$$

where the equality holds if and only if $v \in \textsc{Par}(L')$.

Proof. Each block introduces at most two new atoms beyond the separators and distinctness forces maximal domain size. ◀

In particular, the number of distinct atoms is linearly bounded by the number of appearances of #.

Fix a positive integer C. Let $\boldsymbol{\sigma}$ in L' be in form (12), where $n_i = C4^{i+1}$ for $i = 0, 1, \ldots, k+1$, and let $v = \textsc{Par}(\boldsymbol{\sigma})$. Then,

$$v(\#) = 2k+2, \quad \| \textsc{dom}(v) \| = 2k+4\,.$$

Moreover, for odd indices $v(\tau_{2j+1}) = n_j$, whereas for even indices $v(\tau_{2j}) = n_{j-1} + n_j$. Since n_i grows exponentially, the order of the values of τ_i's is

$$v(\tau_{2k+2}) > v(\tau_{2k+3}) > v(\tau_{2k}) > v(\tau_{2k+1}) > \cdots \tag{13}$$

▶ **Lemma 36.** *Let* $\boldsymbol{\sigma}$ *and* v *be as above. Let* w *be a data vector such that* $w \leq v$, $w(\#) = 0$, $|w| < C$, *and* $v + w \in \textsc{Par}(L)$. *Let* $\boldsymbol{\sigma'} \in L$ *be such that* $\textsc{Par}(\boldsymbol{\sigma'}) = v + w$. *Then* $\boldsymbol{\sigma'}$ *is either*

$$(\tau_1\tau_2)^{m_0}\#\#(\tau_2\tau_3\tau_4)^{m_1}\#\#(\tau_4\tau_5\tau_6)^{m_2}\#\#(\tau_6\tau_7\tau_8)^{m_3}\cdots \\ \#\#(\tau_{2k}\tau_{2k+1}\tau_{2k+2})^{m_k}\#\#(\tau_{2k+2}\tau_{2k+3})^{m_{k+1}}\,.$$

for some integers $m_0, m_1, \ldots, m_{k+1}$ *such that* $m_i \geq n_i$, $i = 0, 1, \ldots, k+1$, *or the reverse of the latter.*

We shall say that $\boldsymbol{\sigma}$ is *commutatively stable* up to C. That is, small perturbations of v must correspond to words with the same block structure as $\boldsymbol{\sigma}$.

Proof. The gaps between values of distinct τ_i exceed $4C$, therefore adding w preserves their relative order. That is, if $v(a) \leq v(b)$, then also $(v+w)(a) \leq (v+w)(b)$.

Since $(v+w)(\#) = v(\#)$, $\boldsymbol{\sigma'}$ has the same number of blocks as $\boldsymbol{\sigma}$, therefore it is of the form,

$$(\omega_1\omega_2)^{m_0}\#\#(\omega_2\omega_3\omega_4)^{m_1}\#\#(\omega_4\omega_5\omega_6)^{m_2}\#\#(\omega_6\omega_7\omega_8)^{m_3}\cdots \\ \#\#(\omega_{2k}\omega_{2k+1}\omega_{2k+2})^{m_k}\#\#(\omega_{2k+2}\omega_{2k+3})^{m_{k+1}}\,.$$

for a sequence $\omega_1, \omega_2, \ldots, \omega_{2k+3} \in \textsc{Atoms} \setminus \{\#\}$.

Since $w \leq v$, $\textsc{dom}(w) \subseteq \textsc{dom}(v)$ and $\textsc{dom}(v+w) = \textsc{dom}(v)$. Thus, the atoms ω_i are a permutation of τ_j.

We show that the heaviest symbols must remain at their positions, from which we conclude that all the symbols remain in their positions, modulo reversal.

We contend that τ_{2k+2} (the heaviest atom) is either ω_2 or ω_{2k+2}. First, notice it must be in an even position, because it has the largest value in v as well as in $v+w$. Assume to the contrary, that it is placed somewhere in the middle: $\tau_{2k+2} = \omega_{2p}$. Then

$$\begin{aligned} m_{p-1} + m_p &= (v+w)(\omega_{2p}) = (v+w)(\tau_{2k+2}) \\ &\geq v(\tau_{2k+2}) = C4^{k+2} + C4^{k+1} = 20C4^k\,. \end{aligned}$$

Thus, one of m_{p-1} or m_p must be greater than $10C4^K$, implying that two more atoms have values greater than $10C4^k$. However, this is a contradiction, because only τ_{2k+3} has a value greater than $10C4^k$ and by (13) the next largest value is at τ_{2k}:

$$(v+w)(\tau_{2k}) \leq C4^{k+1} + C4^k + C \leq C(5 \cdot 4^k + 1)\,.$$

We may assume that $\tau_{2k+2} = \omega_{2k+2}$, because the case of $\tau_{2k+2} = \omega_2$ results in the reversal of the word.

