Parikh Automata with Pushdown Stack

von
Wong Karianto
Matrikelnummer 216778

Diplomarbeit
im Studiengang Informatik

vorgelegt der Fakultät für
Mathematik, Informatik und Naturwissenschaften der
Rheinisch-Westfälischen Technischen Hochschule Aachen

im Dezember 2004

angefertigt am
Lehrstuhl für Informatik VII
Logik und Theorie Diskreter Systeme
Universitätsprofessor Dr. Wolfgang Thomas

Typeset in Computer Modern and AMSFonts using LaTeX 2ε

Wong Karianto
Franzstr. 111
D-52064 Aachen
Germany
Email: karianto@i7.informatik.rwth-aachen.de

First edition: 14 December 2004
Second impression, with errata: 16 December 2004
Third impression, with errata: 5 April 2005

to my beloved Mother and Father

teruntuk Ibunda dan Ayahanda tercinta

Contents

List of Figures

List of Tables

Acknowledgements

I am very grateful to all the people who have contributed to this thesis both directly and indirectly.

First of all, I would like to thank my supervisor, Prof. Wolfgang Thomas. He has suggested the topic for this thesis, and his guidance has always been of great value to my work on this thesis.

Special thanks go to Prof. Aloys Krieg, who has provided me with an insight into the number-theoretical analysis underlying many results in Chapter 5 (in particular Proposition 5.12 and Theorem 5.20). In fact, many results there were parts of my collaboration with him and Prof. Wolfgang Thomas some time ago.

I would like to thank Prof. Klaus Indermark for his kind readiness to co-examine this thesis.

I am deeply indebted to Tobias Ganzow and Philipp Rohde for their help with proofreading the first draft of this thesis. Their numerous comments and suggestions have been invaluable to this thesis and have helped shape this thesis.

I would like to thank Christof Löding, Michael Ummels, and Stefan Wöhrle for some helpful discussions.

I would like to thank Yenny for her help, patience, and encouragement while I was writing this thesis.

Finally, I would like to express my gratitude to my parents, without whose support and love this thesis would not have been possible.

Chapter 1

Introduction

The Parikh mapping, introduced by Parikh [Par66], provides a connection between formal language theory and number theory; it associates a word (that is, a string of symbols) with the vector of natural numbers that reflects the number of the occurrences of the symbols in this word. In this view, the order of the occurrences of symbols does not matter anymore; Parikh's theorem states that the class of context-free languages, with respect to the Parikh images (that is, the images under the Parikh mapping), does not differ from the class of regular languages. Moreover, Parikh succeeded in characterizing the Parikh images of context-free (and thus also of regular) languages, namely in the framework of semi-linear sets.

Semi-linear sets, subsequently, have been extensively studied, in particular by Ginsburg and Spanier [GS64, GS66]. It turned out that the class of semi-linear sets enjoys many good properties, not only with respect to closure properties, but also with respect to decision properties.

The many good properties of semi-linear sets justify their application in solving many decision problems in formal language theory. Most prominently, the decidability of the emptiness problem for a language immediately follows once we have shown that the Parikh image of this language effectively yields a semi-linear set. For example, the decidability of the emptiness problem for context-free languages immediately follows from Parikh's theorem. Another example of such an approach to the emptiness problem for languages involves classes of languages recognized by automata with reversal-bounded counters (see Ibarra [Iba78]).

Semi-linear sets, by themselves of purely number-theoretical nature, are therefore tightly connected with formal language theory. In addition, the notion of semi-linear sets is related to logic; the class of semi-linear sets turned out to coincide with the class of sets definable in Presburger arithmetic (the

first-order theory of the natural numbers with addition), which is known to be decidable. This connection, in fact, confirms the many good properties of semi-linear sets.

Presburger arithmetic and semi-linear sets play an important role in current research. In the field of system verification, on the one hand, many decision problems concerning model-checking infinite-state systems can be reduced to the emptiness problem for semi-linear sets. Examples of such infinite-state systems are discrete-timed pushdown automata (see Dang et al. [DIB+00]), several extensions of reversal-bounded counter machines (see Ibarra et al. [IBS00]), and some classes of semi-linear systems (see Bouajjani and Habermehl [BH96]). Moreover, Presburger arithmetic appears to be a useful tool in the verification of infinite-state systems; some examples of this application are found in the safety analysis of multiple-counter automata (see Comon and Jurski [CJ98]) and the model-checking of discrete-timed automata in terms of parametric-timed branching temporal logic (see Bruyére et al. [BDR03]).

On the other hand, Presburger arithmetic has been taken under consideration in the study of XML documents and XML schemata. Tree automata play an important role in this subject of research; a tree automaton, for instance, can be used as a recognizing device for XML documents. Dal Zilio and Lugiez [DL03] and Seidl et al. [SSM03], independently from each other, developed tree automata whose transition functions are constrained by Presburger arithmetic formulas and showed that many decision problems for these automata, such as the emptiness and the membership problem, remain decidable.

The application of Presburger arithmetic or semi-linear sets, equivalently, in automata theory has also been pursued by Klaedtke and Rueß [KR02, KR03]; they introduced a new model of word acceptors and tree acceptors that they collectively called *Parikh automata*.

Generally, the main idea of a Parikh automaton over words is the following. We use an automaton, a set of vectors of natural numbers, and a mapping that, similar to the Parikh mapping, associates every word with a vector of natural numbers. A word is accepted by the Parikh automaton if, first, this word is accepted by the underlying automaton, and second, the image of this word under the pre-defined mapping belongs to the pre-defined set; in this sense, the latter set is also referred to as the constraint set of the Parikh automaton. Analogously, applying this idea to trees instead of words results in Parikh automata over trees.

In their papers on Parikh automata, Klaedtke and Rueß considered finite word automata and finite tree automata, for the automata component of Parikh automata, and semi-linear sets, for the constraint set, and referred

to the resulting automata as *Parikh finite word automata* and *Parikh finite tree automata*, respectively. They investigated several closure and decision properties of these automata. In particular, they showed that the emptiness problem for these automata is decidable, again, by reducing this problem to the emptiness problem for semi-linear sets. Furthermore, they gave a characterization of these automata in terms of weak monadic second-order logic that is extended by cardinality constraints, thereby extending the classical connection between logic and automata.

Objectives and Outline

Parikh automata over words, specifically, will be our starting point in this thesis. In particular, we are interested in the issue of applying the idea of Parikh automata to other automata models and to other classes of constraint sets while retaining the decidability of the emptiness problem. This issue, in turn, gives rise to the following questions, with which this thesis is concerned:

1. Do we obtain more expressiveness if we use pushdown automata instead of finite automata, while still using semi-linear constraint sets? Is the emptiness problem for the resulting automata still decidable?
2. Is the emptiness problem still decidable if we use automata beyond pushdown automata?
3. Is the emptiness problem still decidable if we use constraint sets beyond semi-linear sets?

In Chapter 2 we fix our notations. Moreover, we present basic facts concerning semi-linear sets, Presburger arithmetic, and Parikh's theorem.

In Chapter 3 we present the core idea of Parikh automata more precisely. We give the definitions of Parikh finite word automata and present the main results of Klaedtke and Rueß regarding the closure and decision properties of these automata. In particular, we will see that the emptiness problem for these automata is decidable.

Chapter 4 is addressed to the first question above. We introduce 'Parikh pushdown automata' formally, show some closure properties, and analyze some decision problems. We will see that, as already noted by Klaedtke and Rueß, the decidability of the emptiness problem for Parikh finite word automata carries over to Parikh pushdown automata. Then, we investigate the relations between the classes of languages recognized by Parikh automata and the classes of languages in the Chomsky hierarchy. In particular, we show that the languages recognized by Parikh pushdown automata are context-sensitive.

In Chapter 5 we consider 'level 2 pushdown automata,' which generalize pushdown automata, as candidates for the automata component of Parikh automata. As an approach to the second question above, we analyze the Parikh images of languages recognized by level 2 pushdown automata. We extend the framework of semi-linear sets to the so-called 'semi-polynomial sets' and show that every semi-polynomial set is the Parikh image of the language recognized by a level 2 pushdown automaton. On the other hand, we also show that there are level 2 pushdown automata that recognize languages whose Parikh images are not semi-polynomial. Moreover, as an approach to the third question above, we study the issue of using semi-polynomial sets as the constraint set of Parikh automata. More precisely, we study the emptiness problem for the intersection of a semi-polynomial and a semi-linear set. We show two partial results regarding this problem; first, weakening the notion of semi-linear sets leads to decidability, and second, strengthening the notion of semi-polynomial sets leads to undecidability.

Contributing to the application of semi-linear sets in current research, in Chapter 6 we apply the idea of Parikh automata to infinite graphs, which are an active subject of research in the field of system verification. This application yields new classes of infinite graphs. We show the decidability of the reachability problem for these new classes, by reducing this problem to the emptiness problem for semi-linear sets. Further, we study the relations between these new classes and other existing classes of infinite graphs.

Chapter 2

Languages, Automata, and Semi-Linear Sets

We recall some basic terminology concerning formal languages, automata, and semi-linear sets. In particular, we fix the notations we are going to use throughout this thesis. Note that this chapter is not intended to be comprehensive. For introductory readings to this subject, the reader is referred to [HU79] and [Har78].

2.1 Languages and Automata

Alphabets, Words, and Languages

An *alphabet* is a finite, nonempty set of *symbols*. For an alphabet Σ, a *word* over Σ is a (possibly empty) finite sequence of symbols taken from Σ. We denote the empty word by ε. The set of all words over Σ is denoted by Σ^*, and the set of all nonempty words is denoted by Σ^+. A *language* over Σ is a subset of Σ^*.

The *concatenation* of words u and v, denoted by uv, is obtained by appending v to u. Further, the concatenation of languages K and L is defined by $KL := \{uv \mid u \in K, v \in L\}$. For a language L and a natural number $i \geq 1$, let L^i be the concatenation of L with itself i-times. The *Kleene closure* of L is defined by $L^* := \{\varepsilon\} \cup \bigcup_{i \geq 1} L^i$. For convenience, we use L^+ as an abbreviation for $\bigcup_{i \geq 1} L^i$.

Regular expressions over an alphabet Σ are built up inductively from $\emptyset$, ε, and all symbols $a \in \Sigma$ by using concatenation, the Kleene closure, and union (denoted by $+$).

The *length* of a word w is denoted by $|w|$. For a word $w \in \Sigma^*$ and a symbol $a \in \Sigma$, we denote by $|w|_a$ the number of occurrences of a in w.

Given two alphabets Σ and Π, a *homomorphism* from Σ^* to Π^* is a mapping $h\colon \Sigma^* \to \Pi^*$ which satisfies

- $h(\varepsilon) = \varepsilon$ and
- $h(uv) = h(u)h(v)$, for each $u, v \in \Sigma^*$.

The homomorphism h is completely specified if $h(a)$ is given, for each $a \in \Sigma$.

Finite Automata

A (*nondeterministic*) *finite automaton* (*FA*) $\mathfrak{A}$ over an alphabet Σ is a system $(Q, \Sigma, \delta, q_o, F)$, where Q is a finite, nonempty set of states, $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function ($\mathcal{P}(Q)$ denotes the power set of Q), $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. A *deterministic finite automaton* (*DFA*) over an alphabet Σ is an FA $\mathfrak{A}$ over Σ where $|\delta(q, a)| \leq 1$, for each $q \in Q$ and $a \in \Sigma$.

A *run* ρ of $\mathfrak{A}$ from state p to state q via word $w = w_1 \cdots w_n$ is a sequence $p_0, \ldots, p_n$ of states such that $p = p_0$, $q = p_n$, and $p_i \in \delta(p_{i-1}, w_i)$, for $i = 1, \ldots, n$. We write $\mathfrak{A}\colon p \xrightarrow{w} q$ if such a run exists. Given a word w, a run of $\mathfrak{A}$ on w is a run of $\mathfrak{A}$ from the initial state q_0 to a state q via w. Such a run is called *accepting* if $q \in F$. The FA $\mathfrak{A}$ *accepts* w if there is an accepting run of $\mathfrak{A}$ on w. The language *recognized* by $\mathfrak{A}$ is denoted by $L(\mathfrak{A}) := \{w \in \Sigma^* \mid \mathfrak{A} \text{ accepts } w\}$.

The class of languages recognized by FA's (or DFA's, equivalently) over Σ is denoted by $\mathcal{L}_{\mathrm{FA}}(\Sigma)$ or $\mathcal{L}_{\mathrm{FA}}$, whenever Σ is clear from the context.

Pushdown Automata

A (*nondeterministic*) *pushdown automaton* (*PDA*) $\mathfrak{A}$ over an alphabet Σ is a system $(Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$, where Q is a finite, nonempty set of states, Γ is the stack alphabet, $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ is the transition function, q_0 is the initial state, $\bot$ is the initial stack symbol, and $F \subseteq Q$ is the set of final states. A *deterministic pushdown automaton* (*DPDA*) over an alphabet Σ is a PDA $\mathfrak{A}$ over Σ where $|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$, for each $q \in Q$, $a \in \Sigma$, and $Z \in \Gamma$.

A *configuration* of $\mathfrak{A}$ is a pair (q, α), where q is a state in Q, and α is a word over Γ. In such a configuration, the word α represents the current stack content, where the *leftmost* symbol of α represents the *topmost* symbol of the stack. The initial configuration of $\mathfrak{A}$ is $(q_0, \bot)$. Let $p, q \in Q$, $Z \in \Gamma$, and $\alpha, \beta \in \Gamma^*$. The PDA $\mathfrak{A}$ can reach configuration $(q, \beta\alpha)$ from configuration $(p, Z\alpha)$ by reading an input symbol $a \in \Sigma$ or without reading any input symbol if $\delta(p, a, Z)$ or $\delta(p, \varepsilon, Z)$, respectively, contains (q, β). The PDA $\mathfrak{A}$ *accepts* a

word $w \in \Sigma^*$ if $\mathfrak{A}$ reaches a configuration (q, α), for some final state $q \in F$, from the initial configuration by reading w (acceptance by final state). The language *recognized* by $\mathfrak{A}$ is denoted by $L(\mathfrak{A}) := \{w \in \Sigma^* \mid \mathfrak{A} \text{ accepts } w\}$.

The class of languages recognized by PDA's over Σ is denoted by $\mathcal{L}_{\mathrm{PDA}}(\Sigma)$ or $\mathcal{L}_{\mathrm{PDA}}$, whenever Σ is clear from the context. Analogously, the class of languages recognizable by DPDA's over Σ is denoted by $\mathcal{L}_{\mathrm{DPDA}}(\Sigma)$ or $\mathcal{L}_{\mathrm{DPDA}}$.

2.2 Semi-Linear Sets

In this section we introduce the notion of semi-linear sets in the sense of Parikh [Par66]. In order to define semi-linear sets properly, we first fix some notations concerning vectors of natural numbers and operations on them.

We denote the set of nonnegative integers (natural numbers) by $\mathbb{N}$. For a natural number $n \geq 1$, we denote by $\mathbb{N}^n$ the Cartesian product of $\mathbb{N}$ with itself n-times. The members of $\mathbb{N}^n$ are called *vectors* (of natural numbers) of dimension n. The vector that consists of n zeroes, called the *zero vector* of dimension n, is denoted by 0^n, or simply $\bar{0}$ whenever the dimension of the zero vector is clear from the context. For each $\bar{x} \in \mathbb{N}^n$ and $i = 1, \ldots, n$, we refer to the i-th component of $\bar{x}$ as $(\bar{x})_i$. Further, for each $A \subseteq \mathbb{N}^n$, we define the set

$$(A)_i := \{(\bar{x})_i \mid \bar{x} \in A\}.$$

For $i = 1, \ldots, n$, the *i-th unit vector* in $\mathbb{N}^n$, denoted by $\bar{e}_i$, is the one with the i-th component equal to 1 and all the other components equal to 0.

Let $\bar{x} := (x_1, \ldots, x_n)$ and $\bar{y} := (y_1, \ldots, y_n)$ be vectors of dimension $n \geq 1$, and let k be a natural number. We componentwise define the vectors

$$\bar{x} + \bar{y} := (x_1 + y_1, \ldots, x_n + y_n)$$

and

$$k\bar{x} := (kx_1, \ldots, kx_n).$$

Further, for vector sets $A, B \subseteq \mathbb{N}^n$, we define

$$A + B = \{\bar{x} + \bar{y} \mid \bar{x} \in A \text{ and } \bar{y} \in B\}.$$

Let $\bar{x} := (x_1, \ldots, x_n)$ and $\bar{y} := (y_1, \ldots, y_m)$ be vectors of dimension $n \geq 1$ and $m \geq 1$, respectively. The *concatenation* of $\bar{x}$ and $\bar{y}$ is the vector

$$\bar{x} \otimes \bar{y} := (x_1, \ldots, x_n, y_1, \ldots, y_m)$$

of dimension $n+m$, which is obtained by appending $\bar{y}$ to $\bar{x}$. Further, for vector sets $A \subseteq \mathbb{N}^n$ and $B \subseteq \mathbb{N}^m$, we define

$$A \otimes B := \{\bar{x} \otimes \bar{y} \mid \bar{x} \in A \text{ and } \bar{y} \in B\}\,.$$

For natural numbers $n \geq 2$ and $i = 1, \ldots, n$, we define the (*i-th*) *projection function* $p_i^n \colon \mathbb{N}^n \to \mathbb{N}^{n-1}$ by

$$p_i^n(x_1, \ldots, x_n) := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)\,,$$

for each vector $(x_1, \ldots, x_n) \in \mathbb{N}^n$, resulting in a vector of dimension $n-1$. Analogously, for each $A \subseteq \mathbb{N}^n$, we define

$$p_i^n(A) := \{p_i^n(\bar{x}) \mid \bar{x} \in A\}\,.$$

We are now ready to define linear and semi-linear sets. In general, a linear set is a set of vectors of natural numbers whose elements are 'linearly generated' by some particular vectors, which we then refer to as the generators of this linear set. The following definition formalizes this concept.

Definition 2.1. A set $A \subseteq \mathbb{N}^n$, $n \geq 1$, is said to be *linear* if there are vectors $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_m \in \mathbb{N}^n$, $m \geq 0$, such that

$$A = \{\bar{x}_0 + k_1\bar{x}_1 + \ldots + k_m\bar{x}_m \mid k_1, \ldots, k_m \in \mathbb{N}\}\,.$$

The vector $\bar{x}_0$ is called the *constant vector* while the vectors $\bar{x}_1, \ldots, \bar{x}_m$ are called the *periods*, and all of them together are called the *generators* of A. The set $A \subseteq \mathbb{N}^n$ is said to be *semi-linear* if it is a finite union of linear sets.

Remark 2.2. (a) Without loss of generality, we may assume that all the periods of a linear set are not the zero vector since such periods, obviously, can be eliminated without affecting the linear set under consideration.

(b) Every singleton subset of $\mathbb{N}^n$, $n \geq 1$, is a linear set, namely such without periods.

(c) Every finite subset of $\mathbb{N}^n$, $n \geq 1$, is semi-linear since it is a finite union of (linear) singleton sets.

(d) The empty subset of $\mathbb{N}^n, n \geq 1$, is semi-linear since it is the (finite) union of zero linear sets.

(e) The set $\mathbb{N}^n$, for each $n \geq 1$, is a linear set since

$$\mathbb{N}^n = \{k_1\bar{e}_1 + \cdots + k_n\bar{e}_n \mid k_1, \ldots, k_n \in \mathbb{N}\}\,,$$

where $\bar{e}_i$ is the i-th unit vector in $\mathbb{N}^n$, for $i = 1, \ldots, n$.

Example 2.3. (a) The set $A := \{(x_1, x_2, x_3) \in \mathbb{N}^3 \mid 2x_1 = 3x_2 + 7 \text{ and } x_3 = x_1 + x_2\}$ is linear since it can be defined by

$$\{(5, 1, 6) + k(3, 2, 5) \mid k \in \mathbb{N}\}.$$

(b) The set $B := \{(x_1, x_2, x_3) \in \mathbb{N}^3 \mid x_1 < x_2 < x_3\}$ is linear since it can be defined by

$$\{(0, 1, 2) + k_1(0, 0, 1) + k_2(0, 1, 1) + k_3(1, 1, 1) \mid k_1, k_2, k_3 \in \mathbb{N}\}.$$

The following theorem, due to Ginsburg and Spanier [GS64], summarizes the closure properties of semi-linear sets.

Theorem 2.4 (Ginsburg and Spanier)**.** *The class of semi-linear sets is effectively closed under union, intersection, complementation, concatenation, and the projection functions.*

The following lemma states the closure of the class of semi-linear sets under addition; we will make use of it, particularly, in Chapter 6.

Lemma 2.5. *If $A, B \subseteq \mathbb{N}^n$, $n \geq 1$, are semi-linear, then so is $A + B$.*

Proof. Suppose that $A = \bigcup_{i=1}^{r} A_i$, for some $r \geq 0$, and $B = \bigcup_{j=1}^{s} B_j$, for some $s \geq 0$, where $A_1, \ldots, A_r, B_1, \ldots, B_s$ are linear. Obviously, we have

$$\begin{aligned} A + B &= \bigcup_{i=1}^{r} A_i + \bigcup_{j=1}^{s} B_j \\ &= \bigcup_{i=1}^{r} \bigcup_{j=1}^{s} (A_i + B_j). \end{aligned}$$

Thus, by the fact that the class of semi-linear sets is closed under union, we need only to prove the assertion for linear sets.

Now, suppose A and B are linear, say, with the generators $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_p$ and $\bar{y}_0, \bar{y}_1, \ldots, \bar{y}_q$, respectively, for some $p, q \geq 0$. Then, the set

$$A + B = \{\bar{x}_0 + \bar{y}_0 + k_1\bar{x}_1 + \cdots k_p\bar{x}_p + l_1\bar{y}_1 + \cdots + l_q\bar{y}_q \mid k_1, \ldots, k_p, l_1, \ldots, l_q \in \mathbb{N}\}$$

is, by definition, again a linear set. □

Presburger Sets: a Logical Characterization of Semi-Linear sets

There is a characterization of semi-linear sets in terms of logic, namely in terms of Presburger arithmetic. In the following we briefly recall some definitions and results concerning Presburger arithmetic. In particular, we will mention the connection between Presburger arithmetic and the class of semi-linear sets. We assume that the reader is familiar with the fundamental notions of mathematical logic. For an introduction, the reader is referred to [EFT94].

Definition 2.6. *Presburger arithmetic* is the first-order theory (with equality) of the natural numbers with addition. In other words, it is the set of first-order sentences that are true in the structure $(\mathbb{N}, +)$. We refer to the first-order formulas over this structure as *Presburger formulas*. Similarly, we refer to Presburger formulas without free variables as *Presburger sentences*.

We denote by $\varphi(z_1, \ldots, z_n)$ a Presburger formula with the free variables $z_1, \ldots, z_n$. Such a Presburger formula defines the set

$$[\![\varphi]\!] := \{(x_1, \ldots, x_n) \in \mathbb{N}^n \mid (\mathbb{N}, +) \models \varphi[z_1/x_1, \ldots, z_n/x_n]\}$$

of vectors of natural numbers of dimension n, where $\varphi[z_1/x_1, \ldots, z_n/x_n]$ denotes the Presburger sentence that is obtained by replacing the variable z_i with the natural number x_i, for $i = 1, \ldots, n$. Note that this replacement indeed yields a Presburger sentence since every *fixed* natural number is first-order-definable in the structure $(\mathbb{N}, +)$ (see Remark 2.7 below).

A set A of vectors of dimension $n \geq 1$ is said to be a *Presburger set* if there is a Presburger formula φ with n-many free variables such that $A = [\![\varphi]\!]$.

Remark 2.7. For convenience, we use the following abbreviations when specifying a Presburger formula:

- For any variable z, the formula '$z = 0$' stands for $\forall z_1(z + z_1 = z_1)$.
- For any variables z_1 and z_2, the formulas '$z_1 \leq z_2$' and '$z_1 < z_2$' stand for $\exists z(z_1 + z = z_2)$ and $(z_1 \leq z_2) \wedge \neg(z_1 = z_2)$, respectively.
- For any variable z, the formula '$z = 1$' stands for $\neg(z = 0) \wedge \forall z_1((z_1 = 0) \vee (z \leq z_1))$.
- For any variable z and any natural number $k \geq 2$, the first-order term 'kz' stands for the first-order term $z + \cdots + z$ that consists of k-many z's. The formula '$z = k$' stands for $\exists z_1((z_1 = 1) \wedge (kz_1 = z))$.

Example 2.8. (a) The set A of Example 2.3(a) is definable by the Presburger formula

$$\varphi_A(z_1, z_2, z_3) := (\exists z((z = 7) \wedge (2z_1 = 3z_2 + z))) \wedge (z_3 = z_1 + z_2)\,.$$

(b) The set B of Example 2.3(b) is definable by the Presburger formula

$$\varphi_B(z_1, z_2, z_3) := (z_1 < z_2) \wedge (z_2 < z_3)\,.$$

The main result concerning Presburger arithmetic is that the truth of any Presburger sentence (in the structure $(\mathbb{N}, +)$) is decidable. This result is due to Presburger [Pre30].

Theorem 2.9 (Presburger)**.** *Presburger arithmetic is decidable.*

Actually, Presburger has proven this result over the integers with addition and the ordering relation, that is, the structure $(\mathbb{Z}, +, <)$, and in fact, in the literature the term 'Presburger arithmetic' sometimes refers to the first-order theory of this structure. All the same, as noted by Klaedtke [Kla04, pages 2–3], this difference is not substantial; the decidability of the first-order theory of the natural numbers with addition implies that of the integers with addition and the ordering relation, and vice versa. A direct, automata-based proof of the decidability of Presburger arithmetic over the natural numbers with addition can be found, for instance, in [Büc60] and [Wey02].

By definition, the class of Presburger sets is closed under union, intersection, complementation, and the projection functions, which correspond to the logical connectives $\vee$, $\wedge$, $\neg$, and the existential quantifier $\exists$, respectively. The class of semi-linear sets, on the other hand, possesses these closure properties as well (see Theorem 2.4). Hence, it is not surprising that both classes coincide. This result is due to Ginsburg and Spanier [GS66].

Theorem 2.10 (Ginsburg and Spanier)**.** *The class of semi-linear sets is precisely the class of Presburger sets. Moreover, given a semi-linear set A (by its generators), we can effectively find a Presburger formula that defines A. Conversely, given a Presburger formula φ, we can effectively find the generators of $[\![\varphi]\!]$.*

To be precise, the Presburger formulas used in [GS66] are of a special form; the atomic formulas are *linear equations* of the form

$$k_0 + \sum_{i=1}^{n} k_i z_i = k'_0 + \sum_{i=1}^{n} k'_i z_i \,, \tag{2.1}$$

where $k_0, \ldots, k_n, k'_0, \ldots, k'_n$ are natural numbers and $z_1, \ldots, z_n$ are variables in natural numbers. Again, this difference is not substantial. On the one hand, ordinary atomic first-order formulas in Presburger arithmetic already have this form. On the other hand, formulas of the form (2.1) can easily be translated into ordinary first-order formulas since, as noted in Remark 2.7 above, every fixed natural number is first-order-definable in the structure $(\mathbb{N}, +)$.

As a final remark, the use of Presburger formulas of the form (2.1) also suggests that every semi-linear set is the set of the solutions of a linear equation of this form in natural numbers. This fact is implicitly stated in Ginsburg and Spanier's proof of Theorem 2.10 in [GS66].

Parikh's Theorem

The main interest in semi-linear sets is due to their relation to formal languages. In this context, the bridge between words and vectors is provided by the Parikh mapping.

Definition 2.11. Let $\Sigma = \{a_1, \ldots, a_n\}$, $n \geq 1$, be an alphabet. The *Parikh mapping* $\Phi\colon \Sigma^* \to \mathbb{N}^n$ is defined by

$$\Phi(w) := (|w|_{a_1}, \ldots, |w|_{a_n})\,,$$

for each word $w \in \Sigma^*$. The vector $\Phi(w)$ is called the *Parikh image* of w. Similarly, for a language $L \subseteq \Sigma^*$, the set $\Phi(L) := \{\Phi(w) \mid w \in L\} \subseteq \mathbb{N}^n$ is called the *Parikh image* of L.

Remark 2.12. The Parikh mapping Φ is a monoid homomorphism from Σ^* (with concatenation) to $\mathbb{N}^n$ (with addition) which is defined by

$$\Phi(a_i) := \bar{e}_i\,,$$

where $\bar{e}_i$ is the i-th unit vector in $\mathbb{N}^n$, for $i = 1, \ldots, n$. Thus, we have

- $\Phi(\varepsilon) = 0^n$ and
- $\Phi(uv) = \Phi(u) + \Phi(v)$, for each $u, v \in \Sigma^*$.

An application of the Parikh mapping, which immediately follows from the definition above, lies in solving the emptiness problem for languages; it is not difficult to verify that a language is empty if and only if its Parikh image is empty, which we state in the following remark.