In this case, $\tau_{2k+3} = \omega_{2k+3}$ and

$$m_{k+1} = (v+w)(\tau_{2k+3}) = C4^{k+2} + w(\tau_{2k+3}) \leq C(4^{k+2}+1)\,.$$

Thus, $m_k = (v+w)(\tau_{2k+2}) - m_{k+1} \geq C(4^{k+1}-1)$. The only possible atoms in this block are τ_{2k}, τ_{2k+1} because the next largest value is that of τ_{2k-2}, but its value is at most $C4^k + C4^{k-1} + C$, the latter is less than m_k. Therefore, by (13), necessarily, $\omega_{2k} = \tau_{2k}$, $\omega_{2k+1} = \tau_{2k+1}$, and $m_k = (v+w)(\tau_{2k+1}) \leq C(4^{k+1}+1)$. In the same manner, we conclude that $\omega_i = \tau_i$ for all i. ◀

5.5 Proof of Theorem 34

Assume to the contrary that $\textsc{Par}(L)$ is rational. Therefore, there is a rational expression R defining $\textsc{Par}(L)$. By Proposition 25, we may assume R is presented in linear form and let r be the star-height of R.

Let N_0, M_0 be the constants provided by Proposition 30 for R. Let $n = 2M_0 + 2$, $k = (r+1)N_0 n^{r+1}$, and let $C = (2k+3)N_0$. Let $\boldsymbol{\sigma}$ be in the form of (12) with k blocks and exponents $n_i = C \cdot 4^{i+1}$. Finally, let $v = \textsc{Par}(\boldsymbol{\sigma})$, and let T_v be a parsing tree of v with respect to R.

We define the neighbor set $N(\tau_p)$ of τ_p by $N(\tau_p) = \{\tau_{p-1}, \tau_{p+1}\}$ for $p = 2, 3, \ldots, 2k+2$, $N(\tau_1) = \{\tau_2\}$, and $N(\tau_{2k+3}) = \{\tau_{2k+2}\}$.

The claim shows that odd indices propagate their value to their neighbors.

▷ Claim 37. If $w \leq v$ is a data vector that appears in the label of a vertex in T_v with $\tau_{2j+1} \in \textsc{dom}(w)$ and $\# \notin \textsc{dom}(w)$, then $N(\tau_{2j+1}) \subseteq \textsc{dom}(w)$.

Proof. We duplicate the sub-tree growing from the vertex containing w and then, from the new copy we prune all descendants of the copied vertex. Consequently, the vertex containing w is added solely, resulting in the data vector $v + w$. By Lemma 28, we obtain a parsing tree of $v + w \in R$. That is, there is $\boldsymbol{\sigma'} \in L$ such that $\textsc{Par}(\boldsymbol{\sigma'}) = v + w$. From commutative stability of $\boldsymbol{\sigma}$, we know that $\boldsymbol{\sigma'}$ has the same blocks as $\boldsymbol{\sigma}$. Since the value of τ_{2j+1} increased, the exponent m_j increased as well, implying $m_j > n_j$. Moreover, the values of τ_{2j-1} and τ_{2j+3} either increased or did not change, implying, $m_{j-1} \geq n_{j-1}$ and $m_{j+1} \geq n_{j+1}$. In particular, $(v+w)(\tau_{2j}) = m_{j-1} + m_j > n_{j-1} + n_j = v(\tau_{2j})$ and $(v+w)(\tau_{2j+2}) = m_j + m_{j+1} > n_j + n_{j+1} = v(\tau_{2j+2})$. Therefore, $\tau_{2j}, \tau_{2j+2} \in \textsc{dom}(w)$. ◁

In the following we show that saturated indices also propagate their value to their neighbors.

▷ Claim 38. Let T be a sub-tree of T_v and let w be the value of T. If an atom a is saturated in $w \leq v$ and $\# \notin \textsc{dom}(w)$, then $N(a) \subseteq \textsc{dom}(w)$.

Proof. For odd indices of $a = \tau_{2j+1}$, this is the previous claim.

So assume $a = \tau_{2j}$. Since $v(\tau_{2j+1}) = (2k+3)4^j N_0 > v(\#)N_0$, T_v contains a vertex that is labeled with a data vector w_j such that $w_j(\#) = 0, w_j(\tau_{2j+1}) > 0$. In particular, by the previous claim, also $\tau_{2j} \in N(\tau_{2j+1}) \subseteq \text{DOM}(w_j)$.

Since τ_{2j} is saturated in T and contributes to T, the sub-tree of w_j is contained in T as well. Implying, $w_j \leq w$, thus $\tau_{2j+1} \in \text{DOM}(w)$.

By the same argument for τ_{2j-1}, we conclude $N(\tau_{2j}) \subseteq \text{DOM}(w)$. ◁

We contend that there is some parent vertex in T_v with at least n immediate descendants with their sub-trees containing a non-zero value of #. Assume to the contrary that is not so, then each non-leaf vertex would have at most n immediate descendants with a non-zero value of #. The total contribution to the value of # would be bounded by $N_0(1+n+n^2+\cdots+n^{r+1})$, which is impossible, because $v(\#) = 2k+2 > k = (r+1)N_0 n^{r+1}$.

Let the values of the sub-trees growing from the above mentioned descendants be $v_1, v_2, \ldots, v_n$ and let R' be the star sub-expression that generates them. In particular, $v_i(\#) > 0$ for $i = 1, 2, \ldots, n$. It follows from the definition of M_0 that $\| \text{SUPP}(R') \| \leq M_0$.

For each $j = 1, 2, \ldots, n$, define

$$A_j = \{a \in \text{DOM}(v_j) : v_j(a) = v(a)\},$$
$$B_j = \text{DOM}(v_j) \setminus \text{SUPP}(R').$$

▷ Claim 39. The sets A_j are pairwise disjoint, and $\| A_j \| \geq v_j(\#)$.

Proof. The sets are disjoint because the atoms in each set are saturated. If $\| A_j \| < v_j(\#)$, subtracting v_j (by pruning its sub-tree) results in a data vector v' with domain larger than the number of appearances of #, which is impossible. ◁

▷ Claim 40. For all j, $\| B_j \| \leq v_j(\#)$

Proof. If $\| B_j \| > v_j(\#)$, renaming the atoms in $\text{DOM}(v_j)$ outside the support would increase the domain of v beyond the number of appearances of #. ◁

If A_j and $\text{SUPP}(R')$ are disjoint, then, $B_j = A_j$. Note that, there are at least $n - M_0 = M_0 + 2$ such j's.

For each such j, let t_j be the largest index of a τ in A_j. Since τ_{t_j} is saturated, by Claim 38 $\tau_{t_j+1} \in \text{DOM}(v_j)$. Thus, $\tau_{t_j+1} \in \text{SUPP}(R')$.

Distinct j's yield distinct t_j's. Therefore, $\text{SUPP}(R')$ contains at least $M_0 + 1$ elements, contradicting $\| \text{SUPP}(R') \| \leq M_0$, which concludes the proof of Theorem 34.

6 A motivating result for star-height

In previous work, [15], rationality of Parikh images for one-register languages was obtained via a sequence of reductions, the most technical of which concerns the language of *anti-paths*. The proof relies on a graph-theoretic characterization of Parikh images and invokes a necessary condition for the existence of Hamiltonian cycles in directed graphs. While this establishes rationality, it yields a very large upper bound on the star-height.

Our bound of 2 crucially exploits the fact that the language of anti-paths actually has a Parikh image of star-height 1; i.e., it is a semi-linear set.

The goal of this section is to establish this auxiliary result for anti-paths. Once this is shown, extending the argument to all one-register languages and to one-register block finite-memory automata requires only technical adaptations.

For the proof of semi-linearity of anti-paths, we use the novel notion of *restrained sets* vs. graphs with Hamiltonian cycles employed in [15].

6.1 Restrained sets

Let

$$\Theta = \textsc{Atoms}^{2\neq} = \{(a,b) : a,b \in \textsc{Atoms}, a \neq b\}\,. \tag{14}$$

▶ **Definition 41.** *Let $A \subseteq \Theta$. We say that A is restrained by $(z,w) \in \Theta$, if for every $(a,b) \in A$, we have $a = w$ or $z = b$. If A is not restrained by any element of Θ, we call A unrestrained.*

▶ Remark 42. Every subset of a restrained set is also restrained by the same pair, i.e., this property is downward-closed. In particular, being unrestrained is upward-closed.

The following lemma shows that unrestrained sets are robust under a simple augmentation operation.