Remark 2.13. For any language L over an alphabet Σ, it holds that

$$L = \emptyset \quad \text{if and only if} \quad \Phi(L) = \emptyset\,.$$

To be precise, in the definition above we need a linear ordering on the alphabet Σ, in order to define the Parikh mapping properly. Hence, in the sequel we will assume that the alphabet under consideration is linearly ordered. Actually, in the definition above we have already defined a linear ordering on the alphabet Σ by enumerating the symbols in Σ as $a_1, \ldots, a_n$.

In the literature the term 'commutative image' is sometimes used instead of the term 'Parikh image.' In order to simplify terms, in the sequel we will also say 'the Parikh image of an automaton $\mathfrak{A}$' whenever we actually mean 'the Parikh image of the language recognized by the automaton $\mathfrak{A}$.'

The main result concerning the Parikh mapping is the following theorem, which is due to Parikh [Par66].

Theorem 2.14 (Parikh)**.** *The Parikh image of every context-free language is effectively semi-linear.*

The term 'effective' in Theorem 2.14 refers to the following; if a context-free language L is given by a context-free grammar or by a PDA, equivalently, then we can effectively find the generators of $\Phi(L)$.

The converse of Parikh's theorem also holds. That is, for any semi-linear set A, given by its generators, we can construct a context-free language L or a PDA $\mathfrak{A}$, equivalently, such that the Parikh image of L or $\mathfrak{A}$, respectively, yields A. Moreover, for A we can construct even a regular language or a finite automaton, equivalently, whose Parikh image yields A. The latter implies that every context-free language, with respect to its Parikh image, is equivalent to a regular language.

Parikh's theorem and its converse also give an automata characterization of semi-linear sets by means of finite automata and by means of pushdown automata. As a remark, there are other automata characterizations of semi-linear sets in the literature. For example, Harju et al. [HIKS02] introduced the notion of 'finite-state generators' and proved that each semi-linear set is the Parikh image of a finite-state generator and that the Parikh image of any finite-state generator is semi-linear.

Decision Problems

Many decision problems for semi-linear sets, such as the membership, the infiniteness, and the emptiness problem, are decidable. In the following we briefly present some decision procedures for the membership and the emptiness problem; throughout this thesis, we will make use of the decidability of these two problems.

The membership problem for semi-linear sets is the question

> Given a semi-linear set $A \subseteq \mathbb{N}^n$, $n \geq 1$, and a vector $\bar{x} \in \mathbb{N}^n$, does $\bar{x}$ belong to A?

This problem is decidable regardless of whether A is given by means of its generators or by means of a Presburger formula.

If A is given as a finite union of linear sets, then we can systematically check whether $\bar{x}$ belongs to one of these linear sets. Thus, without loss of generality we can assume that A is a linear set, given by its generators $\bar{x}_0, \ldots, \bar{x}_m$, for some $m \geq 0$. Then, we may systematically try all possible values of $k_1, \ldots, k_m \in \{0, \ldots, \max_{0 \leq i \leq n}(\bar{x})_i\}$ and check whether $\bar{x}$ is equal to the vector obtained from $\bar{x}_0 + k_1\bar{x}_1 + \ldots + k_m\bar{x}_m$.

If A is given by means of a Presburger formula $\varphi(z_1, \ldots, z_n)$, then we can check whether the Presburger sentence $\varphi[z_1/(\bar{x})_1, \ldots, z_n/(\bar{x})_n]$ is true in the structure $(\mathbb{N}, +)$. Then, the decidability of the membership problem follows from the fact that Presburger arithmetic is decidable.

The emptiness problem for semi-linear sets is the question

Given a semi-linear set $A \subseteq \mathbb{N}^n$, $n \geq 1$, is A empty?

As with the membership problem, this problem is decidable regardless of whether A is given by means of its generators or by means of a Presburger formula.

By definition, every linear set is not empty. Thus, by Remark 2.2(d), the semi-linear set A is empty if and only if A is the union of zero linear sets. In other words, if A is given by the set of its (finitely many) generators, it suffices to check whether the latter set is empty.

If A is given by means of a Presburger formula $\varphi(z_1, \ldots, z_n)$, then A is empty if and only if the Presburger sentence $\exists z_1 \ldots \exists z_n \varphi(z_1, \ldots, z_n)$ is false in the structure $(\mathbb{N}, +)$. Again, since Presburger arithmetic is decidable, the emptiness problem for A is decidable too.

The decidability of the emptiness problem for semi-linear sets has many applications, in particular in formal language theory. As noted in Remark 2.13, the emptiness problem for a language can be reduced to the emptiness problem for its Parikh image. Consequently, if the Parikh image of a language is effectively semi-linear, then, by the decidability of the emptiness problem for semi-linear sets, the emptiness problem for this language is decidable. For instance, the decidability of the emptiness problem for context-free languages follows immediately from Parikh's theorem.

Chapter 3

Parikh Automata

We introduce the notion of Parikh automata (on words) in the sense of Klaedtke and Rueß [KR02, KR03]. Most of the definitions and results given here follow those in [KR02].

Generally, we want to equip an automaton with *counting* capabilities by requiring that the Parikh image of the words accepted by this automaton satisfies a given arithmetical constraint. As an illustration, consider the language $L_{\text{abc}} := \{a^k b^k c^k \mid k \geq 1\}$, which is not regular (it is not even context-free) and thus cannot be recognized by a finite automaton. Nevertheless, this language can be recognized as follows. We use a finite automaton which recognizes $a^+b^+c^+$ and additionally require that the Parikh image of each word w recognized by this automaton satisfies the arithmetical constraint $|w|_a = |w|_b = |w|_c$.

In the definition of the Parikh mapping, furthermore, each input symbol is associated with a *unit vector* of natural numbers (see Remark 2.12). We extend this notion to allowing each symbol to be associated with (possibly more than one) vectors of natural numbers that are not necessarily unit vectors. Formally, we work with an enlarged alphabet whose elements are pairs of the form $(a, \bar{x})$, where a is a symbol from the actual input alphabet, and $\bar{x}$ is a vector of natural numbers. In addition, we define a projection function which extracts the symbol component of such a pair and an extension of the Parikh mapping which extracts the vector component.

Definition 3.1. Let Σ be an alphabet and $n \geq 1$. Further, let D be a finite, nonempty subset of $\mathbb{N}^n$. The Σ-*projection* $\Psi\colon (\Sigma \times D)^* \to \Sigma^*$ is defined by

- $\Psi(\varepsilon) := \varepsilon$,
- $\Psi(a, \bar{d}) := a$, for each $(a, \bar{d}) \in \Sigma \times D$, and
- $\Psi(uv) := \Psi(u)\Psi(v)$, for each $u, v \in (\Sigma \times D)^*$.

The *extended Parikh mapping* $\tilde{\Phi}\colon (\Sigma \times D)^* \to \mathbb{N}^n$ is defined by

- $\tilde{\Phi}(\varepsilon) := 0^n$,
- $\tilde{\Phi}(a, \bar{d}) := \bar{d}$, for each $(a, \bar{d}) \in \Sigma \times D$, and
- $\tilde{\Phi}(uv) := \tilde{\Phi}(u) + \tilde{\Phi}(v)$, for each $u, v \in (\Sigma \times D)^*$.

For a language $L \subseteq (\Sigma \times D)^*$, we define the Σ-*projection* of L

$$\Psi(L) := \{\Psi(w) \mid w \in L\} \subseteq \Sigma^* ,$$

and the *extended Parikh image* of L

$$\tilde{\Phi}(L) := \{\tilde{\Phi}(w) \mid w \in L\} \subseteq \mathbb{N}^n .$$

In the definition above, intuitively, Σ represents the actual input alphabet, and D contains all the vectors of natural numbers that can be attached to an input symbol. Thus, given a language L over the extended alphabet $\Sigma \times D$, the language of interest is a subset of the Σ-projection of L, which yields a language over Σ. Further, by using the extended Parikh mapping we can require that the language of interest satisfies certain arithmetical constraints. As arithmetical constraints, we consider sets of vectors of natural numbers.

Definition 3.2. Let Σ be an alphabet and $n \geq 1$. Further, let D be a finite, nonempty subset of $\mathbb{N}^n$. For a language $L \subseteq (\Sigma \times D)^*$ and a (*constraint*) set $C \subseteq \mathbb{N}^n$, we define the *restriction* of L with respect to C by

$$L\restriction_C := \{\Psi(w) \mid w \in L \text{ and } \tilde{\Phi}(w) \in C\} .$$

Definition 3.1 and Definition 3.2 constitute the core idea of a Parikh automaton. We need an automaton for the language $L \subseteq (\Sigma \times D)^*$ (which we will refer to in the following as the *automata component* of the Parikh automaton under consideration) and a set of vectors of natural numbers for the constraint set C. This kind of definition offers two advantages regarding flexibility. For one thing, we can employ any model of automata on words (over $\Sigma \times D$) for the automata component, leaving the definitions of runs, acceptance, and so on as is. For another thing, we can employ any class of vector sets for the constraint set.

For the rest of this chapter, we will consider finite automata and semilinear sets, as Klaedtke and Rueß did in [KR02, KR03]. They referred to the resulting automata as *Parikh finite word automata*. As we will only focus on automata on words, we will simply refer to these automata as Parikh finite automata, thereby omitting the reference to words. In the following section we introduce Parikh finite automata formally and present some closure properties. Then, we conclude this chapter with an analysis of some decision problems for Parikh finite automata.

3.1 Parikh Finite Automata

Definition 3.3. A *Parikh finite automaton* (*PFA*) of dimension $n \geq 1$ over an alphabet Σ is a system $(\mathfrak{A}, C)$ where $\mathfrak{A} := (Q, \Sigma \times D, \delta, q_0, F)$ is a finite automaton over $\Sigma \times D$, for some finite, nonempty set $D \subseteq \mathbb{N}^n$ (called the *auxiliary set*), and $C \subseteq \mathbb{N}^n$ is a semi-linear set (called the *constraint set*). A *deterministic Parikh finite automaton* (*DPFA*) of dimension $n \geq 1$ over an alphabet Σ is a PFA $(\mathfrak{A}, C)$ of dimension n over Σ where the transition function δ satisfies $\sum_{\bar{d} \in D} |\delta(q, (a, \bar{d}))| \leq 1$, for each $q \in Q$ and $a \in \Sigma$. In other words, $\mathfrak{A}$ can only proceed by reading $(a, \bar{d})$, for at most one vector $\bar{d} \in D$.

The language *recognized* by $(\mathfrak{A}, C)$ is defined as $L(\mathfrak{A}, C) := L(\mathfrak{A})\restriction_C$, where $L(\mathfrak{A}) \subseteq (\Sigma \times D)^*$ is the language recognized by $\mathfrak{A}$ (in the usual sense).

A language L over an alphabet Σ is called *PFA-recognizable* or *DPFA-recognizable* if there is a PFA or DPFA, respectively, $(\mathfrak{A}, C)$ of dimension n over Σ, for some $n \geq 1$, such that $L(\mathfrak{A}, C) = L$. The class of PFA-recognizable and DPFA-recognizable languages over Σ are denoted by $\mathcal{L}_{\mathrm{PFA}}(\Sigma)$ and $\mathcal{L}_{\mathrm{DPFA}}(\Sigma)$, respectively. We will omit the reference to Σ whenever Σ is clear from the context.

Intuitively, a PFA of dimension $n \geq 1$ can be seen as a finite automaton that is augmented with n-many counters. Reading the symbol $(a, \bar{d}) \in \Sigma \times D$ corresponds to reading the input symbol $a \in \Sigma$ while incrementing the counters by the vector $\bar{d}$; that is, the i-th counter is incremented by $(\bar{d})_i$, for $i = 1, \ldots, n$. Note that, in this setting, the counters are monotonic in the sense that they may only be incremented, but not decremented. Then, at the end of a computation the value of the counters, that is, the sum of the vectors that have been seen during the computation, is checked against an arithmetical constraint, which is given by a semi-linear set.

In this view, it should also be clear that a finite automaton can be captured as a PFA; the former is just a PFA of dimension 1 that never changes the values of its counters. We state this fact in the following remark.

Remark 3.4. Given a deterministic finite automaton $\mathfrak{A} = (Q, \Sigma, \delta, q_0, F)$, we can construct a DPFA $(\mathfrak{A}', C)$ of dimension 1 that recognizes $L(\mathfrak{A})$ as follows. We use $D := \{\bar{0}\}$ as the auxiliary set and $C := \{\bar{0}\}$ as the constraint set. Note that C, by Remark 2.2(b), is semi-linear. Intuitively, for $\mathfrak{A}'$ we use the same transition structure as $\mathfrak{A}$; we merely attach the zero vector to every transition. Formally, we define $\mathfrak{A}' := (Q, \Sigma \times D, \delta', q_0, F)$, where the transition function δ' is given by

$$\delta'(q, (a, \bar{0})) := \delta(q, a)\,,$$

for each $q \in Q$ and $a \in \Sigma$.

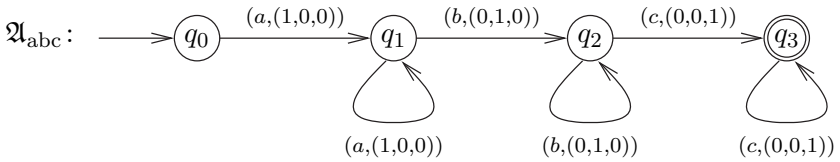


Figure 3.1: Automaton of Example 3.5

Now, for any run of $\mathfrak{A}'$, the sum of the attached vectors is always the zero vector and thus always belongs to C. Thus, whether or not the DPFA $(\mathfrak{A}', C)$ accepts a given input word depends only on whether or not the deterministic finite automaton $\mathfrak{A}$ accepts this word. Hence, we conclude that $L(\mathfrak{A}', C) = L(\mathfrak{A})$.

Since every regular language or FA-recognizable language, equivalently, is recognizable by a deterministic finite automata, it follows that every regular language is DPFA-recognizable. In other words, we have $\mathcal{L}_{\text{FA}} \subseteq \mathcal{L}_{\text{DPFA}}$.

We will now see some examples of Parikh automata. In the following examples, let $\Sigma := \{a, b, c\}$.

Example 3.5. As noted at the beginning of this chapter, the language

$$L_{\text{abc}} = \{a^k b^k c^k \mid k \geq 1\}$$

can be recognized as follows. We use the auxiliary set

$$D_{\text{abc}} := \{(1,0,0), (0,1,0), (0,0,1)\}$$

of dimension 3 and the finite automaton $\mathfrak{A}_{\text{abc}}$ depicted in Figure 3.1. Intuitively, by using $\mathfrak{A}_{\text{abc}}$ we only accept those words of the form $a^+b^+c^+$. Now, by using the semi-linear set

$$C_{\text{abc}} := \{k(1,1,1) \mid k \in \mathbb{N}\}$$

as constraint set, we additionally require that the words accepted have the same number of a's, b's, and c's.

Note that the resulting PFA $(\mathfrak{A}_{\text{abc}}, C_{\text{abc}})$ is deterministic, and thus, L_{abc} is DPFA-recognizable.

Example 3.6. Let

$$L_{\text{unbal}} := \{ucv \mid u, v \in \{a, b\}^* \text{ and } |u| \neq |v|\}.$$

That is, each word of this language is of the form ucv where u and v are words of different length over $\{a, b\}$.

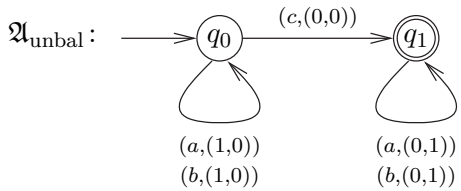


Figure 3.2: Automaton of Example 3.6

The following DPFA of dimension 2 recognizes L_{unbal}. We use the set $D_{\text{unbal}} := \{(1,0),(0,1)\}$ as auxiliary set and the finite automaton $\mathfrak{A}_{\text{unbal}}$ depicted in Figure 3.2 as the automata component. For the constraint set, we use the set $C_{\text{unbal}} := C_> \cup C_<$ defined by

$$C_> := \{(1,0) + k_1(1,0) + k_2(1,1) \mid k_1, k_2 \in \mathbb{N}\}$$

and

$$C_< := \{(0,1) + k_1(0,1) + k_2(1,1) \mid k_1, k_2 \in \mathbb{N}\}\,.$$

Intuitively, $\mathfrak{A}_{\text{unbal}}$ first counts the length of u. After a c has been seen, it counts the length of v. Then, the linear set $C_>$ takes care of the case $|u| > |v|$ whereas the linear set $C_<$ takes care of the case $|u| < |v|$.

In the classical automata theory, it turned out to be useful to require that an automaton recognizing a language possesses certain properties; for instance, we can assume, without loss of generality, that a finite automaton recognizing a regular language is complete in the sense that, for each of its states and for each input symbol, there is always some state to which the automaton can proceed. In this spirit, the following two lemmas assert that each PFA-recognizable language and each DPFA-recognizable language are recognizable by a PFA or a DPFA, respectively, that possesses certain properties.

Definition 3.7. A PFA or DPFA $(\mathfrak{A}, C)$ of dimension $n \geq 1$, where $\mathfrak{A} = (Q, \Sigma \times D, \delta, q_0, F)$, is said to be *complete* if for each state $q \in Q$ and each symbol $a \in \Sigma$ there is a vector $\bar{d} \in D$ such that $\delta(q,(a,\bar{d})) \neq \emptyset$. In other words, the transition function δ satisfies

$$\sum_{\bar{d} \in D} |\delta(q,(a,\bar{d}))| \geq 1\,,$$

for each $q \in Q$ and $a \in \Sigma$.

Lemma 3.8. *For each PFA (DPFA), there exists an equivalent PFA (DPFA) that is complete and whose initial state has no ingoing transitions.*

Proof. Let $(\mathfrak{A}, C)$ be a PFA of dimension $n \geq 1$, where $\mathfrak{A} = (Q, \Sigma \times D, q_0, \delta, F)$ and $C \subseteq \mathbb{N}^n$ is a semi-linear set.

First, if $(\mathfrak{A}, C)$ is not complete yet, then we add some transitions as follows, thereby making $(\mathfrak{A}, C)$ complete. We introduce a new, non-final sink state q_s and arbitrarily choose a vector $\bar{d} \in D$ (recall that D is not empty). Now, for each pair of $q \in Q$ and $a \in \Sigma$ that violates the completeness condition above, we add a new transition from q to q_s via $(a, \bar{d})$. Further, for each $a \in \Sigma$, we add a transition from q_s to itself via $(a, \bar{d})$.

Second, if the initial state of the resulting complete PFA still has some ingoing transitions, then we introduce a new initial state q_0' and copy the outgoing transitions of q_0 to q_0'. In addition, we declare q_0' to be a final state if and only if q_0 is also a final state. Note that this procedure does not affect the completeness of the PFA obtained from the first procedure above.

It is straightforward to see that the resulting PFA recognizes the same language as the original one. Moreover, if the original PFA is deterministic, then so is the resulting PFA. □

The following lemma asserts that, for PFA-recognizable and DPFA-recognizable languages, we may give up the role of the final states, namely by declaring all states to be final states and by coding the information whether a final state has been reached or not in some additional vector components. This idea is already stated implicitly in [KR02].

Lemma 3.9. *For each PFA (DPFA), there exists an equivalent PFA (DPFA) in which all the states are final states. In addition, the latter PFA (DPFA) is complete and its initial state has no ingoing transitions.*

Proof. Let $(\mathfrak{A}, C)$ be a PFA of dimension $n \geq 1$, where $\mathfrak{A} = (Q, \Sigma \times D, q_0, \delta, F)$ and $C \subseteq \mathbb{N}^n$ is a semi-linear set. By Lemma 3.8, we may assume, without loss of generality, that $(\mathfrak{A}, C)$ is complete and that q_0 has no ingoing transitions.

As noted before, the PFA $(\mathfrak{A}, C)$ can be seen as a finite automaton with n-many monotonic counters. In addition to these counters, we now use two monotonic counters c_1 and c_2 to record whether a final state is being assumed or not. The intuition is that whenever we are assuming a final state, the *difference* between c_1 and c_2 should be 1. That is, $c_1 = c_2 + 1$. By contrast, whenever we are assuming a non-final state, then the values of c_1 and c_2 should be equal.

Updating the values of these counters is implemented as follows. We start with both counters having the same values. If we enter a final state from a

non-final state, then we increment c_1 by 1, thereby incrementing the difference between c_1 and c_2 by 1. Conversely, if we leave a final state, then we increment c_2 by 1, thereby balancing up the difference between c_1 and c_2. Using an appropriate constraint set, we require that at the end of a computation the difference between c_1 and c_2 is 1, which signals that a final state is being assumed at the end of the computation.

Formally, we define a PFA $(\mathfrak{A}', C')$ of dimension $n+2$, where $\mathfrak{A}' := (Q, \Sigma \times D', \delta', q_0, Q)$, as follows. We define the auxiliary set by setting

$$D' := D \otimes \{0,1\}^2$$

and the transition function by setting

$$\delta'(p, (a, \bar{d} \otimes (x_1, x_2))) := \{q \mid q \in \delta(p, (a, \bar{d})) \text{ and } (x_1, x_2) = \mathrm{status}_{\mathfrak{A}}(p, q)\},$$

for each $p \in Q$, $a \in \Sigma$, $\bar{d} \in D$, and $(x_1, x_2) \in \{0,1\}^2$, where the function $\mathrm{status}_{\mathfrak{A}} \colon Q \times Q \to \{0,1\}^2$ is given by

$$\mathrm{status}_{\mathfrak{A}}(p,q) := \begin{cases} (1,0) & \text{if } p \neq q_0,\ p \notin F,\ q \in F, \\ (0,1) & \text{if } p \neq q_0,\ p \in F,\ q \notin F, \\ (1,0) & \text{if } p = q_0,\ q \in F, \\ (1,1) & \text{if } p = q_0,\ q \notin F, \\ (0,0) & \text{otherwise,} \end{cases} \tag{3.1}$$

for each $p, q \in Q$. The first case in (3.1) corresponds to entering a final state from a non-final state whereas the second one corresponds to leaving from a final state to a non-final state. The third and the fourth case are concerned with the initial state: If the state following the initial state is a final state, then we signal this by incrementing the first counter. If this state is a non-final state, then we increment the second counter. However, at the beginning of a computation all counters, of course, are set to zero. Thus, we have to increment the first counter as well (recall that being in a non-final state is represented by the *difference* of the counters being balanced up). Note that in the fourth case, we do not assign the zero vector since the zero vector will be used below to deal with the empty word.

Now, the constraint set C', at the end of a computation, must ensure that the values of the original counters belong to C and that the difference between the first and the second counter is 1, which signals that a final state has been reached.

At the beginning of a computation, all counters are set to zero. Thus, our two additional counters cannot signal whether the initial state is already a

final state or not. Hence, we must take care of the empty word separately. For the empty word, we note the following: the PFA $(\mathfrak{A}, C)$ accepts the empty word if and only if the initial state q_0 is a final state and the constraint set C contains the zero vector.

Putting it together, we define the constraint set C' to be

$$C' := (C \otimes \{(1,0) + k(1,1) \mid k \in \mathbb{N}\}) \cup M\,,$$

where

$$M := \begin{cases} \{0^{n+2}\} & \text{if } q_0 \in F \text{ and } 0^n \in C, \\ \emptyset & \text{otherwise.} \end{cases}$$

The set M is semi-linear since it is finite. Since the class of semi-linear sets is closed under concatenation and union, the set C' is semi-linear as well.

It is straightforward to show that $L(\mathfrak{A}', C') = L(\mathfrak{A}, C)$. It should also be clear that if $(\mathfrak{A}, C)$ is deterministic, then so is the resulting $(\mathfrak{A}', C')$. Moreover, it is not difficult to verify that the resulting $(\mathfrak{A}', C')$ is complete and that its initial state has no ingoing transitions.

As a remark, the assumption that the initial state of $(\mathfrak{A}, C)$ has no ingoing transitions is needed in the definition of the function $\text{status}_{\mathfrak{A}}$ in (3.1); there we have implicitly assumed that the initial state only occurs at the beginning of a computation. The assumption that $(\mathfrak{A}, C)$ is complete, actually, is not needed for this lemma; we only included it to simplify the proofs of some closure properties of PFA's and DPFA's below. □

We now consider some operations on languages that preserve PFA-recognizability or DPFA-recognizability, respectively. In all the results below, the proofs are effective and are adapted from standard techniques of automata theory.

Theorem 3.10 (Klaedtke and Rueß)**.** *The classes $\mathcal{L}_{\text{PFA}}$ and $\mathcal{L}_{\text{DPFA}}$ are effectively closed under intersection and union.*

Proof. Let $(\mathfrak{A}_i, C_i)$ be PFA's of dimension $n_i \geq 1$, for $i = 1, 2$, where $\mathfrak{A}_i = (Q_i, \Sigma \times D_i, \delta_i, q_{0i}, F_i)$. By Lemma 3.9, we can assume, without loss of generality, that $(\mathfrak{A}_i, C_i)$ are complete, that $F_i = Q_i$, and that q_{0i} have no ingoing transitions, for $i = 1, 2$.

We simulate $(\mathfrak{A}_1, C_1)$ and $(\mathfrak{A}_2, C_2)$ in parallel, by using the Cartesian product of the state sets Q_1 and Q_2, and by working with vectors of dimension $n_1 + n_2$. On a given input word, we use the first n_1 dimensions and the last n_2 dimensions to store the vector obtained from the computation of $(\mathfrak{A}_1, C_1)$ and $(\mathfrak{A}_2, C_2)$, respectively. Then, at the end of the computation we check whether

the former vector belongs to C_1 and/or whether the latter vector belongs to C_2. Note that we do not need to concern ourselves with the question whether a final state has been reached or not at the end of the computation; all the states are final states.

Formally, let $\mathfrak{A} := (Q_1 \times Q_2, \Sigma \times D, \delta, (q_{01}, q_{02}), Q_1 \times Q_2)$, where the auxiliary set D is given by

$$D := D_1 \otimes D_2 \,,$$

and the transition function δ is given by

$$\delta((p_1, p_2), (a, \bar{d}_1 \otimes \bar{d}_2)) := \{(q_1, q_2) \mid q_1 \in \delta_1(p_1, (a, \bar{d}_1)) \text{ and } q_2 \in \delta_2(p_2, (a, \bar{d}_2))\} \,,$$

for each $p_1 \in Q_1$, $p_2 \in Q_2$, $\bar{d}_1 \in D_1$, $\bar{d}_2 \in D_2$, and $a \in \Sigma$. For the intersection, we use the constraint set

$$C_\cap := C_1 \otimes C_2 \,,$$

and for the union

$$C_\cup := (C_1 \otimes \mathbb{N}^{n_2}) \cup (\mathbb{N}^{n_1} \otimes C_2) \,.$$

By Remark 2.2(e), the sets $\mathbb{N}^{n_1}$ and $\mathbb{N}^{n_2}$ are semi-linear. Since the class of semi-linear sets is closed under concatenation and union, it follows that the sets $C_\cap$ and $C_\cup$ are semi-linear.

Now, it is straightforward to show that $(\mathfrak{A}, C_\cap)$ recognizes $L(\mathfrak{A}_1, C_1) \cap L(\mathfrak{A}_2, C_2)$ and that $(\mathfrak{A}, C_\cup)$ recognizes $L(\mathfrak{A}_1, C_1) \cup L(\mathfrak{A}_2, C_2)$. Note that, for the union, we need the assumption that the given PFA's are complete; otherwise, it might be the case that one of the given PFA's (and thus also $(\mathfrak{A}, C_\cup)$) cannot continue processing the input although the other one might accept. Moreover, if $(\mathfrak{A}_1, C_1)$ and $(\mathfrak{A}_2, C_2)$ are deterministic, then so are the resulting $(\mathfrak{A}, C_\cap)$ and $(\mathfrak{A}, C_\cup)$. □

Theorem 3.11 (Klaedtke and Rueß)**.** *The classes $\mathcal{L}_{\mathrm{PFA}}$ and $\mathcal{L}_{\mathrm{DPFA}}$ are effectively closed under inverse homomorphisms.*

Proof. Let $(\mathfrak{A}, C)$ be a PFA of dimension $n \geq 1$, where $\mathfrak{A} = (Q, \Sigma \times D, \delta, q_0, F)$, and let $h \colon \Pi^* \to \Sigma^*$ be a homomorphism.

We construct a PFA $(\mathfrak{B}, C)$ of dimension n that recognizes $h^{-1}(L(\mathfrak{A}, C))$ as follows, leaving the constraint set C as is. For $\mathfrak{B}$, we use the same set of states, the same initial state, and the same set of final states as $\mathfrak{A}$.

Intuitively, $\mathfrak{B}$ has a transition from state p to state q via $(b, \bar{d})$ if and only if $\mathfrak{A}$ can reach state q from state p via word $w \in (\Sigma \times D)^*$ with $\Psi(w) = h(b)$ and $\tilde{\Phi}(w) = \bar{d}$, where Ψ is the Σ-projection over $\Sigma \times D$, and $\tilde{\Phi}$ is the extended

Parikh mapping over $\Sigma \times D$. Hence, we define the auxiliary set D' of $(\mathfrak{B}, C)$ as

$$D' := \{\tilde{\Phi}(w) \mid w \in (\Sigma \times D)^* \text{ and there is some } b \in \Pi \text{ such that } \Psi(w) = h(b)\}\,.$$

First, we note that the so-defined set D' is nonempty since both Π and D, by definition, are nonempty. Second, the set D' is finite and effectively computable; since Π and D are finite, there are only finitely many words w over $\Sigma \times D$ that satisfies the condition

$$\text{there is some } b \in \Pi \text{ such that } \Psi(w) = h(b)\,.$$

Now, let $\mathfrak{B} := (Q, \Pi \times D', \delta', q_0, F)$, where δ' is given by

$$\delta'(p, (b, \bar{d})) := \{q \in Q \mid \text{there is some } w \in (\Sigma \times D)^* \text{ such that } \Psi(w) = h(b), \tilde{\Phi}(w) = \bar{d}, \text{ and } \mathfrak{A}\colon p \xrightarrow{w} q\}\,,$$

for each $p \in Q$, $b \in \Pi$, and $\bar{d} \in D'$. Again, we note that $\delta'(p, (b, \bar{d}))$ is finite and effectively computable; since D is finite, there is only finitely many words w over $\Sigma \times D$ that satisfies the condition $\Psi(w) = h(b)$. Further, we note that if $h(b) = \varepsilon$, then from each state $p \in Q$ there is a transition back to p via $(b, \bar{0})$, for each $b \in \Pi$.

It is now straightforward to show that the construction indeed yields the desired PFA. Moreover, if $(\mathfrak{A}, C)$ is deterministic, then so is the resulting PFA $(\mathfrak{B}, C)$. □

Theorem 3.12 (Klaedtke and Rueß)**.** *The class $\mathcal{L}_{\text{PFA}}$ is effectively closed under homomorphisms.*

Proof. Let $(\mathfrak{A}, C)$ be a PFA of dimension $n \geq 1$ where $\mathfrak{A} := (Q, \Sigma \times D, \delta, q_0, F)$ and $h\colon \Sigma^* \to \Pi^*$ be a homomorphism. The task is to construct a PFA that recognizes $h(L(\mathfrak{A}, C)) \subseteq \Pi^*$.

Roughly speaking, if in $\mathfrak{A}$ we can make a transition from state p to state q via $(a, \bar{d})$, and $h(a) = b_1 \cdots b_m \in \Pi^*$, then in the new automaton we should have a run from p to q via the word $(b_1, \bar{d}_1) \cdots (b_m, \bar{d}_m)$, where $\bar{d} = \bar{d}_1 + \cdots \bar{d}_m$. The latter condition can be fulfilled by setting $\bar{d}_1$ to be equal to $\bar{d}$ and setting $\bar{d}_2, \ldots, \bar{d}_m$ to be the zero vector. In other words, we attach the vector $\bar{d}$ to the first symbol of $h(a)$.

However, a complication arises with this approach. If $h(a) = \varepsilon$, then it is not clear to which symbol we should attach the vector $\bar{d}$. To overcome this difficulty, we first extend the alphabet Π by introducing a new symbol

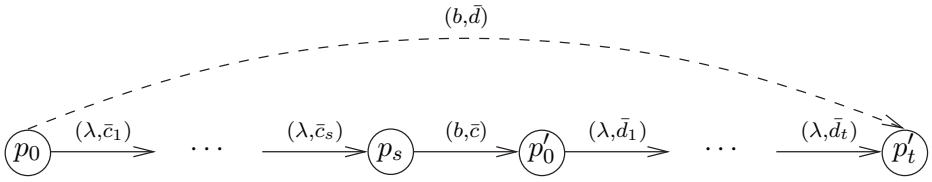


Figure 3.3: Elimination of λ-transitions

$\lambda \notin \Sigma \cup \Pi$, which syntactically represents the empty word ε, and construct a PFA over $\Pi \cup \{\lambda\}$. In the next step, we then eliminate the introduced symbol λ.

Let $\Pi_\lambda := \Pi \cup \{\lambda\}$ and $D' := D \cup \{\bar{0}\}$. We define the finite automaton $\mathfrak{A}' := (Q', \Pi_\lambda \times D', \delta', q_0, F)$ as follows. The state set Q' of $\mathfrak{A}'$ contains all states of $\mathfrak{A}$ and, in addition, all new intermediate states introduced in the definition of δ' below. For each $p, q \in Q$, $a \in \Sigma$, and $\bar{d} \in D$, if $q \in \delta(p, (a, \bar{d}))$, then we introduce the following transitions in $\mathfrak{A}'$:

1. If $h(a) = \varepsilon$, then we define the transition $q \in \delta'(p, (\lambda, \bar{d}))$.
2. If $h(a) = b \in \Pi$, then we define the transition $q \in \delta'(p, (b, \bar{d}))$.
3. If $h(a) = b_1 \cdots b_m \in \Pi^+$, for some $m \geq 2$, then, we introduce $(m-1)$-many new intermediate states $q_1, \ldots, q_{m-1}$ and the transitions
   - $q_1 \in \delta'(p, (b_1, \bar{d}))$,
   - $q_{i+1} \in \delta'(q_i, (b_{i+1}, \bar{0}))$, for $i = 1, \ldots, m-2$, and
   - $q \in \delta'(q_{m-1}, (b_m, \bar{0}))$.

For the constraint set, we use the constraint set C of the original PFA. Then, it is not difficult to show that $(\mathfrak{A}', C)$ recognizes $h(L(\mathfrak{A}, C))$ (up to the special symbol λ).

We turn to the elimination of λ. Without loss of generality, we may assume that the initial state of $\mathfrak{A}'$ has no ingoing transitions. To simplify notations, we assume that $Q' = \{q_0, \ldots, q_r\}$, for some $r \geq 0$, where q_0 is the initial state of $\mathfrak{A}'$.

Basically, we employ the procedure for the elimination of ε-transitions in nondeterministic finite automata. We first choose a non-λ-transition $(b, \bar{c}) \in \Pi \times D'$ and merge with it some λ-transitions *before* and *after* this non-λ-transition. This idea is illustrated in Figure 3.3, where $p_0, \ldots, p_s, p'_0, \ldots, p'_t$ are states in Q', and $\bar{c}_1, \ldots, \bar{c}_s, \bar{c}, \bar{d}_1, \ldots, \bar{d}_t$ are vectors in D'. We merge the transitions depicted as solid lines into the transition $(b, \bar{d})$ that is depicted as a dashed line, where $\bar{d} = \bar{c}_1 + \cdots + \bar{c}_s + \bar{c} + \bar{d}_1 + \cdots + \bar{d}_t$.

In general, the number of the λ-transitions that can be merged in this way is unbounded. To overcome this difficulty, we first only consider sequences of λ-transitions without state repetitions. Then, if a state, say q, occurs in the sequence of λ-transitions under consideration, we must take into account that there might have been a sequence of λ-transitions from q back to q (that is, a loop of λ-transitions), which we have left out. We will take care of the vectors resulting from such loops of λ-transitions later by defining an appropriate constraint set.

Now, we define

$$\begin{aligned}\tilde{D} := \{\bar{d} \mid &\text{ there are some vectors } \bar{c}_1, \ldots, \bar{c}_s, \bar{c}, \bar{d}_1, \ldots, \bar{d}_t \in D', \\ &\text{for some } s, t < |Q'|, \text{ such that} \\ &\bar{d} = \bar{c}_1 + \cdots + \bar{c}_s + \bar{c} + \bar{d}_1 + \cdots + \bar{d}_t\} \,.\end{aligned}$$

Note that $\tilde{D}$ is finite since D' is finite. Intuitively, $\tilde{D}$ contains the vectors that can be attached to a symbol in Π by considering the possible λ-transitions before and after the respective symbol in Π. Moreover, as noted above, we only consider sequences of λ-transitions whose length is less than $|Q'|$ since after $|Q'|$-many λ-transitions, a state repetition must occur.

We will construct a PFA $(\tilde{\mathfrak{A}}, \tilde{C})$ of dimension $n + r$ with the auxiliary set $\tilde{D} \otimes \{0,1\}^r$, which recognizes $h(L(\mathfrak{A}, C))$ or, up to the special symbol λ, $L(\mathfrak{A}', C)$. The r-many additional dimensions will be used to record whether a state occurs in a run of $\mathfrak{A}'$, and thus, as noted before, we will have to deal with the possibility of having a loop of λ-transitions at this state. Note that we only need r-many additional dimensions, instead of $(r+1)$-many (that is, instead of $|Q|$-many), since the initial state occurs in every run.

Let

$$\tilde{\mathfrak{A}} := (Q', \Pi \times (\tilde{D} \otimes \{0,1\}^r), \tilde{\delta}, q_0, \tilde{F}) \,,$$

where the transition function $\tilde{\delta}$ is defined as follows. For each $p, q \in Q'$, $b \in \Pi$, $\bar{d} \in \tilde{D}$, and $\bar{z} \in \{0,1\}^r$, we define

$$q \in \tilde{\delta}(p, (b, \bar{d} \otimes \bar{z}))$$

if and only if there are some states $p_0, \ldots, p_s, p'_0, \ldots, p'_t \in Q'$, for some $s, t < |Q'|$, and there are some vectors $\bar{c}_1, \ldots, \bar{c}_s, \bar{c}, \bar{d}_1, \ldots, \bar{d}_t \in D'$ such that

- the states $p_0, \ldots, p_s$ are pairwise different,
- the states $p'_0, \ldots, p'_t$ are pairwise different,
- $p_0 = p$ and $p'_t = q$,
- $p_i \in \delta'(p_{i-1}, (\lambda, \bar{c}_i))$, for $i = 1, \ldots, s$,

- $p'_0 \in \delta'(p_s, (b, \bar{c}))$,
- $p'_j \in \delta'(p'_{j-1}, (\lambda, \bar{d}_j))$, for $j = 1, \ldots, t$,
- $\bar{d} = \bar{c}_1 + \cdots + \bar{c}_s + \bar{c} + \bar{d}_1 + \cdots + \bar{d}_t$, and
- $(\bar{x})_k = 1$ if and only if $q_k \in Q' \setminus \{q_0\}$ occurs among the states $p_0, \ldots, p_s$, $p'_0, \ldots, p'_t$, for $k = 1, \ldots, r$.

The set $\tilde{F}$ of final states will be defined later.

Toward the definition of the semi-linear constraint set $\tilde{C}$, we define, for $i = 0, \ldots, r$, the set C_i of the vectors resulting from the loops of λ-transitions that start and end in q_i, by setting

$$C_i := \{\tilde{\Phi}(w) \mid w \in (\{\lambda\} \times D')^* \text{ and } \mathfrak{A}' \colon q_i \xrightarrow{w} q_i\} \subseteq \mathbb{N}^n \,, \tag{3.2}$$

where $\tilde{\Phi} \colon (\Pi_\lambda \times D')^* \to \mathbb{N}^n$ is the extended Parikh mapping over $\Pi_\lambda \times D'$. The set C_i is effectively computable and semi-linear, as can be seen as follows. We obtain $\mathfrak{A}'_i$ from $\mathfrak{A}'$ by deleting all non-λ-transitions, declaring q_i as the initial state, and declaring $\{q_i\}$ as the set of final states. Clearly, the set C_i is the extended Parikh image of $L(\mathfrak{A}'_i)$. By Parikh's theorem, the Parikh image of $L(\mathfrak{A}'_i)$ is effectively semi-linear; after all, $\mathfrak{A}'_i$ is a finite automaton. Then, by Lemma 3.19 (see Section 3.2), the extended Parikh image of $L(\mathfrak{A}'_i)$, and thus also C_i, is effectively semi-linear.

With regard to the empty word, we also need to define the set $C_{0,i}$ of the vectors resulting from the sequences of λ-transitions that start in q_0 and end in q_i, by setting

$$C_{0,i} := \{\tilde{\Phi}(w) \mid w \in (\{\lambda\} \times D')^* \text{ and } \mathfrak{A}' \colon q_0 \xrightarrow{w} q_i\} \subseteq \mathbb{N}^n \,,$$

for $i = 0, \ldots, r$. As with C_i, the set $C_{0,i}$ is the extended Parikh image of the finite automaton $\mathfrak{A}'_{0i}$ that is obtained from $\mathfrak{A}'$ as follows. We delete all non-λ-transitions, declare q_0 as the initial state, and declare $\{q_i\}$ as the set of final states. It follows that $C_{0,i}$ is effectively semi-linear.

Now, we consider the empty word. The empty word belongs to $L(\mathfrak{A}', C)$ if and only if

- there is an $i \in \{0, \ldots, r\}$ such that q_i is a final state and reachable from q_0 via a sequence λ-transitions, and
- the vector resulting from this sequence belongs to C.

These two conditions can be checked by checking whether an $i \in \{0, \ldots, r\}$ exists such that $q_i \in F$ and $C_{0,i} \cap C$ is not empty. The latter, in turn, is decidable since the intersection of the (semi-linear) sets $C_{0,i}$ and C effectively yields a semi-linear set, for which the emptiness problem is decidable (see Chapter 2). In case the empty word indeed belongs to $L(\mathfrak{A}', C)$, we have to

declare q_0 as a final state and put the zero vector into the constraint set. Hence, we can effectively define the set $\tilde{F}$ of final states as

$$\tilde{F} := \begin{cases} F \cup \{q_0\} & \text{if } q_i \in F \text{ and } C_{0,i} \cap C \neq \emptyset, \text{ for some } i \in \{0, \ldots, r\}, \\ F & \text{otherwise,} \end{cases}$$

and the set M_ε, as part of the constraint set, as

$$M_\varepsilon := \begin{cases} \{0^{n+r}\} & \text{if } q_i \in F \text{ and } C_{0,i} \cap C \neq \emptyset, \text{ for some } i \in \{0, \ldots, r\}, \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that M_ε is semi-linear since it is finite.

For a nonempty word, let $\bar{y} \otimes \bar{z}$, where $\bar{y} \in \mathbb{N}^n$ and $\bar{z} \in \mathbb{N}^r$, be the vector resulting from the run of $\tilde{\mathfrak{A}}$ on this word. The vector $\bar{y}$ represents the sum of the vectors that have been collected during the run of $\mathfrak{A}'$ on this word. The i-th component of the vector $\bar{z}$, for $i = 1, \ldots, r$, gives information whether the state q_i has been visited during this run, which, in turn, means that there might have been a loop of λ-transitions that starts and ends in q_i. Recall that the vectors that are obtained from the loops of λ-transitions at q_i are collected in the set C_i defined in (3.2) above. We 'accept' $\bar{y} \otimes \bar{z}$ if and only if the sum of $\bar{y}$ and some vectors $\bar{x}_0, \ldots, \bar{x}_r$ from the sets $C_0, \ldots, C_r$, respectively, belongs to C. Note that we may only consider loops of λ-transitions at the states that actually occur in the run of $\mathfrak{A}'$. These states are the initial state q_0 and all the states q_i with $(\bar{z})_i \neq 0$, for $i = 1, \ldots, r$. For the other states, we require that $\bar{x}_i = \bar{0}$ (note that $C_0, \ldots, C_r$ contain $\bar{0}$). Hence, we define the set M, as part of the constraint set, as

$$\begin{aligned} M := \{\, \bar{y} \otimes \bar{z} \mid\ & \bar{y} \in \mathbb{N}^n, \bar{z} \in \mathbb{N}^r \setminus \{0^r\}, \text{ and} \\ & \text{there exist some vectors } \bar{x}_0, \ldots, \bar{x}_r \text{ in } C_0, \ldots, C_r, \\ & \text{respectively, such that:} \\ & \quad \bullet\ \bar{x}_i = \bar{0} \text{ if } (\bar{z})_i = 0, \text{ for } i = 1, \ldots, r, \text{ and} \\ & \quad \bullet\ \bar{y} + \bar{x}_0 + \cdots + \bar{x}_r \in C \,\}\,. \end{aligned} \tag{3.3}$$

Note that here we only consider $\bar{z} \neq \bar{0}$ since we are dealing with nonempty words, which means that at least a state, which is not the initial state, must be visited (recall that the initial state has no ingoing transitions).

We now give a Presburger formula that defines M. Then, by Theorem 2.10, the set M is effectively semi-linear. Let $y_1, \ldots, y_n$, and $z_1, \ldots, z_r$ be variables. Intuitively, these variables correspond to the vectors $\bar{y} \in \mathbb{N}^n$ and $\bar{z} \in \mathbb{N}^r$, respectively. Analogously, the variables $x_{i,j}$ and c_j, for $j = 1, \ldots, n$, correspond to the vectors $\bar{x}_i \in \mathbb{N}^n$ and $\bar{c} \in \mathbb{N}^n$, respectively, for $i = 0, \ldots, r$.

Since the sets $C_0, \ldots, C_r$ and C are effectively semi-linear, we can effectively find some Presburger formulas, say $\varphi_0, \ldots, \varphi_r$, and φ_C, respectively, that define these sets. We simply translate the definition of M in (3.3) into the following Presburger formula

$$\begin{aligned}\varphi_M := \neg &\left(\bigwedge_{i=1}^{r} z_i = 0 \right) \\ &\wedge \exists x_{0,1} \cdots x_{0,n} \cdots x_{r,1} \cdots x_{r,n} c_1 \cdots c_n \\ &\left(\quad \bigwedge_{i=0}^{r} \varphi_i(x_{i,1}, \ldots, x_{i,n}) \right. \\ &\quad \wedge \bigwedge_{i=1}^{r} \Big(z_i = 0 \rightarrow \bigwedge_{j=1}^{n} x_{i,j} = 0 \Big) \\ &\quad \wedge \ \varphi_C(c_1, \ldots, c_n) \\ &\quad \wedge \bigwedge_{j=1}^{n} y_j + x_{0,j} + \cdots + x_{r,j} = c_j \quad \left. \right) .\end{aligned}$$

It is easy to see that the formula φ_M indeed defines M; that is, $M = [\![\varphi_M]\!]$.

Finally, we define the constraint set $\tilde{C}$ as

$$\tilde{C} := M_\varepsilon \cup M \,,$$

which is semi-linear and effectively computable since M_ε and M are. Now, it is straightforward to show that $(\tilde{\mathfrak{A}}, \tilde{C})$ indeed recognizes $h(L(\mathfrak{A}, C))$. □

Theorem 3.13 (Klaedtke and Rueß)**.** *The class $\mathcal{L}_{\mathrm{PFA}}$ is effectively closed under concatenation.*

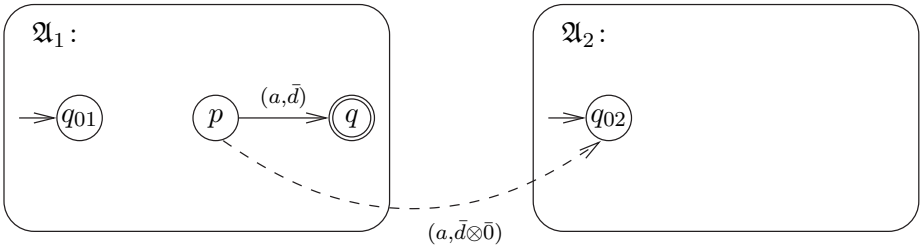
Proof. Let $(\mathfrak{A}_i, C_i)$ be PFA's of dimension $n_i \geq 1$, for $i = 1, 2$, where $\mathfrak{A}_i = (Q_i, \Sigma \times D_i, \delta_i, q_{0i}, F_i)$. Without loss of generality, we assume that the state sets Q_1 and Q_2 are disjoint.

We will construct a PFA $(\mathfrak{A}, C)$ of dimension $n_1 + n_2$ that recognizes $L(\mathfrak{A}_1, C_1)L(\mathfrak{A}_2, C_2)$ as follows. First, we simulate $\mathfrak{A}_1$, while collecting the vectors that are seen during the run of $\mathfrak{A}_1$ in the first n_1 dimensions. Whenever we are about to enter a final state, we can nondeterministically choose to enter the initial state of $\mathfrak{A}_2$ instead. Then, we simulate $\mathfrak{A}_2$, while collecting the vectors that are seen during the run of $\mathfrak{A}_2$ in the last n_2 dimensions.

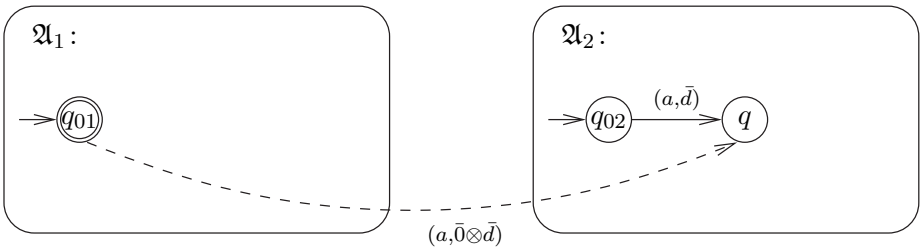
Formally, we define

$$D := (D_1 \otimes \{0^{n_2}\}) \cup (\{0^{n_1}\} \otimes D_2)$$

and $\mathfrak{A} := (Q_1 \cup Q_2, \Sigma \times D, \delta, q_{01}, F)$, where the transition function δ is defined as follows:



(a)



(b)

Figure 3.4: Concatenation of two PFA's

1. For each transition $q \in \delta_1(p,(a,\bar{d}))$ in $\mathfrak{A}_1$, we add a transition $q \in \delta(p,(a,\bar{d}\otimes 0^{n_2}))$ in $\mathfrak{A}$.
2. For each transition $q \in \delta_2(p,(a,\bar{d}))$ in $\mathfrak{A}_2$, we add a transition $q \in \delta(p,(a,0^{n_1}\otimes\bar{d}))$ in $\mathfrak{A}$.
3. For each transition $q \in \delta_1(p,(a,\bar{d}))$ in $\mathfrak{A}_1$ where q is a final state, we add a transition $q_{02} \in \delta(p,(a,\bar{d}\otimes 0^{n_2}))$ in $\mathfrak{A}$ (see Figure 3.4(a)).
4. If q_{01} is a final state of $\mathfrak{A}_1$, then for each transition $q \in \delta_2(q_{02},(a,\bar{d}))$ we add a transition $q \in \delta(q_{01},(a,0^{n_1}\otimes\bar{d}))$ in $\mathfrak{A}$ (see Figure 3.4(b)).

The set of final states is defined by

$$F := \begin{cases} F_2 \cup \{q_{01}\} & \text{if } q_{01} \in F_1 \text{ and } q_{02} \in F_2, \\ F_2 & \text{otherwise.} \end{cases}$$

Finally, the constraint set is defined by $C := C_1 \otimes C_2$. □

Theorem 3.14 (Klaedtke and Rueß)**.** *The class $\mathcal{L}_{\text{DPFA}}$ is effectively closed under complementation.*

Proof. Let $(\mathfrak{A}, C)$ be a DPFA of dimension $n \geq 1$. By Lemma 3.9, we can assume, without loss of generality, that $(\mathfrak{A}, C)$ is complete and that all the states of $\mathfrak{A}$ are final states.

Since $(\mathfrak{A}, C)$ is deterministic, for each word $u \in \Sigma^*$, there is a unique word $w \in (\Sigma \times D)^*$ with $\Psi(w) = u$, where Ψ is the Σ-projection over $\Sigma \times D$. Moreover, this unique word w belongs to $L(\mathfrak{A})$ since all the states of $\mathfrak{A}$ are final states. Consequently, the word u belongs to $L(\mathfrak{A}, C)$ if and only if $\tilde{\Phi}(w)$ belongs to C, where $\tilde{\Phi}$ is the extended Parikh mapping over $\Sigma \times D$.

To sum up, the word u belongs to $\Sigma^* \setminus L(\mathfrak{A}, C)$ if and only if $\tilde{\Phi}(w)$ belongs to $\mathbb{N}^n \setminus C$, for the unique word $w \in (\Sigma \times D)^*$ with $\Psi(w) = u$. Hence, we need only to invert the constraint set C, while leaving the automata component $\mathfrak{A}$ as is, in order to obtain the DPFA $(\mathfrak{A}, \mathbb{N}^n \setminus C)$ recognizing $\Sigma^* \setminus L(\mathfrak{A}, C)$. Note that $\mathbb{N}^n \setminus C$ is effectively semi-linear since the class of semi-linear sets is effectively closed under complementation. □

We turn to some non-closure results. In particular, we will see that the class of PFA-recognizable languages is not closed under complementation and that the class of DPFA-recognizable languages is not closed under homomorphisms.

Our first step toward these results is to find a language that is not PFA-recognizable. The crucial point in finding such a language is the following. Although the idea of Parikh automata enhances the recognition power of automata with regard to *arithmetical* properties, it does not enhance the recognition power of automata with regard to *structural* properties. A language with structural properties, for example, is the language

$$\{uu \mid u \in \{a, b\}^*\},$$

which, in fact, has been shown to be not PFA-recognizable by Klaedtke and Rueß in [KR02] (a revised version of their proof can also be found in [Kla04, pages 82–84]). With regard to some results in the next chapter, we will slightly adapt their proof to the language consisting of palindromes over $\{a, b\}$ with center marker

$$L_{\text{pal}} := \{ucu^{\text{R}} \mid u \in \{a, b\}^*\},$$

where $^{\text{R}}$ denotes the word reversal operator.

Lemma 3.15 (Klaedtke and Rueß)**.** *The language L_{pal} is not PFA-recognizable.*

Proof. Let $\Sigma := \{a, b, c\}$. Toward a contradiction we assume that the PFA $(\mathfrak{A}, C)$ of dimension $n \geq 1$ recognizes L_{pal}, where $\mathfrak{A} := (Q, \Sigma \times D, \delta, q_0, F)$.

Further, let Ψ and $\tilde{\Phi}$ be the Σ-projection and the extended Parikh mapping, respectively, over $\Sigma \times D$.

For a word $u \in \{a,b\}^*$, the pair $(q, \bar{x}) \in Q \times \mathbb{N}^n$ is called a *configuration* of u if

- there is a run of $\mathfrak{A}$ from q_0 to q, say, via the word w, for some $w \in (\Sigma \times D)^*$, such that
- $\Psi(w) = u$, and
- $\tilde{\Phi}(w) = \bar{x}$.

Note that, for the word u, there might exist more than one such configuration.

We denote by $\mathrm{Conf}(u)$ the set of the configurations of u which satisfies the following: the pair $(q, \bar{x}) \in Q \times \mathbb{N}^n$ belongs to $\mathrm{Conf}(u)$ if and only if

- there is a run of $\mathfrak{A}$ from q to a final state, say, via the word w', for some $w' \in (\Sigma \times D)^*$, such that
- $\Psi(w') = cu^{\mathrm{R}}$, and
- $\bar{x} + \tilde{\Phi}(w') \in C$.

Intuitively, the set $\mathrm{Conf}(u)$ contains those configurations of u from which onwards an accepting run on ucu^{R} exists. Thus, for any word $u \in \{a,b\}^*$, the set $\mathrm{Conf}(u)$ is nonempty since, by assumption, ucu^{R} belongs to $L(\mathfrak{A}, C)$. Moreover, for $u, v \in \{a,b\}^*$, if $u \neq v$, then $\mathrm{Conf}(u)$ and $\mathrm{Conf}(v)$ are disjoint; otherwise, the word ucv^{R} would also belong to $L(\mathfrak{A}, C)$.

For each $k \geq 0$, let Conf_k be the union of all the sets of the form $\mathrm{Conf}(u)$ with $u \in \{a,b\}^k$, that is,

$$\mathrm{Conf}_k := \bigcup_{u \in \{a,b\}^k} \mathrm{Conf}(u)\,.$$

It follows that

$$|\mathrm{Conf}_k| \geq 2^k\,, \tag{3.4}$$

for any $k \geq 0$.

On the other hand, for each $k \geq 0$, let C_k be the set of the configurations of the words of length k, that is,

$$C_k := \{(q, \bar{x}) \in Q \times \mathbb{N}^n \mid (q, \bar{x}) \text{ is a configuration of some word } u \in \{a,b\}^k\}\,.$$

By definition, we have

$$\mathrm{Conf}_k \subseteq C_k\,, \tag{3.5}$$

for each $k \geq 0$, since the elements in Conf_k are also configurations of words over $\{a,b\}$ of length k.

By finding an appropriate upper bound for $|C_k|$, we show that there is some $k_0 \geq 0$ such that $|C_{k_0}| < 2^{k_0}$, thus obtaining a contradiction.