▶ **Lemma 43.** *Let $A \subseteq \Theta$ be an unrestrained set, and let $(c,d) \in \Theta$. Then there is an element $(a,b) \in A$ such that*

- *$a \neq d$, $c \neq b$, equivalently, $(a,d),(c,b) \in \Theta$.*
- *The set*

$$A' = A \setminus \{(a,b)\} \cup \{(a,d),(c,b)\} \subseteq \Theta\,, \tag{15}$$

  is unrestrained as well.
- *The set $A'' = A' \cup \{(a,b)\} \subseteq \Theta$ is also unrestrained.*

Note the last item holds automatically, because $A \subseteq A''$ and from Remark 42, it is unrestrained.

Proof. If there is a pair in A with first component c, then this pair satisfies the claim. Indeed, let $(c,b) \in A$ be such a pair. In this case, we put $a = c$. Then $c \neq b$, because $(c,b) \in A \subseteq \Theta$ and $a = c \neq d$, because $(c,d) \in \Theta$. Moreover, the replacement in (15) just adds the element (c,d) to A. Thus, $A' = A \cup \{(c,d)\}$ includes A which is unrestrained. Therefore, by Remark 42, A' is unrestrained as well.

So assume that c is not the first component of any pair in A. Since A is unrestrained, in particular, it is not restrained by the pair $(c,d) \in \Theta$. Hence, there is a pair $(a,b) \in A$ that is not restrained by (c,d), i.e., $a \neq d$ and $c \neq b$.

We contend that A' is unrestrained. Assume to the contrary, that A' is restrained by some $(z,w) \in \Theta$. Since A is unrestrained, in particular, it is not restrained by (z,w). Thus, the only element of A not restrained by (z,w) is (a,b). Therefore, $a \neq w$ and $z \neq b$. However, (z,w) does restrain the new elements (a,d) and (c,b), implying, $z = d$ and $c = w$.

Consider the set $A_2 = A \setminus \{(a,b)\}$. By our assumption that A' is restrained by (z,w), every element $(u,v) \in A_2 \subseteq A'$ is restrained by (z,w) as well. That is, $u = w = c$ or $z = v = d$. Since we assumed that c is not the first component of any pair in A, this simplifies to $z = v$.

Consider now the pair (z,a). Since $z = d$ and $a \neq d$, we have $(z,a) \in \Theta$. Moreover, (z,a) restrains all elements of A_2 and also restrains (a,b). Thus, (z,a) restrains A, contradicting the assumption that A is unrestrained, and the contention follows. ◀

▶ **Lemma 44.** *Let $A \subseteq \Theta$ be an unrestrained set. Then there exists an unrestrained subset A' of A such that $\| A' \| \leq 4$.*

Proof. We contend that there are two elements (a_1, b_1) and (a_2, b_2) of A such that $a_1 \neq a_2$ and $b_1 \neq b_2$. Otherwise, for each two pairs (a_1, b_1) and (a_2, b_2) in A, $a_1 = a_2$ or $b_1 = b_2$. Let (a, b) be a pair in A. Then, the pair (b, a) restrains A, because for every pair $(c, d) \in A$ either $c = a$ or $d = b$.[4] This is impossible because A is unrestrained.

The only two options to restrain both elements (a_1, b_1) and (a_2, b_2) is either by (b_2, a_1) or by (b_1, a_2).

Since A is unrestrained, there are (a_3, b_3) and (a_4, b_4) in A which are not restrained by these pairs, accordingly.

Therefore the set $A' = \{(a_i, b_i) : i = 1, 2, 3, 4\}$ is unrestrained. ◀

6.2 Matching

The following combinatorial argument ensures that any sequence of four pairs from Θ contains a *matching*, i.e., two pairs (a, b) and (c, d) such that the swapped pairs (a, d) and (c, b) are in Θ.

▶ **Lemma 45.** *Let $(a_i, b_i)_{i=1}^4$ be a sequence of pairs from Θ. Then then there are indices $1 \leq i < j \leq 4$, such that $(a_i, b_j) \in \Theta$ and $(a_j, b_i) \in \Theta$.*

Proof. Assume to the contrary, that for all distinct i and j, it holds that $a_i = b_j$ or $a_j = b_i$. Initialize counters $x_i = 0$ for $i = 1, 2, 3$, and 4. For each pair $i < j$, increase x_j by one if $a_i = b_j$, and increase x_i by one if $a_j = b_i$.