Let $k \geq 0$. A configuration $(q, \bar{x})$ in C_k can be found as follows. We first choose a state q from the state set Q and then choose a sequence of k-many vectors from the auxiliary set D that constitutes the vector $\bar{x}$. Note that the order of the sequence of the k-many vectors does not matter since vector addition is commutative. The number of possibilities to choose a state q is given by $|Q|$. The number of possibilities to choose k-many vectors from D, where the order does not matter, can be described by the number of unordered selections of k-many vectors, possibly with repetitions, from the set D. The latter, in turn, is given by (see, for example, [Big89, pages 66–68])

$$\binom{|D| + k - 1}{k}.$$

To sum up, we have

$$\begin{aligned} |C_k| &\leq |Q| \binom{|D| + k - 1}{k} \\ &= |Q| \frac{(|D| + k - 1) \cdots (k + 1)}{(|D| - 1)!} \\ &\leq |Q| (|D| + k - 1) \cdots (k + 1) \\ &\leq |Q| (|D| + k - 1)^{|D|-1}, \end{aligned} \tag{3.6}$$

for any $k \geq 0$. Since $|Q|$ and $|D|$ are fixed for the PFA $(\mathfrak{A}, C)$, it follows that $|C_k|$ is bounded by a polynomial of degree $|D|$ in k. For this polynomial, there exists some $k_0 \geq 0$ such that

$$|Q| (|D| + k_0 - 1)^{|D|-1} < 2^{k_0}. \tag{3.7}$$

Then, for this particular k_0, we have

$$2^{k_0} \overset{(3.4)}{\leq} |\mathrm{Conf}_{k_0}| \overset{(3.5)}{\leq} |C_{k_0}| \overset{(3.6)}{\leq} |Q| (|D| + k_0 - 1)^{|D|-1} \overset{(3.7)}{<} 2^{k_0},$$

which is a contradiction. □

By showing that the complement of L_{pal} is PFA-recognizable, we obtain the non-closure of $\mathcal{L}_{\mathrm{PFA}}$ under complementation.

Theorem 3.16 (Klaedtke and Rueß)**.** *The class $\mathcal{L}_{\mathrm{PFA}}$ is not closed under complementation.*

Proof. We show that $\Sigma^* \setminus L_{\mathrm{pal}}$ is PFA-recognizable.

Let $\Sigma := \{a, b, c\}$. Clearly, a word $w \in \Sigma^*$ is not a palindrome over $\{a, b\}$ with center marker if and only if one of the following cases occurs:

1. The word w contains no c's or contains more than one c; that is, w belongs to the language
$$L_{\mathrm{cc}} := \{u \in \Sigma^* \mid |u|_c = 0 \text{ or } |u|_c > 1\}\,.$$
2. The word w contains exactly one c, but the length of the word to the left and to the right of this c are not equal; that is, w belongs to the language L_{unbal} of Example 3.6.
3. The word w has the form ucv, for some $u, v \in \{a,b\}^m$ and $m \geq 1$, but $v \neq u^{\mathrm{R}}$; that is, w belongs to the language
$$L_{\mathrm{bal}} := \{u_1 \cdots u_m c v_m \cdots v_1 \mid m \geq 1, u_1, \ldots, u_m, v_m, \ldots, v_1 \in \{a,b\}, \text{ and } u_i \neq v_i, \text{ for some } i \in \{1,\ldots,m\}\}\,.$$

In other words, $\Sigma^* \setminus L_{\mathrm{pal}} = L_{\mathrm{cc}} \cup L_{\mathrm{unbal}} \cup L_{\mathrm{bal}}$. Since, by Theorem 3.10, the class $\mathcal{L}_{\mathrm{PFA}}$ is closed under union, it suffices to show that all the languages L_{cc}, L_{unbal}, and L_{bal} are PFA-recognizable.

The language L_{cc} is regular; for instance, it is definable by the regular expression
$$(a+b)^* + (a+b)^* c (a+b)^* c (a+b+c)^*\,.$$
By Remark 3.4, it follows that L_{cc} is DPFA-recognizable, and thus also PFA-recognizable.

In Example 3.6 we have already seen that L_{unbal} is DPFA-recognizable and thus also PFA-recognizable.

For L_{bal}, we construct a PFA $(\mathfrak{A}_{\mathrm{bal}}, C_{\mathrm{bal}})$ of dimension 4 that works as follows. On an input word $w \in \Sigma^*$, we first use $\mathfrak{A}_{\mathrm{bal}}$ to ensure that w has the form ucv with $u, v \in \{a,b\}^+$. To ensure that $|u| = |v|$, we use the first and the second component of the dimensions. Prior to reading c we increment the first component, and afterwards the second one. The constraint set then should only contain those vectors whose first and second component are equal. Now that $|u| = |v|$, we can assume that $u = u_1 \cdots u_m$ and $v = v_m \cdots v_1$, for some $m \geq 1$. We now guess a position $i \in \{1,\ldots,m\}$, for which u_i differs from v_i, by using the third and the fourth component of the dimensions. Prior to reading u_i we increment the third component, and after reading v_i we increment the fourth one. Again, the constraint set then should only contain those vectors whose third and fourth component are equal (and less than the first and the second one, of course). Formally, let $\mathfrak{A}_{\mathrm{bal}}$ be the finite automaton depicted in Figure 3.5, and let
$$C_{\mathrm{bal}} := \{k_1(1,1,1,1) + k_2(1,1,0,0) \mid k_1, k_2 \in \mathbb{N}\}\,.$$
Then, it is straightforward to show that $(\mathfrak{A}_{\mathrm{bal}}, C_{\mathrm{bal}})$ indeed recognizes L_{bal}. □

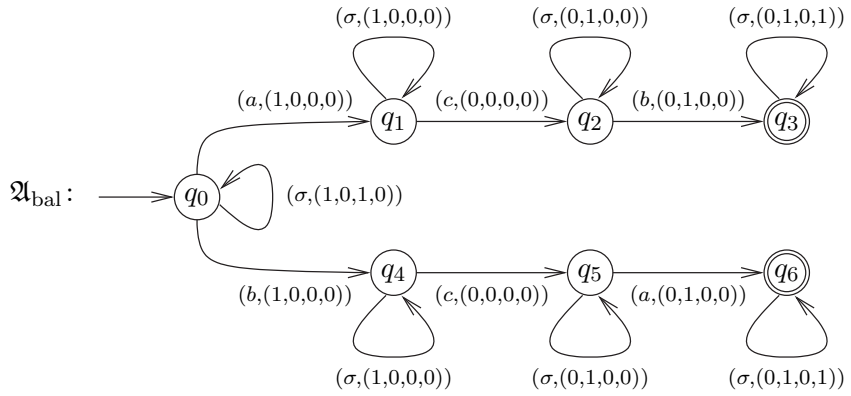


Figure 3.5: Automaton of Theorem 3.16 ($\sigma \in \{a, b\}$)

Now, using a slight modification of L_{bal}, we show that the class of DPFA-recognizable languages is not closed under homomorphisms.

Theorem 3.17 (Klaedtke and Rueß)**.** *The class $\mathcal{L}_{\text{DPFA}}$ is not closed under homomorphisms.*

Proof. We show that there are a DPFA-recognizable language and a homomorphism such that the image of this language under this homomorphism is not DPFA-recognizable.

Let $\Pi := \{a, b, c, \hat{a}, \hat{b}, \tilde{a}, \tilde{b}\}$ be an alphabet, and let

$$\begin{aligned}\tilde{L}_{\text{bal}} := \{u_1 \cdots u_m c v_m \cdots v_1 \mid\ & m \geq 1 \text{ and there is some } i \in \{1, \ldots, m\} \\ & \text{such that } u_1, \ldots, u_{i-1}, v_{i-1}, \ldots, v_1 \in \{a, b\}, \\ & u_i, v_i \in \{\hat{a}, \hat{b}\}, u_i \neq v_i, \text{ and} \\ & u_{i+1}, \ldots, u_m, v_m, \ldots, v_{i+1} \in \{\tilde{a}, \tilde{b}\}\}\,.\end{aligned}$$

The language $\tilde{L}_{\text{bal}}$, in fact, is merely a slight modification of the language L_{bal} of Theorem 3.16 above. Anyhow, the new symbols $\hat{a}, \hat{b}, \tilde{a}, \tilde{b}$ make it possible for a PFA, whose automata component is similar to $\mathfrak{A}_{\text{bal}}$ above, to work deterministically; these symbols mark the position i that has to be guessed nondeterministically in the proof above. Formally, let $\tilde{\mathfrak{A}}_{\text{bal}}$ be the (deterministic) finite automaton depicted in Figure 3.6. For the constraint set, we take C_{bal} of the proof above. Then, $(\tilde{\mathfrak{A}}_{\text{bal}}, C_{\text{bal}})$ is a DPFA that recognizes $\tilde{L}_{\text{bal}}$.

Now, let $h\colon \Pi^* \to \Sigma^*$ be a homomorphism where $h(a) = h(\hat{a}) = h(\tilde{a}) = a$, $h(b) = h(\hat{b}) = h(\tilde{b}) = b$, and $h(c) = c$. Then, it is not difficult to verify that L_{bal} is the image of $\tilde{L}_{\text{bal}}$ under h. Hence, our proof is complete once we have shown that L_{bal} is not DPFA-recognizable.

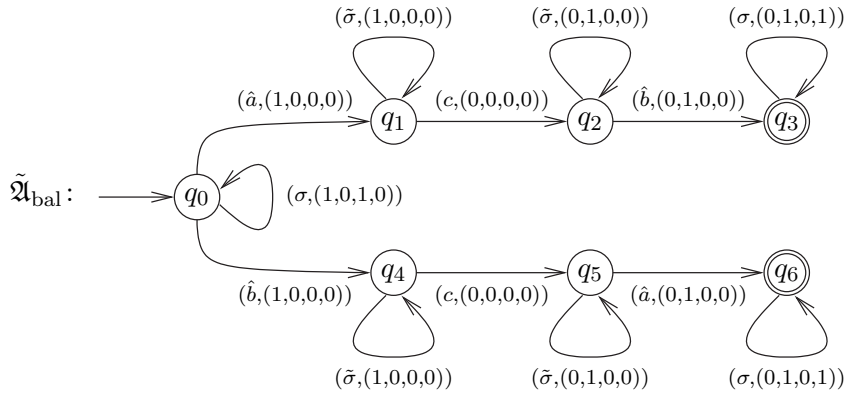


Figure 3.6: Automaton of Theorem 3.17 ($\sigma \in \{a, b\}$)

Toward a contradiction, we now assume that L_{bal} is DPFA-recognizable. In the proof of Theorem 3.16 we have already noted that L_{cc} and L_{unbal} are also DPFA-recognizable. By the closure of $\mathcal{L}_{\text{DPFA}}$ under union (see Theorem 3.10) and under complementation (see Theorem 3.14), it follows that L_{pal} is also DPFA-recognizable, whereupon we obtain a contradiction since L_{pal} is not even PFA-recognizable (see Lemma 3.15). □

3.2 Decision Problems

In this section we investigate some decision problems for PFA's (and DPFA's). In particular, we will look at the membership problem, the emptiness problem, and the equivalence problem.

The membership problem for PFA's is the question:

> Given a PFA $(\mathfrak{A}, C)$ of dimension $n \geq 1$ over Σ, with auxiliary set D, and a word $u \in \Sigma^*$, does u belong to $L(\mathfrak{A}, C)$?

The following procedure, clearly, determines whether u belongs to $L(\mathfrak{A}, C)$:

1. If $u = \varepsilon$ then check whether ε belongs to $L(\mathfrak{A})$ and whether $\bar{0}$ belongs to C. If both conditions are true, answer 'YES.' Otherwise, answer 'NO.'
2. If $u = u_1 \cdots u_m$, for some $m \geq 1$, then do the following, for each sequence $\bar{d}_1, \ldots, \bar{d}_m$ of vectors from the auxiliary set D.
   a) Check whether $w := (u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$ belongs to $L(\mathfrak{A})$.
   b) If w does not belong to $L(\mathfrak{A})$, then go back to step 2 and proceed with the next sequence of vectors.

c) If $\bar{d}_1 + \ldots + \bar{d}_m$ belongs to C, answer 'YES.' Otherwise, go back to step 2 and proceed with the next sequence of vectors.

3. If we have tried all possible sequences of vectors, then answer 'NO.'

Note that the conditions in step 1 and step 2(a) are decidable since the membership problem for finite automata is decidable, and that the condition in step 2(c) is decidable since the membership problem for semi-linear sets is decidable (see Section 2.2). Further, the procedure above terminates since D is finite. Hence, we obtain the decidability of the membership problem for PFA's.

Theorem 3.18. *The membership problem for PFA's is decidable.*

The emptiness problem for PFA's is the question

Given a PFA $(\mathfrak{A}, C)$ of dimension $n \geq 1$, is $L(\mathfrak{A}, C)$ empty?

Our approach to the emptiness problem for PFA's will be, first, to show that the extended Parikh image of any regular language is semi-linear. More generally, we show that the extended Parikh image of any language whose Parikh image is semi-linear is again semi-linear. Then, using the closure properties of semi-linear sets and the decidability of the emptiness problem for semi-linear sets (see Section 2.2), we obtain a decision procedure for the emptiness problem for PFA's.

Lemma 3.19 (Klaedtke and Rueß)**.** *Let Σ be an alphabet and let D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. If the Parikh image of a language $L \subseteq (\Sigma \times D)^*$ is (effectively) semi-linear, then so is its extended Parikh image.*

Proof. Since Σ and D are both finite, we can enumerate the elements of the extended alphabet $\Sigma \times D$ as $\{(a_1, \bar{d}_1), \ldots, (a_m, \bar{d}_m)\}$, where $m := |\Sigma| \cdot |D|$.

Suppose that the Parikh image of L is a semi-linear set of vectors of dimension m, say

$$\Phi(L) = \bigcup_{i=1}^{r} A_i \,, \tag{3.8}$$

for some linear sets $A_1, \ldots, A_r \subseteq \mathbb{N}^m$, for some $r \geq 0$. Without loss of generality, we can assume that each of the linear sets $A_1, \ldots, A_r$ has $s \geq 0$ periods, for a fixed natural number s. Otherwise, we can modify the linear sets constituting $\Phi(L)$ as follows, without affecting the set $\Phi(L)$ itself. Suppose that $s_i \geq 0$ is the number of the periods of A_i, for $i = 1, \ldots, r$. Let $s := \max_{1 \leq i \leq r} s_i$. If $s_i < s$ for some i, then we replace A_i with A_i' that is obtained from A_i by adding $(s - s_i)$-many zero vectors as periods. Clearly, $A_i' = A_i$, for

each $i = 1, \ldots, r$. Hence, we have obtained a representation of $\Phi(L)$ that meets the requirement in our assumption. Under this assumption, now suppose that

$$A_i = \{\bar{x}_{i0} + k_1\bar{x}_{i1} + \cdots + k_s\bar{x}_{is} \mid k_1, \ldots, k_s \in \mathbb{N}\}\,,$$

for each $i = 1, \ldots, r$.

For any word $w \in (\Sigma \times D)^*$, the extended Parikh image of w, by definition, is the sum of those vectors among $\bar{d}_1, \ldots, \bar{d}_m$ that occur in w. Moreover, since vector addition is commutative, only the number of occurrences of these vectors matters. Thus, we have

$$\tilde{\Phi}(w) = |w|_{(a_1,\bar{d}_1)}\bar{d}_1 + \cdots + |w|_{(a_m,\bar{d}_m)}\bar{d}_m\,,$$

where $|w|_{(a_j,\bar{d}_j)}$ denotes the number of occurrences of $(a_j, \bar{d}_j) \in \Sigma \times D$ in w, for $j = 1, \ldots, m$. On the other hand, the j-th component of $\Phi(w)$, the Parikh image of w, by definition, represents $|w|_{(a_j,\bar{d}_j)}$, for $j = 1, \ldots, m$. Replacing the latter term with the former, we obtain

$$\tilde{\Phi}(w) = (\Phi(w))_1\bar{d}_1 + \cdots + (\Phi(w))_m\bar{d}_m\,, \tag{3.9}$$

for each $w \in (\Sigma \times D)^*$.

Now, we consider the extended Parikh image of L, which is a set of vectors of dimension n given by

$$\tilde{\Phi}(L) = \{\tilde{\Phi}(w) \mid w \in L\}\,.$$

Using (3.9) and the Parikh image of L, we obtain

$$\begin{aligned}\tilde{\Phi}(L) &= \{(\Phi(w))_1\bar{d}_1 + \cdots + (\Phi(w))_m\bar{d}_m \mid w \in L\} \\ &= \{(\bar{x})_1\bar{d}_1 + \cdots + (\bar{x})_m\bar{d}_m \mid \bar{x} \in \Phi(L)\}\,.\end{aligned}$$

Toward a representation of $\tilde{\Phi}(L)$ as a semi-linear set, we decompose $\Phi(L)$ into the linear sets $A_1, \ldots, A_r$ of (3.8) and do some transformations, obtaining

$$\begin{aligned}\tilde{\Phi}(L) &= \bigcup_{i=1}^{r}\{(\bar{x})_1\bar{d}_1 + \cdots + (\bar{x})_m\bar{d}_m \mid \bar{x} \in A_i\} \\ &= \bigcup_{i=1}^{r}\{(\bar{x})_1\bar{d}_1 + \cdots + (\bar{x})_m\bar{d}_m \\ &\qquad \mid \bar{x} = \bar{x}_{i0} + k_1\bar{x}_{i1} + \cdots + k_s\bar{x}_{is} \text{ and } k_1, \ldots, k_s \in \mathbb{N}\} \\ &= \bigcup_{i=1}^{r}\{[(\bar{x}_{i0})_1 + k_1(\bar{x}_{i1})_1 + \cdots + k_s(\bar{x}_{is})_1]\bar{d}_1 \\ &\qquad + \cdots \\ &\qquad + [(\bar{x}_{i0})_m + k_1(\bar{x}_{i1})_m + \cdots + k_s(\bar{x}_{is})_m]\bar{d}_m \\ &\qquad \mid k_1, \ldots, k_s \in \mathbb{N}\}\end{aligned}$$

$$
\begin{aligned}
= \bigcup_{i=1}^{r} \{ & [(\bar{x}_{i0})_1 \bar{d}_1 + \cdots + (\bar{x}_{i0})_m \bar{d}_m] \\
& + k_1[(\bar{x}_{i1})_1 \bar{d}_1 + \cdots + (\bar{x}_{i1})_m \bar{d}_m] \\
& + \cdots \\
& + k_s[(\bar{x}_{is})_1 \bar{d}_1 + \cdots + (\bar{x}_{is})_m \bar{d}_m] \\
& \mid k_1, \ldots, k_s \in \mathbb{N} \} \,.
\end{aligned}
$$

By defining

$$
\bar{y}_{ij} := (\bar{x}_{ij})_1 \bar{d}_1 + \cdots + (\bar{x}_{ij})_m \bar{d}_m
$$

as generators, for $i = 1, \ldots, r$ and $j = 0, \ldots, s$, we obtain a representation of $\tilde{\Phi}(L)$ as a finite union of linear sets

$$
\tilde{\Phi}(L) = \bigcup_{i=1}^{r} \{ \bar{y}_{i0} + k_1 \bar{y}_{i1} + \cdots + k_s \bar{y}_{is} \mid k_1, \ldots, k_s \in \mathbb{N} \} \,,
$$

thus justifying the semi-linearity of the extended Parikh image of L. □

Theorem 3.20 (Klaedtke and Rueß)**.** *The emptiness problem for PFA's is decidable.*

Proof. Let $(\mathfrak{A}, C)$ be a PFA of dimension $n \geq 1$ over Σ with the auxiliary set D. Further, let Ψ and $\tilde{\Phi}$ be the Σ-projection and the extended Parikh mapping, respectively.

Obviously, $L(\mathfrak{A}) \subseteq (\Sigma \times D)^*$ is regular, and thus, by Parikh's theorem, $\Phi(L(\mathfrak{A}))$ is effectively semi-linear. Then, by Lemma 3.19, $\tilde{\Phi}(L(\mathfrak{A}))$ is effectively semi-linear as well. We will show that

$$
L(\mathfrak{A}, C) \neq \emptyset \quad \text{if and only if} \quad \tilde{\Phi}(L(\mathfrak{A})) \cap C \neq \emptyset \,. \tag{3.10}
$$

The right-hand side of (3.10) is decidable since the intersection of two semi-linear sets effectively yields a semi-linear set, for which the emptiness problem is decidable. Hence, the left-hand side of (3.10), that is, the emptiness problem for $(\mathfrak{A}, C)$, is decidable.

For the 'only if' part of (3.10), suppose that $u \in L(\mathfrak{A}, C)$, for some $u \in \Sigma^*$. By definition, there is some $w \in L(\mathfrak{A})$ such that $\Psi(w) = u$ and $\tilde{\Phi}(w) \in C$. For this particular w, obviously, $\tilde{\Phi}(w) \in \tilde{\Phi}(L(\mathfrak{A}))$, and together with $\tilde{\Phi}(w) \in C$, we thus have found an element in $\tilde{\Phi}(L(\mathfrak{A})) \cap C$.

Conversely, suppose that $\tilde{\Phi}(L(\mathfrak{A})) \cap C$ is not empty. An element of this set, certainly, must have the form $\tilde{\Phi}(w)$, for some $w \in L(\mathfrak{A})$. Moreover, for this particular w, we have $\tilde{\Phi}(w) \in C$. Thus, by definition, we have $\Psi(w) \in L(\mathfrak{A}, C)$, whereupon we conclude that $L(\mathfrak{A}, C)$ is not empty. □

As noted in Remark 2.13 and at the end of Chapter 2, another approach to the emptiness problem for a language is to show that the Parikh image of this language is effectively semi-linear. Then, the decidability of the emptiness problem for this language follows from the decidability of the emptiness problem for semi-linear sets. The following theorem shows that this approach can also be applied to PFA-recognizable languages; we show that the Parikh images of PFA-recognizable languages are effectively semi-linear. The proof of this result is based on the fact that the Parikh mapping can somehow be 'simulated' by the extended Parikh mapping.

Theorem 3.21. *The Parikh image of any PFA-recognizable language is effectively semi-linear.*

Proof. Let $(\mathfrak{A}, C)$ be a PFA of dimension $n \geq 1$ over $\Sigma := \{a_1, \ldots, a_m\}$, where $\mathfrak{A} = (Q, \Sigma \times D, \delta, q_0, F)$.

We extend $(\mathfrak{A}, C)$ to a new PFA $(\mathfrak{B}, C')$ of dimension $n+m$ that recognizes the same language as $(\mathfrak{A}, C)$ and counts the number of occurrences of the input symbols by using the additional dimensions. The idea is to associate each transition in which a_i is involved with the unit vector $\bar{e}_i$, for $i = 1, \ldots, m$.

Formally, we define $\mathfrak{B} := (Q, \Sigma \times D', \delta', q_0, F)$ as follows. The auxiliary set D' is given by

$$D' := D \otimes \{\bar{e}_1, \ldots, \bar{e}_m\},$$

where $\bar{e}_i$ is the i-th unit vector in $\mathbb{N}^m$, for $i = 1, \ldots, m$. The transition function δ' is defined by

$$\delta'(q, (a_i, \bar{d} \otimes \bar{e}_j)) := \begin{cases} \delta(q, (a_i, \bar{d})) & \text{if } i = j, \\ \emptyset & \text{otherwise,} \end{cases}$$

for each $q \in Q$, $\bar{d} \in D$, and $i, j \in \{1, \ldots, m\}$.

Note that we may not put any constraints on the last m components of the dimensions, since these should only be used to count the number of occurrences of the input symbols. Thus, we define the constraint set to be $C' := C \otimes \mathbb{N}^m$, which is obviously semi-linear.

It should be clear from the construction that $L(\mathfrak{B}, C') = L(\mathfrak{A}, C)$.

To avoid confusion, in the following we will use $\tilde{\Phi}_{\mathfrak{A}}$ to denote the extended Parikh mapping with respect to $\Sigma \times D$ and $\tilde{\Phi}_{\mathfrak{B}}$ to denote the extended Parikh mapping with respect to $\Sigma \times D'$.

By construction, a vector in $\tilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$, in the first n components, contains a vector in $\tilde{\Phi}_{\mathfrak{A}}(L(\mathfrak{A}))$ and, in the last m components, contains the number of occurrences of the input symbols in a word that belongs to $L(\mathfrak{A})$. Still, $\tilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$ may contain vectors that are the extended Parikh images of the

words whose Σ-projections do not belong to $L(\mathfrak{B}, C')$; we remove these vectors by intersecting $\tilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$ with C'. Then, we obtain $\Phi(L(\mathfrak{B}, C'))$, the Parikh image of $L(\mathfrak{B}, C')$ (and thus also of $L(\mathfrak{A}, C)$), by removing the first n components of $\tilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B})) \cap C'$. Formally, we have

$$\Phi(L(\mathfrak{B}, C')) = p_1^{1+m}(p_1^{2+m}(\cdots(p_1^{n+m}(\tilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B})) \cap C'))\cdots))\,,$$

where $p_1^{j+m}\colon \mathbb{N}^{j+m} \to \mathbb{N}^{j+m-1}$ is the first projection function (see Section 2.2) over vectors of dimension $j+m$, for $j = 1, \ldots, n$.

By Lemma 3.19, the set $\tilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$ is effectively semi-linear. Hence, by the fact that the class of semi-linear sets is effectively closed under intersection and under the projection functions, the set $\Phi(L(\mathfrak{B}, C'))$ is effectively semi-linear as well, whereupon we conclude that the Parikh image of $L(\mathfrak{A}, C)$ is effectively semi-linear. □

The equivalence problem for PFA's is the question

> Given two PFA's $(\mathfrak{A}_1, C_1), (\mathfrak{A}_2, C_2)$ of dimension $n_1, n_2 \geq 1$, respectively, is $L(\mathfrak{A}_1, C_1) = L(\mathfrak{A}_2, C_2)$?

Regarding the previous two problems, the classes $\mathcal{L}_{\mathrm{PFA}}$ and $\mathcal{L}_{\mathrm{DPFA}}$ do not differ from each other; the decision procedures for PFA's can be used for DPFA's as well. Regarding the equivalence problem, by contrast, these two classes differ from each other; while this problem is still decidable for DPFA's, it is no longer decidable for PFA's.

Theorem 3.22. *The equivalence problem for DPFA's is decidable.*

Proof. Given two DPFA's $(\mathfrak{A}_1, C_1), (\mathfrak{A}_2, C_2)$ of dimension $n_1, n_2 \geq 1$, respectively, we have $L(\mathfrak{A}_1, C_1) = L(\mathfrak{A}_2, C_2)$ if and only if

$$(L(\mathfrak{A}_1, C_1) \cap (\Sigma^* \setminus L(\mathfrak{A}_2, C_2))) \cup (L(\mathfrak{A}_2, C_2) \cap (\Sigma^* \setminus L(\mathfrak{A}_1, C_1))) = \emptyset\,. \quad (3.11)$$

Since $\mathcal{L}_{\mathrm{DPFA}}$ is closed under the Boolean operations (see Section 3.1), and since the emptiness problem for DPFA's is decidable (Theorem 3.20), it is decidable whether (3.11) holds and thus also whether $L(\mathfrak{A}_1, C_1) = L(\mathfrak{A}_2, C_2)$. □

We show the undecidability of the equivalence problem for PFA's by a reduction of the universality problem for PFA's to the equivalence problem for PFA's. The universality problem for PFA's is the question

> Given a PFA $(\mathfrak{A}, C)$ of dimension $n \geq 1$, is $L(\mathfrak{A}, C) = \Sigma^*$?

The undecidability of this problem, in turn, follows from a reduction of the membership problem for Turing machines.

Theorem 3.23 (Klaedtke and Rueß)**.** *The universality problem for PFA's is undecidable.*

Proof. We roughly describe a reduction of the (non-)membership problem for Turing machines, which is known to be undecidable, to the universality problem for PFA's. In the following, we briefly describe how we can code the computations of a Turing machine on an input word. For an introduction to Turing machines, the reader is referred to [HU79].

Let $\mathfrak{M}$ be a Turing machine with the state set Q and the working alphabet Σ. Without loss of generality, we assume that Q and Σ are disjoint. A configuration of $\mathfrak{M}$ is coded by a word $\kappa = uqav \in \Sigma^* Q \Sigma^+$ where $u \in \Sigma^*$, $q \in Q$, $a \in \Sigma$, and $v \in \Sigma^*$ represent the word to the left of the input head, the current state, the symbol at the input head, and the word to the right of the input head, respectively. A computation of $\mathfrak{M}$ is coded by a sequence $\kappa_1 \$ \kappa_2 \$ \cdots \$ \kappa_m$ of configurations where \$ is a new symbol that appears neither in Σ nor in Q. Without loss of generality, we assume that any accepting computation of $\mathfrak{M}$ consists of at least one step; that is, $m \geq 2$.

Now, given an input word w for $\mathfrak{M}$, we define the language $L_{\mathfrak{M},w}$, which contains those words over $\Sigma \cup Q \cup \{\$\}$ that do not code an accepting computation of $\mathfrak{M}$ on w, as

$$L_{\mathfrak{M},w} := L_1 \cup L_2 \cup L_3 \cup L_4 \cup L_5 \,,$$

where $L_1, \ldots, L_5 \subseteq (\Sigma \cup Q \cup \{\$\})^*$ are described by the following:

- The language L_1 contains the words that do not have \$ as an infix.
- The language L_2 contains the words of the form $\kappa_1 \$ \cdots \$ \kappa_m$, for some $m \geq 1$, where $\kappa_i \in (\Sigma \cup Q)^*$, for $i = 1, \ldots, m$, but at least one of them does not code a configuration of $\mathfrak{M}$.
- The language L_3 contains the words of the form $\kappa_1 \$ \cdots \$ \kappa_m$, for some $m \geq 1$, where $\kappa_1 \in (\Sigma \cup Q)^*$ does not code the initial configuration of $\mathfrak{M}$ on w.
- The language L_4 contains the words of the form $\kappa_1 \$ \cdots \$ \kappa_m$, for some $m \geq 1$, where $\kappa_m \in (\Sigma \cup Q)^*$ does not code an accepting configuration (that is, a configuration with a final state) of $\mathfrak{M}$.
- The language L_5 contains the words of the form $\kappa_1 \$ \cdots \$ \kappa_m$, for some $m \geq 1$, where an $i \in \{1, \ldots, m-1\}$ exists such that $\kappa_{i+1} \in (\Sigma \cup Q)^*$ is not a successor configuration of $\kappa_i \in (\Sigma \cup Q)^*$.

It is straightforward to construct finite automata for the languages $L_1, \ldots, L_4$. Thus, by Remark 3.4, these languages are also PFA-recognizable. The

language L_5 can be recognized by a PFA similar to the one in the proof of Theorem 3.16; the corresponding PFA first guesses an $i \in \{1, \ldots, m-1\}$ for which the configuration κ_{i+1} is not the successor configuration of κ_i and then guesses a position in κ_i and κ_{i+1} that gives evidence of the fact that the successor relation is violated.

By Theorem 3.10, $\mathcal{L}_{\mathrm{PFA}}$ is closed under union, and thus, $L_{\mathfrak{M},w}$ is PFA-recognizable. It remains to show that $\mathfrak{M}$ does not accept w if and only if $L_{\mathfrak{M},w} = (\Sigma \cup Q \cup \{\$\})^*$.

For the first direction, suppose that $\mathfrak{M}$ does not accept w. Then, there is no accepting computation of $\mathfrak{M}$ on w. In other words, any word over $\Sigma \cup Q \cup \{\$\}$ does not code an accepting computation of $\mathfrak{M}$ on w. Therefore, $L_{\mathfrak{M},w} = (\Sigma \cup Q \cup \{\$\})^*$.

For the opposite direction, suppose that $L_{\mathfrak{M},w} = (\Sigma \cup Q \cup \{\$\})^*$. That is, each word over $\Sigma \cup Q \cup \{\$\}$ does not code an accepting computation of $\mathfrak{M}$ on w, whereupon we conclude that $\mathfrak{M}$ does not accept w. □

Corollary 3.24. *The equivalence problem for PFA's is undecidable.*

Proof. Toward a contradiction, we assume that there is a decision procedure for the equivalence problem. Then, given a PFA $(\mathfrak{A}, C)$ of dimension $n \geq 1$, we can apply this procedure to answer the question "$L(\mathfrak{A}, C) = \Sigma^*$?" since the language Σ^* is obviously PFA-recognizable, thus solving the universality problem. Contradiction. □

Summary and Concluding Remarks

In this chapter, we surveyed the main idea of Parikh automata on words, which were introduced by Klaedtke and Rueß [KR02, KR03].

In Section 3.1 we applied this idea to finite automata and semi-linear sets, thereby obtaining the automata models PFA and DPFA and the corresponding language classes $\mathcal{L}_{\mathrm{PFA}}$ and $\mathcal{L}_{\mathrm{DPFA}}$. We studied some closure properties of these language classes. We saw that the class $\mathcal{L}_{\mathrm{PFA}}$ is closed under intersection, union, inverse homomorphisms, homomorphisms, and concatenation, but not under complementation. The class $\mathcal{L}_{\mathrm{DPFA}}$, on the other hand, is closed under intersection, union, inverse homomorphisms, and complementation, but not under homomorphisms.

In addition to these closure properties, Klaedtke and Rueß [KR02] showed the closure of $\mathcal{L}_{\mathrm{PFA}}$ under left and right quotient.

In Section 3.2 we studied the membership, the emptiness, and the equivalence problem for PFA's and DPFA's. We presented decision procedures for

solving the former two problems for PFA's (and DPFA's). Further, the decidability of the equivalence problem for DPFA's follows immediately from the closure properties of $\mathcal{L}_{\mathrm{DPFA}}$. By contrast, we saw that the universality problem (and thus also the equivalence problem) for PFA's is undecidable, which was shown by means of a reduction of the membership problem for Turing machines. In addition, we showed that the Parikh images of PFA-recognizable languages are effectively semi-linear.

The method with which the emptiness problem for PFA's has been solved is remarkable: we have solved this problem by means of a reduction to the emptiness problem for the intersection of semi-linear sets. For one thing, it indicates the application of semi-linear sets in solving decision problems in formal language theory once again. For another thing, it demonstrates a useful interplay between the closure and decision properties of semi-linear sets in solving decision problems in formal language theory.

As noted at the beginning of Section 3.1, a PFA can be viewed as a finite automaton with monotonic counters which, at the end of a computation, may check the values of the counters against a given constraint. This view suggests the connection between PFA's and counter machines. In fact, Klaedtke and Rueß [KR02] showed that PFA's are equivalent to the so-called 'reversal-bounded counter machines' (that is, counter machines where the number of the alternations between increasing and decreasing modes of the counters is bounded), which were introduced by Ibarra [Iba78].

Chapter 4

Parikh Pushdown Automata

In this chapter, we extend the notion of Parikh automata by using pushdown automata, instead of finite automata, for the automata component, while still using semi-linear constraint sets.

Actually, this extension is almost obvious. Parikh's theorem, which has been used quite heavily to show that the emptiness problem for PFA's is decidable, applies not only to regular languages (characterized by finite automata), but also to context-free languages (characterized by pushdown automata). Hence, it is not surprising that the decidability of the emptiness problem for PFA's carries over to this extension, as already noted by Klaedtke and Rueß in [KR02].

In the following section, we introduce this extension formally, and as with PFA's, we investigate some closure properties. In Section 4.2 we will see that for some decision problems, in particular for the emptiness problem, the results of the preceding chapter carry over in a straightforward manner. We conclude this chapter by a comparison of the automata models defined in the preceding and in the present chapter with the automata models in the well-known Chomsky hierarchy.

4.1 Parikh Pushdown Automata

Definition 4.1. A *Parikh pushdown automaton* (*PPDA*) of dimension $n \geq 1$ over an alphabet Σ is a system $(\mathfrak{A}, C)$ where $\mathfrak{A} := (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$ is a pushdown automaton over $\Sigma \times D$, for some finite, nonempty set $D \subseteq \mathbb{N}^n$ (called the *auxiliary set*), and $C \subseteq \mathbb{N}^n$ is a semi-linear set (called the *constraint set*). A *deterministic Parikh pushdown automaton* (*DPPDA*) of dimension $n \geq 1$ over an alphabet Σ is a PPDA $(\mathfrak{A}, C)$ of dimension n over Σ where the transition function δ satisfies $\sum_{\bar{d} \in D} |\delta(q, (a, \bar{d}), Z)| + |\delta(q, \varepsilon, Z)| \leq 1$, for

each $q \in Q$, $a \in \Sigma$, and $Z \in \Gamma$. In other words, $\mathfrak{A}$ can only proceed either by reading $(a, \bar{d})$, for at most one vector $\bar{d} \in D$, or by reading no input symbols.

The language *recognized* by $(\mathfrak{A}, C)$ is defined as $L(\mathfrak{A}, C) := L(\mathfrak{A})\restriction_C$, where $L(\mathfrak{A}) \subseteq (\Sigma \times D)^*$ is the language recognized by $\mathfrak{A}$.

A language L over Σ is called *PPDA-recognizable* or *DPPDA-recognizable* if there is a PPDA or DPPDA, respectively, $(\mathfrak{A}, C)$ of dimension n over Σ, for some $n \geq 1$, such that $L(\mathfrak{A}, C) = L$. The class of PPDA-recognizable and DPPDA-recognizable languages over Σ are denoted by $\mathcal{L}_{\mathrm{PPDA}}(\Sigma)$ and $\mathcal{L}_{\mathrm{DPPDA}}(\Sigma)$, respectively. We omit the reference to Σ whenever Σ is clear from the context.

Note that in the definition above we restrict ourselves to the acceptance by final state in order to avoid complications in the definition of deterministic Parikh pushdown automata; already for deterministic pushdown automata, the acceptance by empty stack is less expressive than the acceptance by final state, with respect to recognized languages.

As with PFA's, a PPDA of dimension $n \geq 1$ can be seen as a finite automaton that is augmented with n-many monotonic counters and a pushdown stack. In addition, a PPDA may make ε-transitions (sometimes also called ε-moves). Still, we should bear in mind that a PPDA may only increment its counters if it reads an input symbol. In other words, a PPDA may not increment its counters while making an ε-transition.

As with PFA's (see Remark 3.4), this view of PPDA's also leads us to the fact that every deterministic pushdown automaton can be simulated by a DPPDA and that every pushdown automaton can be simulated by a PPDA, which we state in the following remark.

Remark 4.2. Given a (deterministic) pushdown automaton, we can construct a (deterministic) PPDA that recognizes the same language as the given automaton.

Example 4.3. We consider the language $L := \{uu^{\mathrm{R}} \in \{a, b\}^* \mid |u|_a = |u|_b\}$ of palindromes of even length where the number of occurrences of a's and b's are equal.

The following PPDA $(\mathfrak{A}, C)$ of dimension 2 recognizes L. Let $D := \{(1, 0), (0, 1)\} \subseteq \mathbb{N}^2$ be an auxiliary set, and let $\mathfrak{A} := (Q, \{a, b\} \times D, \Gamma, \delta, q_0, \bot, F)$ be a pushdown automaton, where $Q := \{q_0, q_1, q_2\}$, $\Gamma := \{\bot, A, B\}$, $F := \{q_2\}$, and δ is defined by

$$\begin{aligned} \delta(q_0, (a, (1,0)), Z) &= \{(q_0, AZ)\} & \delta(q_1, (a, (1,0)), A) &= \{(q_1, \varepsilon)\} \\ \delta(q_0, (b, (0,1)), Z) &= \{(q_0, BZ)\} & \delta(q_1, (b, (0,1)), B) &= \{(q_1, \varepsilon)\} \\ \delta(q_0, \varepsilon, Z) &= \{(q_1, Z)\} & \delta(q_1, \varepsilon, \bot) &= \{(q_2, \varepsilon)\} \end{aligned}$$

for each $Z \in \Gamma$. Further, let $C := \{k(1,1) \mid k \in \mathbb{N}\}$ be the constraint set.

Intuitively, the pushdown stack is used to record the word u that has been seen during a computation (in reverse order) while the dimensions are used to count the occurrences of a's and b's.

We turn to showing some closure properties of $\mathcal{L}_{\mathrm{PPDA}}$ and $\mathcal{L}_{\mathrm{DPPDA}}$. As with PFA's, the proofs below are effective and are adapted from standard techniques of automata theory.

Theorem 4.4. *The class $\mathcal{L}_{\mathrm{PPDA}}$ is effectively closed under union.*

Proof. Let $(\mathfrak{A}_i, C_i)$ be PPDA's of dimension $n_i \geq 1$, for $i = 1, 2$, where $\mathfrak{A}_i = (Q_i, \Sigma \times D_i, \Gamma_i, \delta_i, q_{0i}, \bot_i, F_i)$. Without loss of generality, we assume that Q_1, Q_2, Γ_1, and Γ_2 are pairwise disjoint.

We construct a PPDA $(\mathfrak{A}, C)$ of dimension $n_1 + n_2 + 2$ that recognizes $L(\mathfrak{A}_1, C_1) \cup L(\mathfrak{A}_2, C_2)$. The PPDA $(\mathfrak{A}, C)$ works as follows. On an input word $w \in \Sigma^*$, we first nondeterministically guess (by means of ε-transitions) whether w belongs to $L(\mathfrak{A}_1, C_1)$ or $L(\mathfrak{A}_2, C_2)$. Then, we simulate the PPDA that corresponds to our guess on the word w. During the simulation of $\mathfrak{A}_1$ or $\mathfrak{A}_2$, the first or the second component of the two additional dimensions, respectively, will be incremented. In other words, these two additional dimensions record which guess we have made and thus also gives information which constraint set we should use at the end of a computation. Note that the empty word has to be handled separately since, as noted before, ε-moves may not increment the value of the dimensions.

Formally, let $\mathfrak{A} := (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$ be a PDA that is given as follows. The set of states is given by $Q := Q_1 \cup Q_2 \cup \{q_0\}$, where q_0 is a new state, and the stack alphabet is given by $\Gamma := \Gamma_1 \cup \Gamma_2 \cup \{\bot\}$, where $\bot$ is a new stack symbol. The auxiliary set is given by $D := D_1' \cup D_2'$, where

$$D_1' := D_1 \otimes \{0^{n_2}\} \otimes \{(1,0)\}$$

and

$$D_2' := \{0^{n_1}\} \otimes D_2 \otimes \{(0,1)\}\,.$$

The transition function δ is defined as follows. First, for the guessing, we set

$$\delta(q_0, \varepsilon, \bot) = \{(q_{01}, \bot_1), (q_{02}, \bot_2)\}\,.$$

Second, for the simulation of $\mathfrak{A}_1$, we set

$$(q, \beta) \in \delta(p, (a, \bar{d} \otimes 0^{n_2} \otimes (1,0)), Z) \quad \text{if} \quad (q, \beta) \in \delta_1(p, (a, \bar{d}), Z)$$

and

$$(q,\beta) \in \delta(p,\varepsilon,Z) \quad \text{if} \quad (q,\beta) \in \delta_1(p,\varepsilon,Z)\,,$$

for each $p,q \in Q_1$, $a \in \Sigma$, $\bar{d} \in D_1$, $Z \in \Gamma_1$, and $\beta \in \Gamma_1^*$. Third, for the simulation of $\mathfrak{A}_2$, we set

$$(q,\beta) \in \delta(p,(a,0^{n_1} \otimes \bar{d} \otimes (0,1)),Z) \quad \text{if} \quad (q,\beta) \in \delta_2(p,(a,\bar{d}),Z)$$

and

$$(q,\beta) \in \delta(p,\varepsilon,Z) \quad \text{if} \quad (q,\beta) \in \delta_2(p,\varepsilon,Z)\,,$$

for each $p,q \in Q_2$, $a \in \Sigma$, $\bar{d} \in D_2$, $Z \in \Gamma_2$, and $\beta \in \Gamma_2^*$. The set of final states is defined by $F := F_1 \cup F_2$.

For the constraint set, we use the set $C := C_1' \cup C_2' \cup M$ where

$$C_1' := C_1 \otimes \{0^{n_2}\} \otimes (\mathbb{N} \setminus \{0\}) \otimes \{0\}\,,$$
$$C_2' := \{0^{n_1}\} \otimes C_2 \otimes \{0\} \otimes (\mathbb{N} \setminus \{0\})\,,$$

and

$$M := \begin{cases} \{\bar{0}\} & \text{if } \varepsilon \in L(\mathfrak{A}_1,C_1) \text{ or } \varepsilon \in L(\mathfrak{A}_2,C_2), \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that the conditions to be checked for constructing M are instances of the membership problem for PPDA's, which is decidable (see Theorem 4.9 in Section 4.2). Moreover, it should be clear that the sets C_1', C_2', M, and thus also C are semi-linear. Intuitively, the set C_1' or C_2' takes care of the case where $\mathfrak{A}_1$ or $\mathfrak{A}_2$, respectively, was simulated on a nonempty word while the set M takes care of the empty word.

Now, it is straightforward to show that $(\mathfrak{A},C)$ recognizes $L(\mathfrak{A}_1,C_1) \cup L(\mathfrak{A}_2,C_2)$. □

Theorem 4.5. *The class $\mathcal{L}_{\mathrm{PPDA}}$ is effectively closed under concatenation.*

Proof. Let $(\mathfrak{A}_i,C_i)$ be PPDA's of dimension $n_i \geq 1$, for $i = 1,2$, where $\mathfrak{A}_i = (Q_i,\Sigma \times D_i,\Gamma_i,\delta_i,q_{0i},\perp_i,F_i)$. Without loss of generality, we assume that Q_1, Q_2, Γ_1, and Γ_2 are pairwise disjoint.

The main idea is the following. We introduce a new initial state q_0 and a new initial stack symbol $\perp$. First, using an ε-transition, we push $\perp_1$ into the stack and go to state q_{01}. Then, we simulate $\mathfrak{A}_1$. If $\mathfrak{A}_1$ reaches a final state, we push $\perp_2$ into the stack and go to state q_{02}, by using an ε-transition. Note that the new initial stack symbol $\perp$ is needed to prevent the stack from being empty.

Formally, let $(\mathfrak{A}, C)$ be a PPDA of dimension $n_1 + n_2$, where $\mathfrak{A} := (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$ is given as follows. The set of states is defined by $Q := Q_1 \cup Q_2 \cup \{q_0\}$, and the stack alphabet is defined by $\Gamma := \Gamma_1 \cup \Gamma_2 \cup \{\bot\}$. The auxiliary set is given by

$$D := (D_1 \otimes \{0^{n_2}\}) \cup (\{0^{n_1}\} \otimes D_2)\,.$$

The transition function δ is defined as follows. First of all, we set

$$\delta(q_0, \varepsilon, \bot) = \{(q_{01}, \bot_1 \bot)\}\,.$$

Then, for each $p, q \in Q_1$, $a \in \Sigma$, $\bar{d} \in D_1$, $Z \in \Gamma_1$, and $\beta \in \Gamma_1^*$, we set

$$(q, \beta) \in \delta(p, (a, \bar{d} \otimes 0^{n_2}), Z) \quad \text{if} \quad (q, \beta) \in \delta_1(p, (a, \bar{d}), Z)$$

and

$$(q, \beta) \in \delta(p, \varepsilon, Z) \quad \text{if} \quad (q, \beta) \in \delta_1(p, \varepsilon, Z)\,.$$

For each final state $p \in F_1$ and each stack symbol $Z \in \Gamma_1 \cup \{\bot\}$, we add the transition

$$(q_{02}, \bot_2 Z) \in \delta(p, \varepsilon, Z)\,.$$

Note that $\mathfrak{A}_2$ cannot use the stack symbols that lie below $\bot_2$ in the stack since these symbols, by the disjointness of Γ_1 and Γ_2, do not belong to Γ_2. For each $p, q \in Q_2$, $a \in \Sigma$, $\bar{d} \in D_2$, $Z \in \Gamma_2$, and $\beta \in \Gamma_2^*$, we set

$$(q, \beta) \in \delta(p, (a, 0^{n_1} \otimes \bar{d}), Z) \quad \text{if} \quad (q, \beta) \in \delta_2(p, (a, \bar{d}), Z)$$

and

$$(q, \beta) \in \delta(p, \varepsilon, Z) \quad \text{if} \quad (q, \beta) \in \delta_2(p, \varepsilon, Z)\,.$$

The set of final states is defined by $F := F_2$. Finally, we define the constraint set as $C := C_1 \otimes C_2$. □

Theorem 4.6. *The class $\mathcal{L}_{\mathrm{PPDA}}$ is effectively closed under inverse homomorphisms.*

Proof. We combine the proof idea of the closure of context-free languages under inverse homomorphisms, as described by Hopcroft and Ullman [HU79, page 132], with the proof idea of the closure of PFA-recognizable languages under inverse homomorphisms (see Theorem 3.11).

Let $(\mathfrak{A}, C)$ be a PPDA of dimension $n \geq 1$ over Σ, where $\mathfrak{A} = (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$. Furthermore, let $h\colon \Pi^* \to \Sigma^*$ be a homomorphism.

We construct a PPDA $(\mathfrak{A}', C')$ of dimension n over Π with $L(\mathfrak{A}', C') = h^{-1}(L(\mathfrak{A}, C))$ as follows. For the constraint set C', we use the set C. Let

D' be the auxiliary set of $(\mathfrak{A}', C')$ (to be defined later). The main idea in constructing $\mathfrak{A}'$ is, given $(b, \bar{d}) \in \Pi \times D'$, to guess a word $w \in (\Sigma \times D)^*$ with $\Psi(w) = h(b)$ and $\tilde{\Phi}(w) = \bar{d}$ and to store w in an extra buffer in the finite-state control. Then, $\mathfrak{A}'$ simulates the (partial) computation of $\mathfrak{A}$ on w. This extra buffer, roughly speaking, serves as a partial pseudo-input tape for $\mathfrak{A}$.

Of course, we cannot have an infinite amount of memory in the finite-state control. Thus, the length of the buffer, that is, the length of a word $w \in (\Sigma \times D)^*$ that may be stored in the finite-state control, must be bounded. To ensure this boundedness, we allow reading $(b, \bar{d}) \in \Pi \times D'$ from the input of $\mathfrak{A}'$ (and thereby guessing a word $w \in (\Sigma \times D)^*$, which has to be stored in the buffer) only when the buffer is empty. Then, the length of the buffer never exceeds $\max_{b \in \Pi} |h(b)|$ since the guessed word w must satisfy $\Psi(w) = h(b)$.

Formally, let $\mathfrak{A}' := (Q', \Pi \times D', \Gamma, \delta', q_0', \bot, F')$, where

- $Q' := \{[q, w] \mid q \in Q, w \in (\Sigma \times D)^*, \text{ and } |w| \leq \max_{b \in \Pi} |h(b)|\}$,
- $D' := \{\tilde{\Phi}(w) \mid w \in (\Sigma \times D)^* \text{ and } \Psi(w) = h(b), \text{ for some } b \in \Pi\}$,
- $q_0' := [q_0, \varepsilon]$, and
- $F' := \{[q, \varepsilon] \mid q \in F\}$.

Note that both Q' and D' are nonempty and finite since both Π and D are. Now, the transition function δ' is defined as follows:

1. If the extra buffer is empty, $\mathfrak{A}'$ may read $(b, \bar{d}) \in \Pi \times D'$, guess $w \in (\Sigma \times D)^*$ with $\Psi(w) = h(b)$ and $\tilde{\Phi}(w) = \bar{d}$, and store w in the buffer. Formally, for each $(b, \bar{d}) \in \Pi \times D'$, $q \in Q$, and $Z \in \Gamma$, we set

$$([q, w], Z) \in \delta'([q, \varepsilon], (b, \bar{d}), Z)\,,$$

   for each $w \in (\Sigma \times D)^*$ with $\Psi(w) = h(b)$ and $\tilde{\Phi}(w) = \bar{d}$. Note that if $h(b) = \varepsilon$ for some $b \in \Pi$, then we have the transition $([q, \varepsilon], Z) \in \delta'([q, \varepsilon], (b, \bar{0}), Z)$, for each $q \in Q$ and $Z \in \Gamma$.

2. In order to simulate ε-moves of $\mathfrak{A}$, we set

$$([q, w], \beta) \in \delta'([p, w], \varepsilon, Z) \quad \text{if} \quad (q, \beta) \in \delta(p, \varepsilon, Z)\,,$$

   where $p, q \in Q$, $Z \in \Gamma$, $\beta \in \Gamma^*$, and $w \in (\Sigma \times D)^*$ with $|w| \leq \max_{b \in \Pi} |h(b)|$.

3. If the buffer is not empty, then $\mathfrak{A}'$ simulates the moves of $\mathfrak{A}$ on input $(a, \bar{d}) \in \Sigma \times D$ and removes $(a, \bar{d})$ from the buffer; we set

$$([q, w], \beta) \in \delta'([p, (a, \bar{d})w], \varepsilon, Z) \quad \text{if} \quad (q, \beta) \in \delta(p, (a, \bar{d}), Z)\,,$$

   where $p, q \in Q$, $Z \in \Gamma$, $\beta \in \Gamma^*$, $(a, \bar{d}) \in \Sigma \times D$, and $w \in (\Sigma \times D)^*$ with $|w| < \max_{b \in \Pi} |h(b)|$.

The correctness of the proof follows from the construction in a straightforward manner. □

The following theorem resembles the classical connection between context-free languages and regular languages, namely that the class of context-free languages is closed under intersection with regular languages. Here, we show that the intersection of a PPDA-recognizable language and a PFA-recognizable language is PPDA-recognizable. The proof is a slight generalization of the proof for the case of context-free and regular languages.

Theorem 4.7. *The class $\mathcal{L}_{\mathrm{PPDA}}$ is effectively closed under intersection with PFA-recognizable languages. The class $\mathcal{L}_{\mathrm{DPPDA}}$ is effectively closed under intersection with DPFA-recognizable languages.*

Proof. Let $(\mathfrak{A}_1, C_1)$ be a PPDA of dimension $n_1 \geq 1$, where $\mathfrak{A}_1 = (Q_1, \Sigma \times D_1, \Gamma, \delta_1, q_{01}, \bot, F_1)$, and let $(\mathfrak{A}_2, C_2)$ be a PFA of dimension $n_2 \geq 1$, where $\mathfrak{A}_2 = (Q_2, \Sigma \times D_2, \delta_2, q_{02}, F_2)$.

We construct a PPDA $(\mathfrak{A}, C)$ of dimension $n_1 + n_2$ by using a product construction similar to the proof of Theorem 3.10.

Intuitively, we simulate $\mathfrak{A}_1$ using the finite-state control and the stack and simulate $\mathfrak{A}_2$ at the same time using the finite-state control. Formally, let $\mathfrak{A} := (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$, where

- $Q := Q_1 \times Q_2$,
- $D := D_1 \otimes D_2$,
- $q_0 := (q_{01}, q_{02})$,
- $F := F_1 \times F_2$,

and the transition function δ is defined as follows. If $\mathfrak{A}_1$ and $\mathfrak{A}_2$ each make a non-ε-move, then we simulate both of them; we define

$$\delta((p_1, p_2), (a, \bar{d}_1 \otimes \bar{d}_2), Z) := \{((q_1, q_2), \beta) \mid (q_1, \beta) \in \delta_1(p_1, (a, \bar{d}_1), Z) \text{ and } q_2 \in \delta_2(p_2, (a, \bar{d}_2))\},$$

for each $p_1 \in Q_1$, $p_2 \in Q_2$, $a \in \Sigma$, $\bar{d}_1 \in D_1$, $\bar{d}_2 \in D_2$, and $Z \in \Gamma$. If $\mathfrak{A}_1$ makes an ε-move, then $\mathfrak{A}_2$ must remain in its current state; we define

$$\delta((p_1, p_2), \varepsilon, Z) := \{((q_1, p_2), \beta) \mid (q_1, \beta) \in \delta_1(p_1, \varepsilon, Z)\},$$

for each $p_1 \in Q_1$, $p_2 \in Q_2$, and $Z \in \Gamma$. Finally, we define the constraint set to be $C := C_1 \otimes C_2$.

Note that the resulting PPDA $(\mathfrak{A}, C)$ is deterministic if both $(\mathfrak{A}_1, C_1)$ and $(\mathfrak{A}_2, C_2)$ are deterministic. □

The following theorem shows the closure of $\mathcal{L}_{\mathrm{PPDA}}$ under ε-free homomorphisms. Unfortunately, we have not yet succeeded in showing the closure nor the non-closure of this class under homomorphisms for the general case.

Theorem 4.8. *The class $\mathcal{L}_{\mathrm{PPDA}}$ is closed under ε-free homomorphisms.*

Proof. Suppose that $(\mathfrak{A}, C)$ is a PPDA of dimension $n \geq 1$, where $\mathfrak{A} = (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$, and $h\colon \Sigma^* \to \Pi^*$ is an ε-free homomorphism.

For the construction of a PPDA that recognizes $h(L(\mathfrak{A}, C))$, we use the same idea as the proof of the closure of PFA-recognizable languages under homomorphisms (see Theorem 3.12). Nonetheless, the difficulty that arises there does not occur here since the homomorphism under consideration is ε-free.

We construct the PPDA $(\mathfrak{A}', C')$ as follows. Basically, whenever $\mathfrak{A}$ reads $(a, \bar{d}) \in \Sigma \times D$, while replacing the topmost stack symbol $Z \in \Gamma$ with $\beta \in \Gamma^*$, then we want $\mathfrak{A}'$ to read a sequence $(b_1, \bar{d}_1) \cdots (b_m, \bar{d}_m)$ with $b_1 \cdots b_m = h(a)$ and $\bar{d}_1 + \cdots + \bar{d}_m = \bar{d}$, and to perform the same action on the stack.

Formally, let $D' := D \cup \{\bar{0}\}$ be an auxiliary set and let $\mathfrak{A}' := (Q', \Pi \times D', \Gamma, \delta', q_0, \bot, F)$ be the PDA described as follows. The set Q' contains all the states of $\mathfrak{A}$ and, additionally, all the new intermediate states introduced in the definition of δ' below. For each $p, q \in Q$, $a \in \Sigma$, $\bar{d} \in D$, $Z \in \Gamma$, and $\beta \in \Gamma^*$, if $(q, \beta) \in \delta(p, (a, \bar{d}), Z)$ is a transition in $\mathfrak{A}$, then we introduce the following transitions in $\mathfrak{A}'$:

1. If $h(a) = b \in \Pi$, then we define the transition $(q, \beta) \in \delta'(p, (b, \bar{d}), Z)$.
2. If $h(a) = b_1 \cdots b_m \in \Pi^+$, for some $m \geq 2$, then, we introduce $(m-1)$-many new intermediate states $q_1, \ldots, q_{m-1}$ and the transitions
   - $(q_1, Z) \in \delta'(p, (b_1, \bar{0}), Z)$,
   - $(q_{i+1}, Z) \in \delta'(q_i, (b_{i+1}, \bar{0}), Z)$, for $i = 1, \ldots, m-2$, and
   - $(q, \beta) \in \delta'(q_{m-1}, (b_m, \bar{d}), Z)$.