Each pair contributes at least one increment, and there are six such pairs, implying

$$\sum_{i=1}^{4} x_i \geq \binom{4}{2} = 6.$$

From pigeon-hole principle, for some index t

$$x_t \geq \frac{6}{4} > 1\,.$$

Hence, the atom b_t is equal to both a_r and a_s for two distinct indices r and s, implying, $a_r = a_s$. In this case, $a_r = a_s \neq b_s$ and $a_s = a_r \neq b_r$, contradicting our assumption. ◀

6.3 Anti-paths

Let $\Pi = \textsc{Atoms} \times \textsc{Atoms}$, a word $\boldsymbol{\sigma}$ over Π is an *anti-path* if it is of the form;

$$(b_0, a_1)\ (b_1, a_2)\ \cdots\ (b_n, a_{n+1}), \tag{16}$$

and it satisfies $a_i \neq b_i$ for $i = 1, 2, \ldots, n$. Let P be the language of all anti-paths of the form (16) for $n \geq 0$.

Unrestrained anti-paths

Each anti-path $\boldsymbol{\sigma}$ of the form (16) induces a subset $A_{\boldsymbol{\sigma}}$ of Θ (14),

$$A_{\boldsymbol{\sigma}} = \{(a_i, b_i) : i = 1, 2, \ldots, n\}\,. \tag{17}$$

[4] That is, (a, b) is (a_1, b_1) and (c, d) is (a_2, b_2).

An anti-path $\boldsymbol{\sigma}$ is said to be *unrestrained* if $A_{\boldsymbol{\sigma}}$ is unrestrained. The subset $P^{ur} \subseteq P$ are all the unrestrained anti-paths.

An *anti-cycle* $\boldsymbol{\rho}$ is an anti-path that is a cycle, i.e, it is a word over Π in the form;

$$(d_0, c_1)\ (d_1, c_2)\ \cdots\ (d_m, c_{m+1}), \tag{18}$$

where $c_i \neq d_i$ for $i = 1, 2, \ldots, m$ and $c_{m+1} \neq d_0$.

Let $C \subseteq P$ be the set of all anti-cycles.

For a constant N, let P_N, P_N^{ur}, C_N be the sets of all anti-paths, unrestrained anti-paths, and anti-cycles of lengths at most N.

▶ **Lemma 46.** *The language of unrestrained anti-paths has a semi-linear Parikh image. In fact, there are constants N_0, N_1 such that,*

$$\textsc{Par}\left(P^{ur}\right) = \textsc{Par}\left(P_{N_0}^{ur}\right) + \left(\textsc{Par}\left(C_{N_1}\right)\right)^* . \tag{19}$$

Proof. We shall show that the lemma holds for $N_0 = 29$ and $N_1 = 4$.

Let $\boldsymbol{\sigma} \in P^{ur}$ be an unrestrained anti-path (16) and let $\boldsymbol{\rho} \in C$ be an anti-cycle (18). Since $\boldsymbol{\rho}$ is an anti-cycle, $(c_{m+1}, d_0) \in \Theta$.

By Lemma 43, for the unrestrained set $A_{\boldsymbol{\sigma}}$ and the pair (c_{m+1}, d_0), there is a pair $(a_i, b_i) \in A_{\boldsymbol{\sigma}}$ such that $a_i \neq d_0$ and $c_{m+1} \neq b_i$.

Let $\boldsymbol{\omega}$ result in the insertion of anti-cycle $\boldsymbol{\rho}$ into anti-path $\boldsymbol{\sigma}$ immediately after (b_{i-1}, a_i).

That is, for the decomposition $\boldsymbol{\sigma} = \boldsymbol{\sigma}_1(b_{i-1}, a_i)(b_i, a_{i+1})\boldsymbol{\sigma}_2$,

$$\boldsymbol{\omega} = \boldsymbol{\sigma}_1(b_{i-1}, a_i)(d_0, c_1)(d_1, c_2)\cdots\ (d_m, c_{m+1})(b_i, a_{i+1})\boldsymbol{\sigma}_2 .$$

Note that, $\boldsymbol{\omega}$ is an anti-path.

In particular, $A_{\boldsymbol{\omega}}$ is either

$$A_{\boldsymbol{\omega}} = A_{\boldsymbol{\sigma}} \cup \{(a_i, d_0), (c_{m+1}, b_i)\} \cup A_{\boldsymbol{\rho}} \setminus \{(a_i, b_i)\} ,$$

or

$$A_{\boldsymbol{\omega}} = A_{\boldsymbol{\sigma}} \cup \{(a_i, d_0), (c_{m+1}, b_i)\} \cup A_{\boldsymbol{\rho}} ,$$

cf. Lemma 43. In either case $A_{\boldsymbol{\omega}}$ is unrestrained. Thus, $\boldsymbol{\omega} \in P^{ur}$, implying the inclusion

$$\textsc{Par}\left(P^{ur}\right) \supseteq \textsc{Par}\left(P^{ur}\right) + \textsc{Par}\left(C\right) .$$

In particular,

$$\textsc{Par}\left(P^{ur}\right) \supseteq \textsc{Par}\left(P_{N_0}^{ur}\right) + \left(\textsc{Par}\left(C_{N_1}\right)\right)^* ,$$

cf. (19).