   Note that the topmost symbol Z of the stack is replaced by β just in the last step in order to prevent the stack from being empty.

In addition, for each $p, q \in Q$, $\bar{d} \in D$, $Z \in \Gamma$, and $\beta \in \Gamma^*$, we set

$$(q, \beta) \in \delta'(p, \varepsilon, Z) \quad \text{if} \quad (q, \beta) \in \delta(p, \varepsilon, Z)\,.$$

For the constraint set, we use the given constraint set C.

It is now straightforward to show that $(\mathfrak{A}', C)$ indeed recognizes the language $h(L(\mathfrak{A}, C))$. □

4.2 Decision Problems

In this section we look at the counterparts of the decision problems considered in Section 3.2 for (deterministic) Parikh pushdown automata.

First, let us consider the membership problem. For PFA's, we have given a decision procedure solving this problem (see page 36 and Theorem 3.18). As noted there, the crucial points in this procedure are the decidability of the membership problem for finite automata and for semi-linear sets. Since the membership problem for pushdown automata are also known to be decidable (see, for example, [HU79, pages 139–141]), it follows that this procedure can also be used to solve the membership problem for PPDA's, whereupon we obtain the following result.

Theorem 4.9. *The membership problem for PPDA's is decidable.*

Now, let us consider the emptiness problem. For PFA's, the decidability of this problem relies on the fact that the extended Parikh images of finite automata are semi-linear (see Lemma 3.19). The latter fact, in turn, follows from Parikh's theorem. Since this theorem holds for pushdown automata as well, it is not surprising that this decidability result carries over to PPDA's, as already noted by Klaedtke and Rueß in [KR02, page 6].

Theorem 4.10 (Klaedtke and Rueß)**.** *The emptiness problem for PPDA's is decidable.*

Proof. Let $(\mathfrak{A}, C)$ be a PPDA of dimension $n \geq 1$ over Σ.

Since $\mathfrak{A}$, by definition, is a pushdown automaton, it follows from Parikh's theorem that the Parikh image of $L(\mathfrak{A})$ is effectively semi-linear. Therefore, by Lemma 3.19, the extended Parikh image of $L(\mathfrak{A})$ is effectively semi-linear too. From this point onwards, the proof just follows the proof of Theorem 3.20. □

Adapting the proof of Theorem 3.21, we also obtain the semi-linearity of PPDA-recognizable languages.

Theorem 4.11. *The Parikh image of any PPDA-recognizable language is effectively semi-linear.*

Proof. Let $(\mathfrak{A}, C)$ be a PPDA of dimension $n \geq 1$ over $\Sigma := \{a_1, \ldots, a_m\}$, where $\mathfrak{A} = (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$.

As with PFA's, we extend $(\mathfrak{A}, C)$ to a new PPDA $(\mathfrak{B}, C')$ of dimension $n + m$ that recognizes the same language as $(\mathfrak{A}, C)$ and counts the number of occurrences of the input symbols by associating each transition that involves a_i with the unit vector $\bar{e}_i$, for $i = 1, \ldots, m$.

Formally, we first define the auxiliary set

$$D' := D \otimes \{\bar{e}_1, \ldots, \bar{e}_m\},$$

where $\bar{e}_i$ is the i-th unit vector in $\mathbb{N}^m$, for $i = 1, \ldots, m$. We then define the pushdown automaton $\mathfrak{B} := (Q, \Sigma \times D', \Gamma, \delta', q_0, \bot, F)$, where the transition function δ' is given by

$$\delta'(q, \varepsilon, Z) := \delta(q, \varepsilon, Z)$$

and

$$\delta'(q, (a_i, \bar{d} \otimes \bar{e}_j), Z) := \begin{cases} \delta(q, (a_i, \bar{d}), Z) & \text{if } i = j, \\ \emptyset & \text{otherwise,} \end{cases}$$

for each $q \in Q$, $\bar{d} \in D$, $Z \in \Gamma$, and $i, j \in \{1, \ldots, m\}$. Further, we define the constraint set to be $C' := C \otimes \mathbb{N}^m$.

It should be clear from the construction that $L(\mathfrak{B}, C') = L(\mathfrak{A}, C)$, and from this point onwards, the proof just follows the proof of Theorem 3.21. □

The universality and the equivalence problem for PPDA's are undecidable, since these problems are already undecidable for PFA's. For DPPDA's, these problems remain open.

4.3 Parikh Automata and the Chomsky Hierarchy

In this section, we relate the classes of Parikh automata introduced in the previous and the present chapter to the language classes in the Chomsky hierarchy. In particular, we will consider the classes $\mathcal{L}_{\text{FA}}$, $\mathcal{L}_{\text{DPDA}}$, $\mathcal{L}_{\text{PDA}}$, $\mathcal{L}_{\text{LBA}}$, and $\mathcal{L}_{\text{TM}}$, where the latter two classes are the class of context-sensitive languages, which are characterized by linear-bounded automata (LBA), and the class of recursively enumerable languages, which are characterized by Turing machines (TM). Recall that an LBA is a nondeterministic Turing machine for which the amount of space used in every computation is bounded by a linear function in the length of the input word. For an introduction to the Chomsky hierarchy, the reader is referred, for example, to [HU79].

Unless stated otherwise, in the following we will assume that the alphabet under consideration consists of at least two symbols. This restriction is justified by the fact that already the class of context-free and regular languages over a one-symbol alphabet coincide (see, for example, [Har78, Chapter 6]). Besides, as noted by Salomaa [Sal73, page 68], it is not difficult to show that a language over a one-symbol alphabet is regular if and only if its Parikh image is semi-linear, which follows from the fact that every semi-linear set is the Parikh image of a regular language, as has been remarked on page 13.

Recall that, by Theorem 3.21 and Theorem 4.11, the Parikh images of PFA-recognizable and PPDA-recognizable languages are semi-linear, whereupon we conclude that PFA-recognizable and PPDA-recognizable languages over a one-symbol alphabet are regular.

Under this assumption, the language classes mentioned above constitute a strictly increasing chain of inclusions, namely

$$\mathcal{L}_{\mathrm{FA}} \subsetneq \mathcal{L}_{\mathrm{DPDA}} \subsetneq \mathcal{L}_{\mathrm{PDA}} \subsetneq \mathcal{L}_{\mathrm{LBA}} \subsetneq \mathcal{L}_{\mathrm{TM}} \ .$$

First of all, let us consider using an LBA as the automata component of a Parikh automaton while still using a semi-linear set as the constraint set. For convenience, we refer to the resulting Parikh automaton as a *Parikh LBA*. It should be clear that any LBA is equivalent to a Parikh LBA; the corresponding Parikh LBA just does not make use of its counters (see Remark 3.4). Moreover, it turns out that the converse also holds; that is, any Parikh LBA is equivalent to an LBA. Hence, we do not need to introduce a new language class for the former automata model.

Theorem 4.12. *Each Parikh LBA is equivalent to an LBA, and vice versa.*

Proof. The converse, namely that each LBA is equivalent to a Parikh LBA, has already been justified in our remark above.

Now, let $(\mathfrak{M}, C)$ be a Parikh LBA of dimension $n \geq 1$ over Σ with the auxiliary set $D \subseteq \mathbb{N}^n$. We sketch how a Turing machine $\mathfrak{M}'$ over Σ that recognizes the language recognized by $(\mathfrak{M}, C)$ works. Then, we show that the amount of space used by $\mathfrak{M}'$ is linear in the length of its input, which implies that $\mathfrak{M}'$ is indeed an LBA.

Given an input word $u := u_1 \cdots u_m \in \Sigma$, $m \geq 0$, we first guess nondeterministically the vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ and maintain the sum $\bar{x}$ of these vectors somewhere in the working tape (with a unary representation described below). Then we simulate the computation of $\mathfrak{M}$ on $(u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$. If $\mathfrak{M}$ accepts, then it remains to check whether the vector $\bar{x}$ belongs to C.

Without loss of generality, we assume that C is linear since, otherwise, we can nondeterministically choose one of the finitely many linear sets that constitute C to work with. Let $\bar{x}_0$ be the constant vector and $\bar{x}_1, \ldots, \bar{x}_r$ be the periods of C. First, we subtract $\bar{x}_0$ from $\bar{x}$. Then, one by one, we nondeterministically choose a period $\bar{x}_j$ and subtract it from $\bar{x}$. If $\bar{x}$ belongs to C, then it will eventually become the zero vector, whereupon we go to an accepting state.

We turn to analyzing the amount of space used by $\mathfrak{M}'$ to accept an input word $u \in L(\mathfrak{M}, C)$. Simulating $\mathfrak{M}$ requires only linear space since $\mathfrak{M}$, by assumption, is an LBA. Thus, the critical point lies in how much space is needed

to maintain the vector $\bar{x}$ in the above construction and to check whether $\bar{x}$ belongs to C. The latter task does not need extra space since we only subtract some vectors from $\bar{x}$ and do not add some to $\bar{x}$.

For the former task, we use a unary representation of the vector $\bar{x}$ by means of the word

$$\underbrace{|_1 \cdots |_1}_{(\bar{x})_1\text{-times}} \quad \$ \quad \cdots \quad \$ \quad \underbrace{|_n \cdots |_n}_{(\bar{x})_n\text{-times}} \ ,$$

where $|_1, \ldots, |_n, \$$ are new symbols. Let t be the greatest natural number occurring in D. That is,

$$t := \max\{(\bar{d})_i \mid \bar{d} \in D \text{ and } 1 \leq i \leq n\} .$$

It is not difficult to see that the length of the unary representation of $\bar{x}$ can be bounded by $n \cdot t \cdot |u| + n - 1$, which is linear in $|u|$ since n and t are fixed (the term $n - 1$ represents the number of \$'s). Hence, we conclude that $\mathfrak{M}'$ is indeed an LBA. □

Note that the idea of the proof of Theorem 4.12 above applies to the case of TM's as well. That is to say, if we use a TM as the automata component of a Parikh automaton with a semi-linear constraint set (referred to as a *Parikh TM*), then we do not gain more recognition power than with TM's.

Corollary 4.13. *Each Parikh TM is equivalent to a TM, and vice versa.*

The next corollary of Theorem 4.12 implies the inclusion of $\mathcal{L}_{\text{PPDA}}$ in $\mathcal{L}_{\text{LBA}}$. Moreover, this inclusion is strict.

Corollary 4.14. *The class $\mathcal{L}_{\text{PPDA}}$ is strictly contained in the class $\mathcal{L}_{\text{LBA}}$.*

Proof. Since the automata component of every PPDA, by definition, is a PDA, every PPDA is also a Parikh LBA and thus equivalent to an LBA.

The strictness of the inclusion can be seen as follows. The language

$$L_{\text{quad}} := \{a^k b^{k^2} \mid k \in \mathbb{N}\} \subseteq \{a, b\}^* \tag{4.1}$$

is recognizable by an LBA; we scan the prefix a^k k-times and every time we scan an a we mark a b as read. In the next chapter we will see that the Parikh image of this language is not semi-linear (see Proposition 5.4). By Theorem 4.11, the Parikh image of any PPDA-recognizable language is semi-linear, and therefore L_{quad} is not PPDA-recognizable. □

Using the following two lemmas, we will state the relations among the classes $\mathcal{L}_{\text{FA}}$, $\mathcal{L}_{\text{DPDA}}$, $\mathcal{L}_{\text{PDA}}$, $\mathcal{L}_{\text{DPFA}}$, $\mathcal{L}_{\text{PFA}}$, and $\mathcal{L}_{\text{PPDA}}$ in Theorem 4.17 below.

Lemma 4.15. *There is a DPDA-recognizable language which is not PFA-recognizable.*

Proof. Our candidate will be the language L_{pal} of palindromes over $\{a, b\}$ with center marker (see page 31). By Lemma 3.15, L_{pal} is not PFA-recognizable. Thus, we have proven this lemma if we have shown that L_{pal} is DPDA-recognizable.

The following DPDA $\mathfrak{A}$ recognizes L_{pal}; before c has been seen, $\mathfrak{A}$ stores the input symbols consumed in the stack (in reverse order). After c has been seen, $\mathfrak{A}$ just compares the rest of the input with the symbols stored in the stack. Formally, we define $\mathfrak{A} := (\{q_0, q_1, q_2\}, \{a, b, c\}, \{\bot, A, B\}, \delta, q_0, \bot, \{q_2\})$ with

$$\begin{aligned} \delta(q_0, a, Z) &= \{(q_0, AZ)\} & \delta(q_1, a, A) &= \{(q_1, \varepsilon)\} \\ \delta(q_0, b, Z) &= \{(q_0, BZ)\} & \delta(q_1, b, B) &= \{(q_1, \varepsilon)\} \\ \delta(q_0, c, Z) &= \{(q_1, Z)\} & \delta(q_1, \varepsilon, \bot) &= \{(q_2, \varepsilon)\} \end{aligned}$$

for each $Z \in \{\bot, A, B\}$. Clearly, $\mathfrak{A}$ is a DPDA that recognizes L_{pal}. □

Lemma 4.16. *There is a DPFA-recognizable language that is not context-free (and thus not PDA-recognizable).*

Proof. The language $L_{\text{abc}} = \{a^k b^k c^k \mid k \geq 1\}$ of Example 3.5 is DPFA-recognizable, but not context-free. The proof of the latter fact can be found in standard textbooks of formal language theory, such as [HU79, page 127]. □

Theorem 4.17. *(a) Each of the classes $\mathcal{L}_{\text{DPFA}}$ and $\mathcal{L}_{\text{PFA}}$ are not comparable with each of the classes $\mathcal{L}_{\text{DPDA}}$ and $\mathcal{L}_{\text{PDA}}$.*

(b) $\mathcal{L}_{\text{FA}} \subsetneq \mathcal{L}_{\text{DPFA}} \subsetneq \mathcal{L}_{\text{PFA}} \subsetneq \mathcal{L}_{\text{PPDA}}$.

(c) $\mathcal{L}_{\text{FA}} \subsetneq \mathcal{L}_{\text{DPDA}} \subsetneq \mathcal{L}_{\text{PDA}} \subsetneq \mathcal{L}_{\text{PPDA}}$.

Proof. Part (a) of the theorem follows from Lemma 4.15 and Lemma 4.16: On the one hand, the language L_{pal} is DPDA-recognizable and thus also PDA-recognizable, but not PFA-recognizable and thus also not DPFA-recognizable. On the other hand, the language L_{abc} is DPFA-recognizable and thus also PFA-recognizable, but not PDA-recognizable and thus also not DPDA-recognizable.

For part (b) of the theorem, the inclusion $\mathcal{L}_{\text{FA}} \subseteq \mathcal{L}_{\text{DPFA}}$ follows from Remark 3.4 whereas the other inclusions follow from the definitions immediately. The strictness of the first inclusion follows from the fact that L_{abc} is not regular. The strictness of the second inclusion follows from the fact that the classes

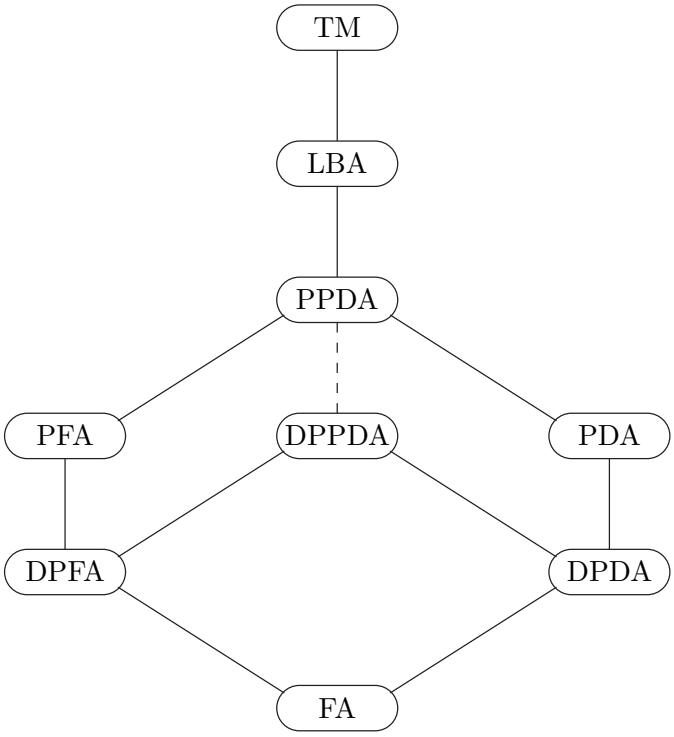


Figure 4.1: Hierarchy of language classes

$\mathcal{L}_{\text{DPFA}}$ and $\mathcal{L}_{\text{PFA}}$ have different closure properties (see Chapter 3). The strictness of the third inclusion follows from the fact that L_{pal} is DPDA-recognizable and thus also PPDA-recognizable, but not PFA-recognizable.

For part (c) of the theorem, the first and the second strict inclusion are well-known results in formal language theory (see, for example, [HU79]). The third inclusion follows from Remark 4.2, and its strictness follows from the fact that the language L_{abc} is DPFA-recognizable and thus also PPDA-recognizable, but not PDA-recognizable. □

Corollary 4.14 and Theorem 4.17 constitute the main results of this section, which are summarized in Figure 4.1. For simplicity, we refer to the language classes by the automata models characterizing them (for example, we write PFA instead of $\mathcal{L}_{\text{PFA}}$). Strict inclusions are indicated by upward, solid edges.

For the class $\mathcal{L}_{\text{DPPDA}}$, we only have some partial results regarding Figure 4.1, which we briefly describe in the following.

The inclusions $\mathcal{L}_{\text{DPFA}} \subseteq \mathcal{L}_{\text{DPPDA}}$ and $\mathcal{L}_{\text{DPDA}} \subseteq \mathcal{L}_{\text{DPPDA}}$ are immediate from the definitions. Moreover, these inclusions are strict since the classes

$\mathcal{L}_{\mathrm{DPFA}}$ and $\mathcal{L}_{\mathrm{DPDA}}$, by Theorem 4.17(a), are not comparable to each other. Another consequence from the latter fact is that $\mathcal{L}_{\mathrm{DPPDA}}$ is not contained in $\mathcal{L}_{\mathrm{PFA}}$ nor in $\mathcal{L}_{\mathrm{PDA}}$. The inclusion $\mathcal{L}_{\mathrm{DPPDA}} \subseteq \mathcal{L}_{\mathrm{PPDA}}$ is also immediate from the definitions, yet we do not know whether this inclusion is strict or not.

Summary and Concluding Remarks

In this chapter, we applied the idea of Parikh automata to pushdown automata and semi-linear sets, thereby concerning ourselves with the first of the questions posed in Chapter 1.

In Section 4.1 we introduced PPDA and DPPDA formally. We showed that $\mathcal{L}_{\mathrm{PPDA}}$ are closed under union, concatenation, inverse homomorphisms, ε-free homomorphisms, and intersection with PFA-recognizable languages. We also showed that $\mathcal{L}_{\mathrm{DPPDA}}$ is closed under intersection with DPFA-recognizable languages.

In Section 4.2 we saw that the membership and the emptiness problem for PPDA's (and DPPDA's) are decidable, mainly by adapting the decision procedures for the counterparts of these problems for PFA's. On the other hand, the question whether the equivalence problem for DPPDA's is decidable remains an unsolved problem. As with PFA's, we showed that the Parikh images of PPDA-recognizable languages are effectively semi-linear.

In Section 4.3 we looked at the relationships between the classes of languages recognized by Parikh automata and the classes of languages in the Chomsky hierarchy. One of the main results is that the class of PPDA-recognizable languages is strictly contained in the class of context-sensitive languages. An open problem is concerned with the relation between the classes $\mathcal{L}_{\mathrm{DPPDA}}$ and $\mathcal{L}_{\mathrm{PPDA}}$; we do not know whether these two classes is different or not.

As a PFA can be viewed as a finite automaton with monotonic counters, a PPDA can also be viewed as a finite automaton with monotonic counters and, additionally, a pushdown stack. In other words, a PPDA can be viewed as a PFA that is augmented with a pushdown stack. In this view, the equivalence between PFA's and reversal-bounded counter machines, as noted at the end of Chapter 3, might suggest that PPDA's are equivalent to reversal-bounded counter machines that are augmented with a pushdown stack, which were also introduced by Ibarra [Iba78]. However, a proof is still missing; Klaedtke and Rueß's proof of the counterpart of this equivalence for PFA's (see [KR02]) relies on the fact that $\mathcal{L}_{\mathrm{PFA}}$ is closed under homomorphisms, which we have not been able to prove for $\mathcal{L}_{\mathrm{PPDA}}$ yet.

Chapter 5

Beyond Semi-Linearity: Level 2 Pushdown Automata

The model of a Parikh automaton, generally, comprises two components: an automaton and a constraint set. As mentioned in Chapter 3, an advantage of this model is that we can use any model of automata, for the automata component, and any class of vector sets, for the constraint set. Despite this flexibility, we should keep in mind which decision problems for the resulting automata model remain decidable. In this chapter, we will focus on the emptiness problem, which appears to have a wide range of applications.

In Chapter 3 and Chapter 4 we have seen that the emptiness problem for PFA's and PPDA's is decidable. These results are essentially based on the following idea. The extended Parikh image of the automata component effectively yields a certain set of vectors of natural numbers. On the other hand, the constraint set is also a set of vectors of natural numbers (of the same dimension as the former). Then, the emptiness problem is decidable if and only if the emptiness problem for the intersection of the former set with the latter set is decidable.

Hence, when applying the idea of Parikh automata to a certain model of automata and to a certain class of sets of vectors of natural numbers, with regard to the emptiness problem, we might be concerned with the following issues:

(a) How can the extended Parikh images of this particular automata model be characterized?

(b) How can the sets from this particular class of vector sets be characterized?

(c) Is it decidable whether the intersection of a set from item (a) with a set from item (b) is empty?

Apparently, the model of pushdown automata and the class of semi-linear sets, with regard to these issues, are quite well understood. However, apart from them, to date the author is not aware of any automata models whose (extended) Parikh images are well-studied to serve as candidates for the automata component of Parikh automata nor of any classes of vector sets that are well-studied to serve as candidates for the constraint set of Parikh automata.

Our efforts and contributions in this chapter are addressed to these issues. Nevertheless, we will focus only on the Parikh image of an automaton, but not on its extended Parikh image. This focus is based on the following consideration. If the Parikh images of an automata class are effectively semi-linear, then, by Lemma 3.19, its extended Parikh images are effectively semi-linear too. If, on the other hand, the Parikh images of an automata class are not effectively semi-linear (in the sense that there is an automaton from this class whose Parikh image is not effectively semi-linear), then we can also find an automaton from this class whose extended Parikh image is not effectively semi-linear.

In the next section we introduce a generalization of pushdown automata called 'level 2 pushdown automata' (2-PDA), which are a special case of 'higher-order pushdown automata' (HOPDA). Then, we investigate the Parikh images of languages recognizable by 2-PDA's. As remarked before, the author is not aware of any significant results on the Parikh images of 2-PDA-recognizable (HOPDA-recognizable) languages. Therefore, it would be interesting to know the recognition power of 2-PDA's (HOPDA's) with respect to the Parikh images.

As an attempt to characterize the Parikh images of 2-PDA-recognizable languages, we develop the notion of 'semi-polynomial sets.' For this purpose, the semi-linear sets serve as our starting point. Moreover, the newly-defined class of vector sets also takes us to the issue of using sets from this class as the constraint set of Parikh automata; the last section of this chapter is devoted to this issue.

5.1 Level 2 Pushdown Automata

An HOPDA, roughly speaking, is a finite automaton augmented with a nested (pushdown) stack, that is, a stack of stacks ... of stacks. A *level n pushdown automaton* (*n-PDA*) is an HOPDA whose stack is n-fold nested; for instance, a 2-PDA is a finite automaton augmented with a stack whose entries are again stacks. Before we give the formal definitions of 2-PDA's, which will be of our concern in this chapter, we give some background remarks on HOPDA's.

HOPDA's were introduced by Maslov [Mas76]. He studied these automata in connection with generalized indexed languages, which are a generalization of the indexed languages introduced by Aho [Aho68]. Later, Engelfriet [Eng83] showed that the indexed languages are precisely the languages recognizable by 2-PDA's. Moreover, he studied the expressiveness of the two-way-input model of HOPDA's in terms of complexity classes. Damm and Goerdt [DG82] used HOPDA's to characterize the so-called 'OI-hierarchy.'

Recently, HOPDA's have been an active subject of research, in particular in connection with monadic second-order logic. For example, Knapik et al. [KNU02] studied HOPDA's in connection with the monadic second-order theory of infinite trees. To mention another example, Carayol and Wöhrle [CW03] studied the connection between the (infinite) configuration graphs of HOPDA's and the hierarchy of infinite graphs with decidable monadic second-order theories introduced by Caucal [Cau02].

We now introduce 2-PDA's formally. The definitions given here follow those in [CW03].

Definition 5.1. Let Γ be a stack alphabet, and let $\bot \in \Gamma$ be the initial stack symbol. A *level* 1 (*pushdown*) *stack* over Γ is a sequence of stack symbols denoted by $[Z_m \cdots Z_1]$, where $Z_1, \ldots, Z_m \in \Gamma$, $m \geq 0$, and Z_m is considered as the topmost symbol of this level 1 stack. A *level* 2 (*pushdown*) *stack* over Γ is a sequence of $r \geq 1$ level 1 stacks over Γ denoted by $[s_r, \ldots, s_1]$, where s_r is considered as the topmost level 1 stack of this level 2 stack. Note that a level 2 stack always contains at least one level 1 stack. The empty level 1 stack is denoted by $[\varepsilon]$ while the empty level 2 stack, which contains only a single empty level 1 stack, is denoted by $[[\varepsilon]]$.

An *instruction*, in general, transforms a level 1 stack or a level 2 stack into another level 1 stack or another level 2 stack, respectively. Formally, an instruction is a function on the set of level 1 stacks or on the set of level 2 stacks. The following instructions can be executed on a level 1 stack:

- pushing a stack symbol into the level 1 stack,
- removing the topmost symbol of the level 1 stack, and
- leaving the level 1 stack as is.

Formally, for each level 1 stack $[Z_m \cdots Z_1]$, we define

- $\mathrm{push}_1^Z([Z_m \cdots Z_1]) := [Z Z_m \cdots Z_1]$, for each $Z \in \Gamma$,
- $\mathrm{pop}_1([Z_m \cdots Z_1]) := [Z_{m-1} \cdots Z_1]$, if $m \geq 1$, and
- $\mathrm{id}([Z_m \cdots Z_1]) := [Z_m \cdots Z_1]$.

The set of instructions that can be executed on a level 1 stack is denoted by Instr_1. The following instructions can be executed on a level 2 stack:

- pushing a stack symbol into the topmost level 1 stack of the level 2 stack,

- removing the topmost symbol of the topmost level 1 stack of the level 2 stack,
- pushing a copy of the topmost level 1 stack of the level 2 stack into the level 2 stack,
- removing the topmost level 1 stack of the level 2 stack completely, and
- leaving the level 2 stack as is.

Formally, for each level 2 stack $[s_r, \ldots, s_1]$, we define

- $\mathrm{push}_1^Z([s_r, \ldots, s_1]) := [\mathrm{push}_1^Z(s_r), s_{r-1}, \ldots, s_1]$, for each $Z \in \Gamma$,
- $\mathrm{pop}_1([s_r, \ldots, s_1]) := [\mathrm{pop}_1(s_r), s_{r-1}, \ldots, s_1]$,
- $\mathrm{push}_2([s_r, \ldots, s_1]) := [s_r, s_r, \ldots, s_1]$,
- $\mathrm{pop}_2([s_r, \ldots, s_1]) := [s_{r-1}, \ldots, s_1]$, if $r \geq 2$, and
- $\mathrm{id}([s_r, \ldots, s_1]) := [s_r, \ldots, s_1]$.

Note that pop_2 can only be executed if the resulting stack is again a level 2 stack. The set of instructions that can be executed on a level 2 stack is denoted by Instr_2.

At any point of a computation, a 2-PDA can read only the topmost stack symbol. The access to this symbol is provided by the function top, which returns the topmost stack symbol and does not change the stack at all:

- $\mathrm{top}([\varepsilon]) := \varepsilon$,
- $\mathrm{top}([Z_m \cdots Z_1]) := Z_m$, for each nonempty level 1 stack $[Z_m \cdots Z_1]$, and
- $\mathrm{top}([s_r, \ldots, s_1]) := \mathrm{top}(s_r)$, for each (nonempty) level 2 stack $[s_r, \ldots, s_1]$.

A (*nondeterministic*) *level* 2 *pushdown automaton* (*2-PDA*) $\mathfrak{A}$ over an alphabet Σ is a system $(Q, \Sigma, \Gamma, \delta, q_0, \bot)$, where Q is a finite, nonempty set of states, Γ is the stack alphabet, $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \mathcal{P}(Q \times \mathrm{Instr}_2)$ is the transition function, $q_0 \in Q$ is the initial state, and $\bot \in \Gamma$ is the initial stack symbol.

A *configuration* of $\mathfrak{A}$ is a pair (q, s), where q is a state in Q , and s is a level 2 stack over Γ. The *initial configuration* of $\mathfrak{A}$ is $(q_0, [[\bot]])$. The 2-PDA $\mathfrak{A}$ can reach a configuration (q', s') from a configuration (q, s) by reading an input symbol $a \in \Sigma$ or without reading any input symbol if $\delta(q, a, \mathrm{top}(s))$ or $\delta(q, \varepsilon, \mathrm{top}(s))$, respectively, contains (q', instr), and $\mathit{instr}(s) = s'$. The 2-PDA $\mathfrak{A}$ *accepts* a word $w \in \Sigma^*$ if $\mathfrak{A}$, from the initial configuration, reaches an *accepting configuration*, that is, a configuration of the form $(q, [[\varepsilon]])$, for some $q \in Q$, after reading w. The language *recognized* by $\mathfrak{A}$ is denoted by $L(\mathfrak{A}) := \{w \in \Sigma^* \mid \mathfrak{A} \text{ accepts } w\}$.

A language L over an alphabet Σ is called *2-PDA-recognizable* if there is a 2-PDA $\mathfrak{A}$ over Σ such that $L(\mathfrak{A}) = L$. The class of 2-PDA-recognizable languages over Σ is denoted by $\mathcal{L}_{\text{2-PDA}}(\Sigma)$ or $\mathcal{L}_{\text{2-PDA}}$, whenever Σ is clear from the context.

As a remark, it is not difficult to see that any pushdown automaton can be simulated by a 2-PDA; the corresponding 2-PDA just works with the topmost level 1 stack and does not copy nor remove this level 1 stack. It follows that the class $\mathcal{L}_{\text{PDA}}$ is contained in the class $\mathcal{L}_{\text{2-PDA}}$.

Note that in the above definition we restrict ourselves to acceptance by empty stack. On the other hand, one could also consider acceptance by final state as the acceptance mode. After all, since we only consider nondeterministic 2-PDA's, it is not difficult to show that both acceptance modes, as with nondeterministic pushdown automata, are indeed equivalent.

Moreover, when describing a 2-PDA, we will only give the idea of how it works since writing down the complete transition function of a 2-PDA can become very awkward. The following example demonstrates how to describe a 2-PDA.

Example 5.2. The language $L := \{uu \mid u \in \{a,b\}^*\}$, which is not context-free, is 2-PDA-recognizable. A 2-PDA that recognizes L works as follows. We use the stack alphabet $\{\bot, \#, a, b\}$, where $\bot$ is the initial stack symbol. On an input word of the form $uu \in \{a,b\}^*$, where $u := u_1 \cdots u_m$, we read the prefix u and store it in the stack, in reverse order. Then, we push the symbol # into the stack, obtaining the level 2 stack

$$[[\#u_m \cdots u_1 \bot]] .$$

Next, we copy the topmost level 1 stack and remove the topmost symbol of the topmost level 1 stack, provided that this symbol is not the initial stack symbol $\bot$. The resulting level 2 stack can be illustrated as follows:

$$\begin{aligned} &[[u_m \cdots u_1 \bot], \\ &[\#u_m \cdots u_1 \bot]] . \end{aligned}$$

The last step is repeated until the topmost level 1 stack contains only $\bot$. After we have removed the latter level 1 stack, the resulting level 2 stack looks as follows

$$\begin{aligned} &[[u_1 \bot], \\ &[u_2 u_1 \bot], \\ &\vdots \\ &[u_{m-1} \cdots u_1 \bot], \\ &[u_m u_{m-1} \cdots u_1 \bot], \\ &[\#u_m u_{m-1} \cdots u_1 \bot]] . \end{aligned}$$

We observe that the symbols at the top of the level 1 stacks constitute, in the order from top to bottom, the word u. Now, for each remaining input

symbol, if this symbol is equal to the topmost symbol of the topmost level 1 stack, we remove the topmost level 1 stack completely. If # appears at the top of the stack, then we know that the level 2 stack contains only one level 1 stack; we empty this level 1 stack (by removing all symbols one by one), thus reaching an accepting configuration.

Example 5.2 shows that 2-PDA's are strictly more powerful than PDA's with respect to recognized languages. Lemma 5.3 and Proposition 5.4 below show that 2-PDA's are strictly more powerful than PDA's also with respect to the Parikh images. First, we show that the language L_{quad} of (4.1) on page 56,

$$L_{\text{quad}} = \{a^k b^{k^2} \mid k \in \mathbb{N}\} \subseteq \{a, b\}^* ,$$

is 2-PDA-recognizable. Second, we show that the Parikh image of L_{quad} is not semi-linear. Then, by Parikh's theorem, it follows that L_{quad} is not PDA-recognizable.

Lemma 5.3. *The language L_{quad} is 2-PDA-recognizable.*

Proof. We give an informal description of a 2-PDA that recognizes L_{quad}. We use the stack alphabet $\{\bot, Z_1, Z_2\}$, where $\bot$ is the initial stack symbol. For simplicity, let $Z := Z_1$.

On an input word $w := a^k b^{k^2}$, we read the a^k-prefix of w while pushing $(2k)$-many Z's into the stack, followed by a Z_2. The resulting level 2 stack is

$$[[Z_2 Z^{2k} \bot]] .$$

Then, we copy the topmost level 1 stack and remove two symbols from the stack, provided that none of these symbols are the initial stack symbol $\bot$, obtaining the level 2 stack

$$\begin{aligned} &[[Z^{2k-1} \bot], \\ &[Z_2 Z^{2k} \quad \bot]] . \end{aligned}$$

The latter two steps are repeated until the topmost level 1 stack contains only one Z. Note that this is possible due to the initial stack symbol $\bot$. Now, the resulting level 2 stack is

$$\begin{aligned} &[[Z \bot], \\ &[Z Z Z \bot], \\ &\quad \vdots \\ &[Z^{2k-1} \quad \bot], \\ &[Z_2 Z^{2k} \qquad \bot]] . \end{aligned}$$

Now, the number of Z's that lie above Z_2 is given by

$$\sum_{i=0}^{k-1}(2i+1)\,,$$

which yields k^2. Thus, we only need to remove the Z's one by one while reading the remaining b's from the input until Z_2 has been seen at the top of the stack, whereupon we empty the stack, thus entering an accepting configuration. □

Proposition 5.4. *There is a 2-PDA-recognizable language whose Parikh image is not semi-linear.*

Proof. Clearly, the Parikh image of L_{quad} is given by

$$A_{\text{quad}} := \{(x_1, x_2) \in \mathbb{N}^2 \mid x_1^2 = x_2\}\,.$$

We verify that A_{quad} is not semi-linear, thus showing the assertion above.

Toward a contradiction, we assume that A_{quad} is a finite union of linear sets, say $A_{\text{quad}} = \bigcup_{i=1}^{r} B_i$, for some $r \geq 1$. Since A_{quad} is infinite, there is some linear set $B \in \{B_1, \ldots, B_r\}$ that contains at least two elements. Let $\bar{y}_0$ be the constant vector of B, and let $\bar{y}_1, \ldots, \bar{y}_m$ be the periods of B. If $m = 0$, or if all periods of B are $\bar{0}$, then B has only one element, namely $\bar{y}_0$. Contradiction. Thus, we can assume that $\bar{y}_1 \neq \bar{0}$. By the definition of linear sets, $\bar{y}_0 + k\bar{y}_1 \in B$, and thus $\bar{y}_0 + k\bar{y}_1 \in A_{\text{quad}}$, for all $k \in \mathbb{N}$. To simplify notations, let $\bar{y}_0 = (y_{01}, y_{02})$ and $\bar{y}_1 = (y_{11}, y_{12})$. Then, by the definition of A_{quad}, we have for *all* $k \in \mathbb{N}$

$$(y_{01} + ky_{11})^2 = y_{02} + ky_{12}\,. \tag{5.1}$$

Since $\bar{y}_1 \neq \bar{0}$, at least one of y_{11} and y_{12} is not zero. Hence, (5.1) is a polynomial equation of degree one or two in k, which has *at most two* solutions. Contradiction. □

Now, having seen that the class of semi-linear sets cannot capture the Parikh images of 2-PDA-recognizable languages, we want to find a class of sets of vectors of natural numbers which characterizes the Parikh images of 2-PDA-recognizable languages. For such a class, we are mainly concerned with the questions

- whether each set from this class is the Parikh image of a 2-PDA-recognizable language, and
- whether the Parikh image of each 2-PDA-recognizable language belongs to this class.

In the following section we propose a generalization of semi-linear sets and answer these two questions for this new class.

5.2 Semi-Polynomial Sets

Let us recall that a subset A of $\mathbb{N}^n$, $n \geq 1$, is called linear if there are vectors $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_m \in \mathbb{N}^n$, $m \geq 0$, such that each vector $\bar{x}$ belongs to A if and only if

$$\bar{x} = \bar{x}_0 + k_1\bar{x}_1 + \ldots + k_m\bar{x}_m\,, \tag{5.2}$$

for some $k_1, \ldots, k_m \in \mathbb{N}$. If we replace the vector $\bar{x}_i$ with its components $x_{i1}, \ldots, x_{in}$, for $i = 0, \ldots, m$, then we can restate (5.2) as

$$\bar{x} = (x_{01}, \ldots, x_{0n}) + (k_1x_{11}, \ldots, k_1x_{1n}) + \ldots + (k_mx_{m1}, \ldots, k_mx_{mn})\,.$$

As x_{ij} is fixed, for $i = 0, \ldots m$ and $j = 1, \ldots, n$, we can consider each term k_ix_{ij} as a linear function in k_i. We now generalize linear sets by replacing such terms with polynomials in k_i.

Definition 5.5. A subset A of $\mathbb{N}^n$, $n \geq 1$, is said to be a *polynomial set of degree* $d \geq 1$ if there are a vector $\bar{x}_0$ and a family of vectors $(\bar{x}_{i,j})_{1\leq i\leq m, 1\leq j\leq d}$ in $\mathbb{N}^n$, for some $m \geq 0$, such that

$$\begin{aligned} A = \{\bar{x}_0 &+ k_1\bar{x}_{1,1} + k_1^2\bar{x}_{1,2} + \cdots + k_1^{d-1}\bar{x}_{1,d-1} + k_1^d\bar{x}_{1,d} \\ &+ \cdots + k_m\bar{x}_{m,1} + k_m^2\bar{x}_{m,2} + \cdots + k_m^{d-1}\bar{x}_{m,d-1} + k_m^d\bar{x}_{m,d} \\ &\mid k_1, \ldots, k_m \in \mathbb{N}\}\,. \end{aligned} \tag{5.3}$$

The vector $\bar{x}_0$ is called the *constant vector* while the vectors $\bar{x}_{i,j}$ ($1 \leq i \leq m$, $1 \leq j \leq d$) are called the *periods* of A. Together, they are called the *generators* of A.

The set A is said to be a *semi-polynomial set of degree* d if it is a finite union of polynomial sets of degree d. The set A is said to be a *polynomial* (*semi-polynomial*) *set* if it is a polynomial (semi-polynomial) set of degree d, for some $d \geq 1$.

Example 5.6. (a) Every (semi-)linear set is a (semi-)polynomial set, namely of degree 1. Moreover, it is also a (semi-)polynomial set of degree d, for each $d \geq 1$, by the definition above.

(b) The set A_{quad} of Proposition 5.4 above is a polynomial set of degree 2 since

$$A_{\text{quad}} = \{(0,0) + k(1,0) + k^2(0,1) \mid k \in \mathbb{N}\}\,.$$

(c) For $d \geq 3$, let $A_d := \{(x_1, x_2) \in \mathbb{N}^2 \mid x_1^d = x_2\}$. The set A_d is a polynomial (and thus also a semi-polynomial) set of degree d since

$$A_d = \{(0,0) + k(1,0) + k^d(0,1) \mid k \in \mathbb{N}\}\,.$$

Moreover, by using arguments similar to the proof of Proposition 5.4, we can show that A_d is not semi-polynomial of degree $d-1$.

(d) The set $\{(x_1, x_2, x_3) \in \mathbb{N}^3 \mid x_1^2 + x_2^2 = x_3\}$ is a polynomial set of degree 2 since it coincides with

$$\{k_1(1,0,0) + k_1^2(0,0,1) + k_2(0,1,0) + k_2^2(0,0,1) \mid k_1, k_2 \in \mathbb{N}\}\,.$$

From Semi-Polynomial Sets to 2-PDA's

We turn to answering the question whether each semi-polynomial set is the Parikh image of a 2-PDA-recognizable language. Regarding this question, we indeed have a positive result.

Toward this result, we will need the following two technical lemmas. First, we show that every polynomial term of a certain degree is the sum of some polynomial terms of lower degrees. Second, we use the latter lemma to inductively construct, for each polynomial term, a 2-PDA that 'generates' this polynomial term.

Lemma 5.7. *Let $k \geq 0$ and $e > 0$ be natural numbers. Then, we have*

$$k^e = \sum_{i=0}^{k-1} \left(c_{e-1} i^{e-1} + c_{e-2} i^{e-2} + \cdots + c_2 i^2 + c_1 i + 1\right)\,, \tag{5.4}$$

where $c_j := \binom{e}{j}$ for $j = 1, \ldots, e-1$.

Proof. We proceed by induction on k.

For $k = 0$, the sum expression on the right-hand side of (5.4) is empty and therefore yields 0. Equally, on the left-hand side of (5.4) we have $k^e = 0^e = 0$.

For the induction step, we assume that (5.4) holds for some $k \geq 0$. That is,

$$k^e = \sum_{i=0}^{k-1} \left[\binom{e}{e-1} i^{e-1} + \binom{e}{e-2} i^{e-2} + \cdots + \binom{e}{1} i + 1\right]\,. \tag{5.5}$$

Together with the binomial theorem (see, for example, [Big89, pages 68–71]), we obtain

$$\begin{aligned}
(k+1)^e &= k^e + \binom{e}{e-1} k^{e-1} + \binom{e}{e-2} k^{e-2} + \cdots + \binom{e}{1} k + 1 \\
&\overset{(5.5)}{=} \sum_{i=0}^{k-1} \left[\binom{e}{e-1} i^{e-1} + \binom{e}{e-2} i^{e-2} + \cdots + \binom{e}{1} i + 1\right] \\
&\qquad + \binom{e}{e-1} k^{e-1} + \binom{e}{e-2} k^{e-2} + \cdots + \binom{e}{1} k + 1 \\
&= \sum_{i=0}^{k} \left[\binom{e}{e-1} i^{e-1} + \binom{e}{e-2} i^{e-2} + \cdots + \binom{e}{1} i + 1\right]\,,
\end{aligned}$$

which completes our proof. □

Using Lemma 5.7, we inductively construct a 2-PDA that, starting and ending with the level 2 stack $[[Z_d Z^{2k} \bot]]$, processes an input word w (k^d)-times, for any word w and any natural numbers $k \geq 0$ and $d \geq 1$. This construction is captured by the following lemma.

Lemma 5.8. *Let $d \geq 1$ and $\Gamma := \{\bot, Z_1, \ldots, Z_d\}$ be a stack alphabet, where $\bot$ is the initial stack symbol. For convenience, we simply write Z instead of Z_1. Further, let Σ be an input alphabet and w be a word over Σ. Then, for each $e = 1, \ldots, d$ and $k \in \mathbb{N}$, we can construct a 2-PDA with designated states p and q such that the 2-PDA proceeds from the configuration $(p, [[Z_d Z^{2k} \bot]])$ to the configuration $(q, [[Z_d Z^{2k} \bot]])$ after reading the input word w^{k^e} (that is, the concatenation of the word w with itself (k^e)-times).*

Proof. We proceed by induction on d.

First of all, let us consider the case $d = 1$ (thus, we only have the subcase $e = 1$). In this case, we have $k^e = k$. That is, starting from the configuration $(p, [[Z_d Z^{2k} \bot]])$, we have to read the input word w k-times and afterwards proceed to the configuration $(q, [[Z_d Z^{2k} \bot]])$. We proceed as follows. First, we copy the topmost level 1 stack and remove Z_d, obtaining the level 2 stack

$$\begin{aligned} &[[Z^{2k} \bot], \\ &[Z_d Z^{2k} \bot]]\,. \end{aligned}$$

Then, we remove two Z's, for each w, until we have seen $\bot$. Finally, we remove the topmost level 1 stack, which at this point contains only $\bot$, completely and go to the state q.

Next, let us consider the case $d = 2$. For the subcase $e = 1$, we proceed as above. For the subcase $e = 2$, we proceed as in Lemma 5.3. Starting from the configuration $(p, [[Z_d Z^{2k} \bot]])$, we first copy the topmost level 1 stack and remove two symbols from the stack, provided that none of these symbols are the initial stack symbol $\bot$, obtaining the level 2 stack

$$\begin{aligned} &[[Z^{2k-1} \bot], \\ &[Z_d Z^{2k} \quad \bot]]\,. \end{aligned}$$

We repeat the latter two steps until the topmost level 1 stack contains only one Z, obtaining the level 2 stack

$$\begin{aligned} &[[Z \bot], \\ &[Z Z Z \bot], \\ &\quad\vdots \\ &[Z^{2k-1} \quad \bot], \\ &[Z_d Z^{2k} \qquad \bot]]\,. \end{aligned}$$

As with Lemma 5.3, the number of Z's that lie above Z_d yields k^2. Now, we only need to remove the Z's one by one, each time while reading the input word w, until Z_d has been seen. Finally, we proceed to the state q, thus reaching the configuration $(q, [[Z_d Z^{2k} \bot]])$.

As the induction step, let us consider the case $d \geq 3$. For the subcases $e = 1, 2$, we can proceed as in the cases $d = 1, 2$ above (without using the induction hypothesis at all). For the subcases $e = 3, \ldots, d$, we proceed as follows. We process the input word w precisely (k^e)-times by applying Lemma 5.7, which asserts that

$$k^e = \sum_{i=0}^{k-1} \left(c_{e-1} i^{e-1} + c_{e-2} i^{e-2} + \cdots + c_2 i^2 + c_1 i + 1 \right) ,$$

where $c_j := \binom{e}{j}$, for $j = 1, \ldots, e-1$ (note that c_j does not depend on k). For intermediate stages of the construction, we introduce new auxiliary states $q_0^{e-1}, \ldots, q_{c_{e-1}}^{e-1}, \ldots, q_0^1, \ldots, q_{c_1}^1$.

Starting from the configuration $(p, [[Z_d Z^{2k} \bot]])$, we copy the topmost level 1 stack, remove three symbols from the stack, provided that none of these symbols are the initial stack symbol $\bot$, and push Z_{e-1} into the stack. The resulting level 2 stack is

$$\begin{array}{rcl} & [Z_{e-1} Z^{2(k-1)} \bot], & \\ [Z_d & Z^{2k} & \bot]] \, . \end{array}$$

We repeat these steps until the topmost level 1 stack contains only $\bot$ and then remove this level 1 stack, resulting in the level 2 stack

$$\begin{array}{rcl} & & [[Z_{e-1} \bot], \\ & & [Z_{e-1} Z Z \bot], \\ & & [Z_{e-1} Z Z Z Z \bot], \\ & & \vdots \\ & [Z_{e-1} \quad Z^{2(k-1)} & \bot], \\ [Z_d & Z^{2k} & \bot]] \, . \end{array}$$

Then, we go to the state q_0^{e-1}.

Before we continue our construction, we give some remarks on the intuition behind this construction. We observe that, in the level 2 stack above, each level 1 stack $[Z_{e-1} Z^{2i} \bot]$ represents a natural number $i \in \{0, \ldots, k-1\}$. Let us consider such a level 1 stack $[Z_{e-1} Z^{2i} \bot]$. Since $e - 1 < d$, the induction hypothesis can be applied. That is, for each $e' = 1, \ldots, e-1$ we can construct

a 2-PDA with designated states p' and q' such that this 2-PDA proceeds from the configuration $(p', [[Z_{e-1}Z^{2i}\bot]])$ to the configuration $(q', [[Z_{e-1}Z^{2i}\bot]])$ after processing the input word w precisely $(i^{e'})$-times. Later, we will apply the induction hypothesis in this manner several times. These particular 2-PDA's (or, to be precise, their transition functions) will be parts of a larger 2-PDA we are constructing (the 'main program,' roughly speaking).

For $i = 0, \ldots, k-1$, if the current state is q_0^{e-1} and the topmost level 1 stack is $[Z_{e-1}Z^{2i}\bot]$, we do the following:

- By the induction hypothesis, we use a 2-PDA that reaches the state q_1^{e-1} after processing the input word w precisely (i^{e-1})-times, with the same stack content as in the state q_0^{e-1}. We repeat this step until we have reached the state $q_{c_{e-1}}^{e-1}$, having processed the input word w $(c_{e-1}i^{e-1})$-times. Then, we go to the state q_0^{e-2}.
- Similarly, we can construct a 2-PDA that, starting from the state q_0^{e-2}, reaches the state $q_{c_{e-2}}^{e-2}$ after reading the input word w $(c_{e-2}i^{e-2})$-times. Note that the topmost level 1 stack from the previous step remains.
- We repeat these steps until we have reached the state $q_{c_1}^1$. Then, we remove the topmost level 1 stack $[Z_{e-1}Z^{2i}\bot]$ completely, while reading w once. At this point, the number of the successive input words w that have been processed is given by

$$(c_{e-1}i^{e-1} + c_{e-2}i^{e-2} + \cdots + c_2 i^2 + c_1 i + 1)\,.$$

  Finally, we go to the state q_0^{e-1} (and proceed with $i+1$).

The procedure described above ends (that is, we are done with $i = 0, \ldots, k-1$) if Z_d appears at the top of the stack. At this point, the number of the successive input words w that have been processed is given by

$$\sum_{i=0}^{k-1} \left(c_{e-1}i^{e-1} + c_{e-2}i^{e-2} + \cdots + c_2 i^2 + c_1 i + 1\right)\,,$$

which, by Lemma 5.7, yields precisely k^e. □

Theorem 5.9. *Every semi-polynomial set is the Parikh image of a 2-PDA-recognizable language.*

Proof. It is not difficult to prove that the class of 2-PDA recognizable languages is effectively closed under union, for instance, by adapting the proof of the effective closure of PDA-recognizable languages under union. Thus, it suffices to restrict ourselves to polynomial sets.

Let n and d be positive natural numbers. Suppose that $A \subseteq \mathbb{N}^n$ is a polynomial set of degree d, given by its constant vector $\bar{x}_0$ and its periods $\bar{x}_{i,j}$ ($1 \leq i \leq m$, $1 \leq j \leq d$), for some $m \geq 0$. We define $\Sigma := \{a_1, \ldots, a_n\}$ and assign to each generators of A a word in $a_1^* \cdots a_n^*$ such that the Parikh image of this word yields the corresponding generators. Let us call these words w_0 and $w_{i,j}$ ($1 \leq i \leq m$, $1 \leq j \leq d$), respectively.

We give an inductive construction of a 2-PDA $\mathfrak{A}$ such that the Parikh image of $L(\mathfrak{A})$ yields A. More precisely, $L(\mathfrak{A})$ will be of the form

$$\begin{aligned} &\{w_0 \\ &\quad w_{1,1}^{k_1} w_{1,2}^{k_1^2} \cdots w_{1,d-1}^{k_1^{d-1}} w_{1,d}^{k_1^d} \\ &\quad \cdots \\ &\quad w_{m,1}^{k_m} w_{m,2}^{k_m^2} \cdots w_{m,d-1}^{k_m^{d-1}} w_{m,d}^{k_m^d} \mid k_1, \ldots, k_m \in \mathbb{N}\}\,. \end{aligned} \tag{5.6}$$

We will use the stack alphabet $\Gamma := \{\bot, Z_1, \ldots, Z_d\}$, where $\bot$ is the initial stack symbol. To simplify notation, let $Z := Z_1$. For each word $w_{i,j}$ and for each k_i ($1 \leq i \leq m$, $1 \leq j \leq d$), we will apply Lemma 5.8 to obtain a 2-PDA that reads the input word $w_{i,j}$ (k_i^j)-times. We then combine these so-obtained 2-PDA's to obtain the desired 2-PDA $\mathfrak{A}$. For intermediate stages of the construction, we introduce new auxiliary states $q_1, \ldots, q_{m+1}$, $q'_1, \ldots, q'_m$, and $q_{i,j}$, for $i = 1, \ldots, m$ and $j = 1, \ldots, d$.

Starting from the initial configuration, we first check whether the input word contains the prefix w_0, without using the stack at all. Then, we go to the state q_1.

For $i = 1, \ldots, m$, starting from the configuration $(q_i, [[\bot]])$, we do the following:

- We first guess k_i by pushing $(2k_i)$-many Z's, followed by a Z_d, into the stack. The resulting level 2 stack is $[[Z_d Z^{2k_i} \bot]]$. Then, we go to the state q'_i.
- By Lemma 5.8, we can construct a 2-PDA that proceeds from the configuration $(q'_i, [[Z_d Z^{2k_i} \bot]])$ to the configuration $(q_{i,1}, [[Z_d Z^{2k_i} \bot]])$ after reading the input word $w_{i,1}$ precisely (k_i)-times.
- Similarly, for each $j = 2, \ldots, d$, we can construct a 2-PDA that proceeds from the configuration $(q_{i,j-1}, [[Z_d Z^{2k_i} \bot]])$ to the configuration $(q_{i,j}, [[Z_d Z^{2k_i} \bot]])$ after reading the input word $w_{i,j}$ precisely (k_i^j)-times.
- We repeat the latter step successively until we have reached the state $q_{i,d}$. At this point, from the state q_i onwards, we have processed the word

$$w_{i,1}^{k_i} w_{i,2}^{k_i^2} \cdots w_{i,d-1}^{k_i^{d-1}} w_{i,d}^{k_i^d}$$

from the input. As we are done with k_i, we remove the level 1 stack that codes k_i from the stack completely, leaving the stack to be $[[\bot]]$. Finally, we go to the state q_{i+1} (and proceed with $i+1$).

The procedure described above ends if we have reached the state q_{m+1}.

By the time the state q_{m+1} has been reached, from the initial configuration onwards, the word

$$w_0 \;\; w_{1,1}^{k_1} w_{1,2}^{k_1^2} \cdots w_{1,d-1}^{k_1^{d-1}} w_{1,d}^{k_1^d} \;\; \cdots \;\; w_{m,1}^{k_m} w_{m,2}^{k_m^2} \cdots w_{m,d-1}^{k_m^{d-1}} w_{m,d}^{k_m^d}$$

has been read from the input, for some $k_1, \ldots, k_m \in \mathbb{N}$. Now, being in the configuration $(q_{m+1}, [[\bot]])$, we remove the initial stack symbol $\bot$, thus reaching the accepting configuration $(q_{m+1}, [[\varepsilon]])$.

It is straightforward to show that $L(\mathfrak{A})$ has the form described by (5.6), and obviously, A is the Parikh image of $L(\mathfrak{A})$. □

From 2-PDA's to Semi-Polynomial Sets

We turn to the question whether the class of semi-polynomial sets exhausts the Parikh images of 2-PDA-recognizable languages. Unfortunately, the following results give a negative answer to this question. We present two kinds of 2-PDA-recognizable languages whose Parikh images are not semi-polynomial. The first one is concerned with using *exponential* terms instead of polynomial terms while the second one is concerned with using terms with *mixed* variables.

For the first kind, we will show that the language

$$L_{\exp} := \{a^k b^{2^k} \mid k \in \mathbb{N}\}$$

is 2-PDA-recognizable, but its Parikh image is not semi-polynomial.

Lemma 5.10. *The language $L_{\exp}$ is 2-PDA-recognizable.*

Proof. The proof idea is due to Carayol and Wöhrle[1].

We use the stack alphabet $\{\bot, \#, Z, 0, 1\}$, where $\bot$ is the initial stack symbol. Intuitively, we use bits (that is, 0 or 1) on top of level 1 stacks as a binary representation of numbers. Using this idea, we can construct a 2-PDA that implements a kind of 'binary counting.'

On an input word $w := a^k b^{2^k}$, we first read the a^k-prefix of w while pushing k-many Z's, followed by a $\#$, into the stack. We then copy the topmost level 1 stack, remove two symbols, provided that none of these symbols are the initial

[1] Arnaud Carayol and Stefan Wöhrle. Communicated by Wolfgang Thomas.

stack symbol $\perp$, and push a 0 into the stack, obtaining the level 2 stack

$$\begin{array}{r}[[0Z^{k-1}\perp],\\ [\#Z^{k}\quad\perp]]\,.\end{array}$$

The latter three steps are repeated until the topmost level 1 stack contains only $\perp$. After removing this level 1 stack completely, we obtain the level 2 stack

$$\begin{array}{r}[[0\perp],\\ [0Z\perp],\\ \vdots\\ [0\ Z^{k-2}\ \perp],\\ [0\ Z^{k-1}\quad\perp],\\ [\#Z^{k}\qquad\perp]]\,.\end{array}$$

A level 2 stack with this structure, in fact, represents a k-bit binary number; for each $i = 0, \ldots, k-1$, the level 1 stack $[x_i Z^i]$, where $x_i \in \{0,1\}$, represents the fact that the i-th bit of the corresponding binary number is x_i, where the 0-th bit is considered as the least significant bit of the binary number.