For the converse inclusion $\subseteq$, let $\boldsymbol{\sigma} \in P^{ur}$ be an unrestrained anti-path of length at least N_0. By Lemma 44, $A_{\boldsymbol{\sigma}}$ contains an unrestrained subset $A' \subseteq A_{\boldsymbol{\sigma}}$ of size at most 4. Since every pair in $A_{\boldsymbol{\sigma}}$ is related to two letters in $\boldsymbol{\sigma}$,[5] there are at most four consecutive pairs of such letters and we mark these letters. This gives us 5 sub-words of $\boldsymbol{\sigma}$ not containing the marked letters. Since $\frac{N_0 - 8}{5} > 4$, there is a sub-word

$$(f_0, e_1)(f_1, e_2)\cdots(f_4, e_5) ,$$

of $\boldsymbol{\sigma}$ not containing a marked letter.

[5] Recall that $\boldsymbol{\sigma}$ is a word over $\Pi = \textsc{Atoms}^2$.

This sequence $(e_i, f_i)_{i=1}^4$ consists of elements in Θ, implying by Lemma 45, that there are $1 \leq i < j \leq 4$ such that $e_i \neq f_j$ and $e_j \neq f_i$. Therefore, deleting the inner sub-word $(f_i, e_{i+1}) \cdots (f_{j-1}, e_j)$ from $\boldsymbol{\sigma}$ results in an anti-path $\boldsymbol{\omega}$. Moreover, the deleted sub-word is an anti-cycle of length at most four. Note that $\boldsymbol{\omega}$ is an anti-path containing all marked letters. Thus, $A_{\boldsymbol{\omega}}$ contains A' and, therefore, is unrestrained, implying the desired converse inclusion $\subseteq$ of (19). ◀

Restrained anti-paths

If $A \subseteq \Theta$ is restrained by $(z, w) \in \Theta$, there are three options:

1. $A \subseteq \{w\} \times \textsc{Atoms}$; or
2. $A \subseteq \textsc{Atoms} \times \{z\}$; or
3. $A \subseteq \{w\} \times \textsc{Atoms} \cup \textsc{Atoms} \times \{z\}$ and for some $x, y \in \textsc{Atoms}$, $(w, x), (y, z) \in A$.

For fixed distinct atoms $w \neq z$ and $i = 1, 2, 3$, let $P^{i,z,w}$ be the set of all anti-paths $\boldsymbol{\sigma} \in P$, where $A_{\boldsymbol{\sigma}}$ restrained by (z, w) of type (i). Let $P^{z,w} = \bigcup_{i=1}^3 P^{i,z,w}$. It is clear that,

$$P^{1,z,w} = \{(c, w) : c \in \textsc{Atoms}\} \cdot \{(c, w) : w \neq c \in \textsc{Atoms}\}^* \cdot \{(c, d) : c, d \in \textsc{Atoms}, c \neq w\}\,.$$

Therefore,

$$\textsc{Par}\left(P^{1,z,w}\right) = \left(\bigcup_{c \in \textsc{Atoms}} (c, w)\right) + \left(\bigcup_{c \in \textsc{Atoms}\setminus\{w\}, d \in \textsc{Atoms}} (c, d)\right) + \left(\bigcup_{c \in \textsc{Atoms}\setminus\{w\}} (c, w)\right)^*,$$

is a linear set. A symmetric argument shows that $\textsc{Par}\left(P^{2,z,w}\right)$ is linear as well.

Let $C^{z,w}$ be the set of all anti-cycles $\boldsymbol{\rho} \in C$ (18) such that $A_{\boldsymbol{\rho}} \cup \{(c_{m+1}, d_0)\}$ is restrained by (z, w). Let $P_N^{3,z,w}$ and $C_N^{z,w}$ be the set of anti-paths in $P^{3,z,w}$ and anti-cycles in $C^{z,w}$ of length at most N, respectively.

▶ **Lemma 47.** *The language $P^{3,z,w}$ has a semi-linear Parikh image. In fact, there are constants N_0, N_1 such that,*

$$\textsc{Par}\left(P^{3,z,w}\right) = \textsc{Par}\left(P_{N_0}^{3,z,w}\right) + \left(\textsc{Par}\left(C_{N_1}^{z,w}\right)\right)^*\,. \tag{20}$$

Proof. We shall show that the lemma holds for $N_0 = 24, N_1 = 8$. First, we prove that any anti-cycle $\boldsymbol{\rho} \in C^{z,w}$ can be inserted into any anti-path in $P^{3,z,w}$ such that the result of the insertion remains an anti-path in $P^{3,z,w}$. Let $\boldsymbol{\sigma} \in P^{3,z,w}$ and let $\boldsymbol{\rho} \in C^{z,w}$. For some x and y, $\boldsymbol{\sigma}$ contains the letters (x, w) and (z, y). Since $A_{\boldsymbol{\rho}} \cup \{(c_{m+1}, d_0)\}$ is restrained by (z, w), either $c_{m+1} = w$ or $d_0 = z$. If $c_{m+1} = w \neq d_0$, then $\boldsymbol{\rho}$ can be inserted immediately after (x, w). If $d_0 = z \neq c_{m+1}$, then $\boldsymbol{\rho}$ can be inserted immediately before (z, y). That is we have the inclusion

$$\textsc{Par}\left(P^{3,z,w}\right) \supseteq \textsc{Par}\left(P^{3,z,w}\right) + \textsc{Par}\left(C_{N_1}^{z,w}\right)\,.$$

In particular,

$$\textsc{Par}\left(P^{3,z,w}\right) \supseteq \textsc{Par}\left(P_{N_0}^{3,z,w}\right) + \left(\textsc{Par}\left(C_{N_1}^{z,w}\right)\right)^*\,,$$

cf. (20).

For the converse inclusion $\subseteq$, let $\boldsymbol{\sigma} \in P^{3,z,w}$ of length at least than N_0. From definition of $P^{3,z,w}$, there are x and y, such that $\boldsymbol{\sigma}$ contains (x, w) and (z, y). In $\boldsymbol{\sigma}$ we mark one letter with the second component w and one letter with the first component z. This gives us 3 sub-words of $\boldsymbol{\sigma}$ not containing the marked letters. Since $\frac{N_0 - 2}{3} > 7$, there is a sub-word

$$(f_0, e_1)(f_1, e_2) \cdots (f_7, e_8)\,.$$

of $\boldsymbol{\sigma}$ not containing a marked letter.

This sequence $(e_i, f_i)_{i=1}^{7}$ consists of elements in Θ. These elements are also elements of $A_{\boldsymbol{\sigma}}$, thus, they are restrained by (z, w). Therefore, either there are four elements with $e_i = w$ or four elements with $f_i = z$. Without loss of generality, assume the former. By Lemma 45 there are $1 \leq i < j \leq 7$ such that $w = e_i \neq f_j$ and $w = e_j \neq f_i$.

Therefore, deleting the inner sub-word $(f_i, e_{i+1}) \cdots (f_{j-1}, e_j)$ from $\boldsymbol{\sigma}$ results in an anti-path $\boldsymbol{\omega}$. Moreover, the deleted sub-word is an anti-cycle of length at most four. Note that $\boldsymbol{\omega}$ is an anti-path containing all marked letters. Thus, $A_{\boldsymbol{\omega}}$ contains the marked letters and $A_{\boldsymbol{\omega}}$ is restrained by (z, w). Consequently, $\boldsymbol{\omega} \in P^{3,z,w}$, implying the desired converse inclusion $\subseteq$ of (20). ◀

Every restrained anti-path is restrained by some $(z, w) \in \Theta$, and therefore lies in some $P^{z,w}$. Taking the union of all distinct atom pairs $(z, w) \in \textsc{Atoms}^{2\neq}$ yields the following lemma.

▶ **Lemma 48.** *The language of restrained anti-paths has a semi-linear Parikh image.*

▶ **Lemma 49.** *The language of anti-paths has a semi-linear Parikh image.*

Proof. The proof follows from Lemmas 46 and 48. ◀

7 Concluding remarks

Several natural questions remain open. Most notably, the case of *two registers* remains unresolved, both for finite-memory automata and for context-free grammars. At present, no counterexample is known for a commutatively stable language generated by either model. It is therefore unclear whether these models always admit rational Parikh images, or whether they generate irrational images that nevertheless coincide. More generally, given a commutatively stable language, the techniques developed in this paper appear robust enough to establish irrationality and Parikh in-equivalence between grammars and automata, should suitable counterexamples exist.

We mention several decision problems. For one-register automata, star-height of zero of the Parikh image is easily decidable, as it coincides with boundness. This raises the question of whether it is decidable if the Parikh image has star-height of one. A positive answer would yield a complete decision procedure for determining the exact star-height of Parikh images of one-register automata.

References

1 Mikołaj Bojanczyk. Slightly infinite sets, 2019. A draft of a book available at `https://www.mimuw.edu.pl/~bojan/paper/atom-book`.

2 Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata with group actions. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 355–364. IEEE, IEEE Computer Society, 2011. `doi:10.1109/LICS.2011.48`.

3 Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3):1–44, 2014. `doi:10.2168/LMCS-10(3:4)2014`.

4 Taolue Chen, Fu Song, and Zhilin Wu. Formal reasoning on infinite data values: An ongoing quest. In Jonathan P. Bowen, Zhiming Liu, and Zili Zhang, editors, *Engineering Trustworthy Software Systems - Second International School, SETSS 2016, Chongqing, China, March 28 - April 2, 2016, Tutorial Lectures*, volume 10215 of *Lecture Notes in Computer Science*, pages 195–257, 2016. `doi:10.1007/978-3-319-56841-6_6`.

5 Edward Y. C. Cheng and Michael Kaminski. Context-free languages over infinite alphabets. *Acta Informatica*, 35(3):245–267, 1998. doi:10.1007/s002360050120.

6 Lorenzo Clemente, Slawomir Lasota, and Radoslaw Piórkowski. Determinisability of register and timed automata. *Log. Methods Comput. Sci.*, 18(2):Paper No. 9, 37, 2022. doi:10.46298/lmcs-18(2:9)2022.

7 Yoav Danieli. Star complexity of parikh images of languages over infinite alphabets, 2026. arXiv:2605.09435.

8 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.

9 Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: https://www.worldcat.org/oclc/310535248.

10 Samuel Eilenberg and Marcel Paul Schützenberger. Rational sets in commutative monoids. *J. Algebra*, 13(2):173–191, 1969.

11 Diego Figueira and Anthony Widjaja Lin. Reasoning on data words over numeric domains. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 37:1–37:13. ACM, 2022. doi:10.1145/3531130.3533354.

12 Florian Frank, Daniel Hausmann, Stefan Milius, Lutz Schröder, and Henning Urbat. Alternating nominal automata with name allocation. In *40th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2025, Singapore, June 23-26, 2025*, pages 57–70. IEEE, 2025. doi:10.1109/LICS65433.2025.00012.

13 Daniel Genkin, Michael Kaminski, and Liat Peterfreund. A note on the emptiness problem for alternating finite-memory automata. *Theor. Comput. Sci.*, 526:97–107, 2014. doi:10.1016/j.tcs.2014.01.020.

14 Matthew Hague, Artur Jez, and Anthony W. Lin. Parikh's theorem made symbolic. *Proc. ACM Program. Lang.*, 8(POPL):1945–1977, 2024. doi:10.1145/3632907.

15 Piotr Hofman, Marta Juzepczuk, Slawomir Lasota, and Mohnish Pattathurajan. Parikh's theorem for infinite alphabets. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470626.

16 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

17 Michael Kaminski and Daniel Zeitlin. Finite-memory automata with non-deterministic reassignment. *Int. J. Found. Comput. Sci.*, 21(5):741–760, 2010. doi:10.1142/S0129054110007532.

18 Ahmet Kara. *Logics on data words: Expressivity, satisfiability, model checking*. PhD thesis, Technical University of Dortmund, Germany, 2016. URL: https://hdl.handle.net/2003/35216.

19 Bartek Klin, Slawomir Lasota, and Szymon Torunczyk. Nondeterministic and co-nondeterministic implies deterministic, for data languages. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 365–384. Springer, 2021. doi:10.1007/978-3-030-71995-1_19.

20 Slawomir Lasota and Mohnish Pattathurajan. Parikh images of register automata. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, Virtual Conference, December 15-17, 2021*, volume 213 of *LIPIcs*, pages 50:1–50:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.50.

21 Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 1981. URL: https://www.worldcat.org/oclc/299372519.

22 Andrzej S. Murawski, Steven J. Ramsay, and Nikos Tzevelekos. Bisimilarity in fresh-register automata. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 156–167. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.24.

23 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

24 Rohit Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966. doi:10.1145/321356.321364.

25 Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959. doi:10.1147/rd.32.0114.

26 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006. doi:10.1007/11874683_3.