We now implement the binary counting as follows. Suppose that at some point of the computation the topmost level 1 stack is $[x_i Z^i]$, for some $i \in \{0, \ldots, k-1\}$ and $x_i \in \{0,1\}$. If $x_i = 0$, then we switch it to 1, without changing the stack structure, while reading an input symbol b. If $x_i = 1$, then we copy the topmost level 1 stack and remove the topmost symbol. If now the topmost stack symbol is Z, then we replace it with 0. If, otherwise, the topmost stack symbol is $\perp$, then we remove the topmost level 1 stack completely and continue removing the topmost level 1 stack, until a 0 appears at the top of the stack.

The procedure above is executed each time a 0-bit appears at the top of a level 1 stack. In particular, we execute the above procedure right after we have built the level 2 stack with the structure described above.

It is not difficult to verify that by the time the special symbol # eventually appears at the top of the stack, $(2^k - 1)$-many input symbol b's have been read. Thus, it remains to read one more input symbol b, then to empty the stack, and to go to an accepting configuration. □

Proposition 5.11. *There is a 2-PDA-recognizable language whose Parikh image is not semi-polynomial.*

Proof. Clearly, the Parikh image of L_{exp} is given by

$$A_{\text{exp}} := \{(x_1, x_2) \in \mathbb{N}^2 \mid 2^{x_1} = x_2\}\,.$$

Using an argument similar to the proof of Proposition 5.4, we verify that A_{exp} is not semi-polynomial.

Toward a contradiction, we assume that A_{exp} is a finite union of polynomial sets of degree $d \geq 1$, say $A_{\text{exp}} = \bigcup_{i=1}^{r} B_i$, for some $r \geq 1$. Since A_{exp} is infinite, there is some polynomial set $B \in \{B_1, \ldots, B_r\}$ of degree d that contains at least two elements. Let $\bar{y}_0$ be the constant vector of B, and let $\bar{y}_{i,j}$ ($1 \leq i \leq m$, $1 \leq j \leq d$) be the periods of B, for some $m \geq 0$. If $m = 0$, or if all periods of B are $\bar{0}$, then B has only one element, namely $\bar{y}_0$. Contradiction. Thus, we can assume that $\bar{y}_{i,j} \neq \bar{0}$, for some $1 \leq i \leq m$ and $1 \leq j \leq d$. By the definition of polynomial sets, $\bar{y}_0 + k_i^j \bar{y}_{i,j} \in B$, and thus $\bar{y}_0 + k_i^j \bar{y}_{i,j} \in A_{\text{exp}}$, for all $k_i \in \mathbb{N}$. Then, by the definition of A_{exp}, we have for *all* $k_i \in \mathbb{N}$

$$2^{(\bar{y}_0)_1 + k_i^j (\bar{y}_{i,j})_1} = (\bar{y}_0)_2 + k_i^j (\bar{y}_{i,j})_2\,. \tag{5.7}$$

Since $\bar{y}_{i,j} \neq \bar{0}$, at least one of $(\bar{y}_{i,j})_1$ and $(\bar{y}_{i,j})_2$ is not zero. If one of them is zero, then (5.7) is either a polynomial equation of degree j in k_i or an exponential equation in k_i. If both of them are not zero, then (5.7) is an equation whose left-hand side is an exponential term in k_i and whose right-hand side is a polynomial term in k_i. In all of these cases (5.7) can only have *finitely many* solutions. In other words, (5.7) *cannot* hold for *all* $k_i \in \mathbb{N}$. Contradiction. □

As mentioned before, the second kind of 2-PDA-recognizable languages whose Parikh images are not semi-polynomial is concerned with terms that contain *mixed* variables. Recall that, by contrast, in the definition of polynomial sets the variables $k_1, \ldots, k_m$ of (5.3) may not be mixed. Therefore, it is expected that semi-polynomial sets cannot capture sets with mixed variables such as the following 'product relation.' We will show that the language

$$L_{\text{prod}} := \{a^x b^y c^{xy} \mid x, y \in \mathbb{N}\}$$

is 2-PDA-recognizable, but its Parikh image is not semi-polynomial. In order to prove this result, however, the simple comparisons of growth rates that have been employed in Proposition 5.4 and Proposition 5.11 do not suffice.

Proposition 5.12. *The language L_{prod} is 2-PDA-recognizable, but its Parikh image is not semi-polynomial.*

Proof. For the first assertion, we give an informal description of a 2-PDA that recognizes L_{prod}, using the stack alphabet $\{\bot, Z\}$.

On an input word $w := a^x b^y c^{xy}$, we first read the a^x-prefix of w while pushing x-many Z's into the stack, obtaining the level 2 stack

$$[[Z^x \bot]] \,.$$

Then, for each input symbol b, we copy the topmost level 1 stack. Thus, after the prefix $a^x b^y$ has been processed, the resulting level 2 stack is

$$\left.\begin{matrix} [[Z^x \bot], \\ \vdots \\ [Z^x \bot]] \,. \end{matrix}\right\} y\text{-times}$$

At this point, obviously, the number of Z's in the stack is xy. Now, we only need to remove the Z's one by one while reading the c's from the input until the stack has been empty, thus reaching an accepting configuration.

For the second assertion, we verify that the Parikh image of L_{prod}, which is obviously given by

$$A_{\text{prod}} := \{(x, y, xy) \mid x, y \in \mathbb{N}\} \,,$$

is not semi-polynomial. For convenience, in the following we will sometimes write vectors vertically instead of horizontally.

Toward a contradiction, we assume that A_{prod} is a finite union of polynomial sets of degree $d \geq 1$, say $A_{\text{prod}} = \bigcup_{i=1}^{r} B_i$, for some $s \geq 1$.

Let us consider a polynomial set $B \in \{B_1, \ldots, B_r\}$ of degree d with the constant vector $\bar{x}_0$ and the periods $\bar{x}_{i,j}$ $(1 \leq i \leq m,\ 1 \leq j \leq d)$, for some $m \geq 0$. In other words,

$$\begin{aligned} B = \{ & \bar{x}_0 + k_1 \bar{x}_{1,1} + k_1^2 \bar{x}_{1,2} + \cdots + k_1^{d-1} \bar{x}_{1,d-1} + k_1^d \bar{x}_{1,d} \\ & + \cdots + k_m \bar{x}_{m,1} + k_m^2 \bar{x}_{m,2} + \cdots + k_m^{d-1} \bar{x}_{m,d-1} + k_m^d \bar{x}_{m,d} \\ & \mid k_1, \ldots, k_m \in \mathbb{N}\} \,. \end{aligned} \tag{5.8}$$

If we view the vectors in B componentwise, then we can restate (5.8) as

$$B = \left\{ \begin{pmatrix} x + P_1(k_1) + \cdots + P_m(k_m) \\ y + Q_1(k_1) + \cdots + Q_m(k_m) \\ z + R_1(k_1) + \cdots + R_m(k_m) \end{pmatrix} \;\middle|\; k_1, \ldots, k_m \in \mathbb{N} \right\} , \tag{5.9}$$

where $(x, y, z) = \bar{x}_0$, and P_i, Q_i, R_i are polynomials in k_i with nonnegative integer coefficients and without constants, for $i = 1, \ldots, m$. Without loss of

generality, we can assume that, for each $i = 1, \ldots, m$, not all of P_i, Q_i, R_i are zero polynomials since, otherwise, we could remove all of P_i, Q_i, R_i without affecting the set B.

In order to define B, roughly speaking, we need the variables $k_1, \ldots, k_m$, and in general, m might be greater than one. Our first task will be to show that A_{prod} can be represented as a finite union of polynomial sets that can be defined by using at most one variable.

Toward fulfilling this task, we remark that, for any polynomial P with nonnegative integer coefficients and without constants, $P(0) = 0$. Moreover, P is a zero polynomial if and only if there is a positive natural number k_0 such that $P(k_0) = 0$. In other words,

$$P(k) = 0 \text{ for all } k \geq 0 \quad \text{if and only if} \quad P(k_0) = 0 \text{ for some } k_0 > 0\,. \tag{5.10}$$

The 'only if' part is obvious. Conversely, if $P(k_0) = 0$ for some $k_0 > 0$, then all the coefficients of P are zero since, otherwise, $P(k_0)$ would be nonzero. Therefore, P is a zero polynomial.

We now turn back to the polynomial set B. If $m = 0$ or if $m = 1$, we do not need to do anything to fulfill the task mentioned above. Therefore, we can assume that $m \geq 2$.

Let us now consider a natural number j with $2 \leq j \leq m$. By (5.9), the vectors

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}, \begin{pmatrix} x + P_1(1) \\ y + Q_1(1) \\ z + R_1(1) \end{pmatrix}, \begin{pmatrix} x + P_j(1) \\ y + Q_j(1) \\ z + R_j(1) \end{pmatrix}, \begin{pmatrix} x + P_1(1) + P_j(1) \\ y + Q_1(1) + Q_j(1) \\ z + R_1(1) + R_j(1) \end{pmatrix}$$

belong to B and thus also to A_{prod}. Then, by the definition of A_{prod}, the following equations hold:

$$\begin{aligned} z &= xy \\ z + R_1(1) &= [x + P_1(1)][y + Q_1(1)] \\ z + R_j(1) &= [x + P_j(1)][y + Q_j(1)] \\ z + R_1(1) + R_j(1) &= [x + P_1(1) + P_j(1)][y + Q_1(1) + Q_j(1)] \end{aligned} \tag{5.11}$$

Combining these equations, and after doing some transformations, we obtain

$$\begin{aligned} z + R_1(1) + R_j(1) &= (x + P_1(1) + P_j(1))(y + Q_1(1) + Q_j(1)) \\ &= xy + xQ_1(1) + xQ_j(1) \\ &\quad + yP_1(1) + P_1(1)Q_1(1) + P_1(1)Q_j(1) \\ &\quad + yP_j(1) + P_j(1)Q_1(1)) + P_j(1)Q_j(1) \end{aligned}$$

$$\begin{aligned}
&= [xy + xQ_1(1) + yP_1(1) + P_1(1)Q_1(1)] \\
&\quad + [xy + xQ_j(1) + yP_j(1) + P_j(1)Q_j(1)] \\
&\quad - xy + P_1(1)Q_j(1) + P_j(1)Q_1(1) \\
&= [z + R_1(1)] + [z + R_j(1)] \\
&\quad - z + P_1(1)Q_j(1) + P_j(1)Q_1(1)
\end{aligned}$$

and thus

$$P_1(1)Q_j(1) + P_j(1)Q_1(1) = 0\,.$$

It follows from the latter equation that $P_1(1)Q_j(1) = 0$, which implies that either $P_1(1) = 0$ or $Q_j(1) = 0$.

If $P_1(1) = 0$ then $Q_1(1) \neq 0$ since, otherwise, it follows from (5.11) that $R_1(1) = 0$. Then, by (5.10), all of P_1, Q_1, R_1 are zero polynomials, which contradicts our assumption. It follows that $P_j(1) = 0$, and by (5.10), both P_1 and P_j are zero polynomials.

Analogously, if $Q_j(1) = 0$, it follows that both Q_1 and Q_j are zero polynomials.

Since j was arbitrarily chosen, it follows that either all P_i or all Q_i, for $i = 1, \ldots, m$, are zero polynomials, and hence, we have either

$$B \subseteq \left\{ \begin{pmatrix} x \\ k \\ xk \end{pmatrix} \;\middle|\; k \in \mathbb{N} \right\} \subseteq A_{\text{prod}} \tag{5.12}$$

or

$$B \subseteq \left\{ \begin{pmatrix} k \\ y \\ ky \end{pmatrix} \;\middle|\; k \in \mathbb{N} \right\} \subseteq A_{\text{prod}}\,. \tag{5.13}$$

We now use the result above to obtain a representation of A_{prod} as a finite union of polynomial sets that can be defined by using at most one variable. Let us recall that $A_{\text{prod}} = \bigcup_{i=1}^{r} B_i$, for some polynomial sets $B_1, \ldots, B_r$. By the arguments above, each of these sets either can be defined by using at most one variable or satisfies (5.12) or satisfies (5.13). To summarize, we have the following chain of inclusions

$$\begin{aligned}
A_{\text{prod}} &= \bigcup_{i=1}^{r} B_i \\
&\subseteq \bigcup_{i=1}^{r} \left\{ \begin{pmatrix} \tilde{P}_i(k) \\ \tilde{Q}_i(k) \\ \tilde{P}_i(k)\tilde{Q}_i(k) \end{pmatrix} \;\middle|\; k \in \mathbb{N} \right\} \\
&\subseteq A_{\text{prod}}\,,
\end{aligned}$$

and thus we have

$$A_{\text{prod}} = \bigcup_{i=1}^{r} \left\{ \begin{pmatrix} \tilde{P}_i(k) \\ \tilde{Q}_i(k) \\ \tilde{P}_i(k)\tilde{Q}_i(k) \end{pmatrix} \mid k \in \mathbb{N} \right\},$$

for some appropriate polynomials $\tilde{P}_1, \ldots, \tilde{P}_r, \tilde{Q}_1, \ldots, \tilde{Q}_r$ with nonnegative integer coefficients.

By the definition of A_{prod}, the first two components of A_{prod} exhaust $\mathbb{N}^2$. That is,

$$\bigcup_{i=1}^{r} \left\{ \begin{pmatrix} \tilde{P}_i(k) \\ \tilde{Q}_i(k) \end{pmatrix} \mid k \in \mathbb{N} \right\} = \mathbb{N}^2 .$$

Consequently, by using these pairs, we obtain a representation of the set of nonnegative rational numbers, which is denoted by $\mathbb{Q}_+$, as

$$\mathbb{Q}_+ = \bigcup_{i=1}^{r} \left\{ \frac{\tilde{P}_i(k)}{\tilde{Q}_i(k)} \mid \tilde{Q}_i(k) \neq 0 \text{ and } k \in \mathbb{N} \right\}.$$

From our remark in (5.10), it follows that

$$\mathbb{Q}_+ = \bigcup_{i \in M} \left\{ \frac{\tilde{P}_i(k)}{\tilde{Q}_i(k)} \mid k \geq 1 \right\}, \tag{5.14}$$

where $M \subseteq \{1, \ldots, r\}$ is the set of the indices $i \in \{1, \ldots, r\}$ for which Q_i is not a zero polynomial. We will obtain a contradiction by showing that the right-hand side of (5.14), in fact, cannot exhaust $\mathbb{Q}_+$.

Let $i \in M$ (that is, $\tilde{Q}_i$ is not a zero polynomial). Then, we have

$$\lim_{k \to \infty} \frac{\tilde{P}_i(k)}{\tilde{Q}_i(k)} \in \mathbb{Q}_+ \cup \{\infty\},$$

which means that for any arbitrarily chosen rational number $\epsilon > 0$, almost all values $\tilde{P}_i(k)/\tilde{Q}_i(k)$ (that is, up to finitely many exceptions) are greater than $1/\epsilon$ (if $\lim_{k\to\infty} \tilde{P}_i(k)/\tilde{Q}_i(k) = \infty$) or lie between $\xi_i - \epsilon$ and $\xi_i + \epsilon$, for some appropriate $\xi_i \in \mathbb{Q}_+$ (if $\lim_{k\to\infty} \tilde{P}_i(k)/\tilde{Q}_i(k) = \xi_i$).

Recapitulating, for any rational number $\epsilon > 0$, almost all members of the right-hand side of (5.14) belong to

$$A' := \bigcup_{i \in M'} \{q \in \mathbb{Q}_+ \mid \xi_i - \epsilon < q < \xi_i + \epsilon\} \cup \{q \in \mathbb{Q}_+ \mid q > \frac{1}{\epsilon}\}, \tag{5.15}$$

where M' is the set of the indices $i \in M$ for which $\lim_{k\to\infty} \tilde{P}_i(k)/\tilde{Q}_i(k) = \xi_i$, for some appropriate $\xi_i \in \mathbb{Q}_+$. On the other hand, since $\mathbb{Q}_+$ is dense, ϵ can

be chosen to be arbitrarily small. Therefore, we can find a rational number $\epsilon > 0$ such that $\mathbb{Q}_+ \setminus A'$ contains infinitely many nonnegative rational numbers. Contradiction. □

Concluding this section, we remark that similar results, for the special case of one-symbol alphabet, have been achieved by Lisovik and Karnaukh [LK03]. They, however, used another approach, namely by means of indexed languages, which have been shown to be precisely the languages recognized by 2-PDA's (see Section 5.1, page 63). They showed that the indexed languages over a one-symbol alphabet can generate polynomial and exponential functions (via the Parikh mapping). Our results, therefore, seems to be not very surprising. Nevertheless, regarding the following two aspects, our results are more general. First, we have considered not only languages over a one-symbol alphabet as Lisovik and Karnaukh did. Second, Lisovik and Karnaukh considered *functions* on the natural numbers. In contrast, we have considered sets of vectors of natural numbers, which correspond to *relations* on the natural numbers.

5.3 Intersection of Semi-Polynomial and Semi-Linear Sets

In this section, we consider the possibility of using semi-polynomial sets as the constraint set of a Parikh automaton, while leaving the Parikh image of the automata component to be semi-linear (for instance, by using a PDA for the automata component). For this consideration, in order to retain the decidability of the emptiness problem for the resulting automata model, the question that arises is whether the emptiness problem for the intersection of a semi-polynomial set and a semi-linear set is decidable, as has been noted at the beginning of this chapter.

Unfortunately, we have not yet succeeded in answering this question. Nevertheless, we will see two partial results. First, we will weaken the form of semi-linear sets and see that the emptiness problem for the intersection of a semi-polynomial set and a semi-linear set of this special kind is decidable. Second, we will show that already a slight generalization of semi-polynomial sets leads to an undecidability result.

Before we proceed with the results, it is worth noting that the membership problem and the emptiness problem for semi-polynomial sets, as with semi-linear sets, are decidable. In fact, the methods employed to solve these problems for semi-linear sets (see Section 2.2) can be applied to the case of semi-polynomial sets as well.

Proposition 5.13. *The membership and the emptiness problem for semi-polynomial sets are decidable.*

Proof. Given a polynomial set $A \subseteq \mathbb{N}^n$, $n \geq 1$, with the generators $\bar{x}_0$ and $\bar{x}_{i,j}$ $(1 \leq i \leq m,\, 1 \leq j \leq d)$, and given a vector $\bar{x} \in \mathbb{N}^n$, we can systematically try all possible values of $k_1, \ldots, k_m \in \{0, \ldots, \max_{0 \leq i \leq n}(\bar{x})_i\}$ and check whether $\bar{x}$ can be obtained by using these values.

Furthermore, if A is a finite union of polynomial sets, then we can check whether $\bar{x}$ belongs to one of these polynomial sets. Therefore, we conclude that the membership problem for semi-polynomial sets is decidable.

As with semi-linear sets, a semi-polynomial set is empty if and only if it has no generator vectors. Therefore, we conclude that the emptiness problem for semi-polynomial sets is decidable. □

5.3.1 Componentwise Semi-Linear Sets

In general, the emptiness problem for the intersection of a semi-polynomial set (of a certain degree) and a semi-linear set would be decidable if we could show that this intersection is again semi-polynomial (of the same degree as the semi-polynomial set under consideration) since, by Proposition 5.13, the emptiness problem for semi-polynomial sets is decidable. Actually, regarding the emptiness problem, we do not need the preservation of the degree of the semi-polynomial set under consideration. Anyhow, the following example shows that we cannot hope for this preservation of degree either.

Example 5.14. Let $A := \{k_1^2(1,0) + k_2^3(0,1) \mid k_1, k_2 \in \mathbb{N}\}$ be a polynomial set of degree 3, and let $B := \{k(1,1) \mid k \in \mathbb{N}\}$ be a linear set. Then, the intersection of A and B is given by

$$\begin{aligned} A \cap B &= \{k_1^2(1,0) + k_2^3(0,1) \mid k_1, k_2 \in \mathbb{N} \text{ and } k_1^2 = k_2^3\} \\ &= \{k^6(1,1) \mid k \in \mathbb{N}\}\,, \end{aligned}$$

which is a polynomial set of degree 6. By using arguments similar to the ones in the proof of Proposition 5.4, we can show that $A \cap B$ is not a semi-polynomial set of degree 3.

Apparently, the non-preservation of degree stems from the fact that by using a semi-linear set, we might establish a connection between the components of a vector of natural numbers. Hence, our approach will be to prohibit the connections between components by using only semi-linear sets that are defined 'componentwise.'

Definition 5.15. A set A of vectors of dimension $n \geq 1$ is called *componentwise linear* if there are linear sets $A_1, \ldots, A_n \subseteq \mathbb{N}$ (of dimension 1) such that

$$A = A_1 \otimes \cdots \otimes A_n \,.$$

The set A is called *componentwise semi-linear* if it is a finite union of componentwise linear sets.

To simplify notation, in the sequel we do not distinguish between an ordinary natural number and a one-dimensional vector of natural numbers.

We will show that the intersection of a componentwise semi-linear set and a semi-polynomial set is again a semi-polynomial set. Moreover, the degree of the given semi-polynomial set is preserved by this intersection. Toward this result, we will need some technical lemmas, in particular concerning the greatest common divisor of natural numbers.

In the sequel, for a finite, nonempty sequence $x_1, \ldots, x_m$ of positive integers (that is, nonzero natural numbers), let $\gcd(x_1, \ldots, x_m)$ be the greatest common divisor of $x_1, \ldots, x_m$. Note that the greatest common divisor, by definition, is a positive integer.

In some proofs below, we will make use of the following facts concerning the greatest common divisor of *two* positive integers. For the proof of these facts, which involves an application of the Euclidean algorithm, the reader is referred to standard textbooks on discrete mathematics or on number theory, for instance, to [GKP94, pages 103–104].

Theorem 5.16. *Let x and y be two positive integers.*

(a) For any positive integer d, if d divides both x and y, then d also divides $\gcd(x, y)$.

(b) We can effectively find two integers k and l such that $\gcd(x, y) = kx + ly$.

Lemma 5.18 below, more or less, extends Theorem 5.16(b); it asserts that the greatest common divisor of a sequence $x_1, \ldots, x_m$ of $m \geq 1$ positive integers can be represented as some kind of a 'linear combination' of $x_1, \ldots, x_m$. Toward proving this lemma, we first show that the computation of the greatest common divisor of more than two positive integers can be reduced to the case of only two positive integers.

Lemma 5.17. *For any sequence $x_1, \ldots, x_m$ of $m \geq 2$ positive integers, it holds that*

$$\gcd(x_1, \ldots, x_m) = \gcd(\gcd(x_1, \ldots, x_{m-1}), x_m) \,.$$

Proof. By induction on m, we show the following assertions simultaneously:

(a) It holds that $\gcd(x_1, \ldots, x_m) = \gcd(\gcd(x_1, \ldots, x_{m-1}), x_m)$.

(b) For any positive integer d, if d divides all of $x_1, \ldots, x_m$, then d also divides $\gcd(x_1, \ldots, x_m)$.

Part (a) of these assertions is exactly the assertion of the lemma to be proven; therefore, we have shown this lemma if we have shown these assertions.

The induction basis, the case $m = 2$, is obvious; part (b) is just the assertion of Theorem 5.16(a) above. For part(a), we only have to observe that, clearly, $\gcd(x_1) = x_1$.

For the induction step, we assume that both assertions hold, for some $m \geq 2$. We show that the assertions also hold for $m + 1$.

We first show part (a) of the induction step by proving

$$\gcd(x_1, \ldots, x_{m+1}) \geq \gcd(\gcd(x_1, \ldots, x_m), x_{m+1}) \tag{5.16}$$

and

$$\gcd(x_1, \ldots, x_{m+1}) \leq \gcd(\gcd(x_1, \ldots, x_m), x_{m+1})\,. \tag{5.17}$$

For (5.16), we observe that $\gcd(\gcd(x_1, \ldots, x_m), x_{m+1})$, by definition, divides $\gcd(x_1, \ldots, x_m)$ as well as x_{m+1}. From the former, it also follows that $\gcd(\gcd(x_1, \ldots, x_m), x_{m+1})$ divides all of $x_1, \ldots, x_m$. To summarize, the right-hand side of (5.16) divides all of $x_1, \ldots, x_{m+1}$ and therefore must be less than or equal to $\gcd(x_1, \ldots, x_{m+1})$, the greatest common divisor of $x_1, \ldots, x_{m+1}$.

For (5.17), we observe that $\gcd(x_1, \ldots, x_{m+1})$, by definition, divides all of $x_1, \ldots, x_m$ and also divides x_{m+1}. From the former, by part (b) of the induction hypothesis, it follows that $\gcd(x_1, \ldots, x_{m+1})$ also divides $\gcd(x_1, \ldots, x_m)$. Thus, we conclude that $\gcd(x_1, \ldots, x_{m+1})$ must be less than or equal to $\gcd(\gcd(x_1, \ldots, x_m), x_{m+1})$, the greatest common divisor of $\gcd(x_1, \ldots, x_m)$ and x_{m+1}.

We now show part (b) of the induction step. Suppose that d is a positive integer that divides all of $x_1, \ldots, x_{m+1}$. By part (b) of the induction hypothesis, it follows that d divides $\gcd(x_1, \ldots, x_m)$. By assumption, d divides x_{m+1} as well. Thus, by Theorem 5.16(a), d divides $\gcd(\gcd(x_1, \ldots, x_m), x_{m+1})$. The latter term, by part (a) of the *induction step* above, precisely yield $\gcd(x_1, \ldots, x_{m+1})$. Thus, we conclude that d divides $\gcd(x_1, \ldots, x_{m+1})$. □

Lemma 5.18. *Let $x_1, \ldots, x_m$ be positive integers, where $m \geq 1$. Then, we can effectively find some natural numbers $k_1, \ldots, k_m$ and a set $I \subseteq \{1, \ldots, m\}$, such that*

$$\gcd(x_1, \ldots, x_m) = \sum_{i \in I} k_i x_i - \sum_{i \in \{1,\ldots,m\} \setminus I} k_i x_i\,.$$

Proof. We proceed by induction on m.

The case $m = 1$ is obvious; since $\gcd(x_1) = x_1$, we only need to define $k_1 := 1$ and $I := \{1\}$, obtaining

$$\gcd(x_1) = k_1 x_1 \,.$$

For the induction step, we assume that the assertion holds, for some $m \geq 1$. We show that then the assertion also holds for $m + 1$.

By Lemma 5.17, we have

$$\gcd(x_1, \ldots, x_{m+1}) = \gcd(\gcd(x_1, \ldots, x_m), x_{m+1}) \,.$$

Then, by the number-theoretical result from Theorem 5.16(b), we can effectively find two integers k and l such that

$$\gcd(x_1, \ldots, x_{m+1}) = k \gcd(x_1, \ldots, x_m) + l x_{m+1} \,. \tag{5.18}$$

Note that k and l may be negative. Nevertheless, since the greatest common divisor of positive integers is positive by definition, at most only one of the numbers k and l is negative.

By the induction hypothesis, we can effectively find some natural numbers $l_1, \ldots, l_m$ and a set $J \subseteq \{1, \ldots, m\}$ such that

$$\gcd(x_1, \ldots, x_m) = \sum_{j \in J} l_j x_j - \sum_{j \in \{1,\ldots,m\} \setminus J} l_j x_j \,. \tag{5.19}$$

Combining (5.18) and (5.19), we obtain

$$\gcd(x_1, \ldots, x_{m+1}) = k \left(\sum_{j \in J} l_j x_j - \sum_{j \in \{1,\ldots,m\} \setminus J} l_j x_j \right) + l x_{m+1} \,,$$

and further,

$$\gcd(x_1, \ldots, x_{m+1}) = \sum_{j \in J} k l_j x_j - \sum_{j \in \{1,\ldots,m\} \setminus J} k l_j x_j + l x_{m+1} \,. \tag{5.20}$$

It remains to find the appropriate natural numbers $k_1, \ldots, k_{m+1}$ and the appropriate set $I \subseteq \{1, \ldots, m+1\}$. For this purpose, we only need to observe which of the terms constituting the representation of $\gcd(x_1, \ldots, x_{m+1})$ in (5.20) are nonnegative and which of them are negative. This observation, in turn, depends on whether one of the numbers k and l is negative or none of

them are (recall that at most only one of them is negative) since $l_1, \ldots, l_m$, by assumption, are natural numbers (and thus nonnegative). We now define

$$I := \begin{cases} J \cup \{m+1\} & \text{if none of the numbers } k \text{ and } l \text{ are negative,} \\ \{1, \ldots, m+1\} \setminus J & \text{if } k \text{ is negative (and } l \text{ is nonnegative),} \\ J & \text{if } l \text{ is negative (and } k \text{ is nonnegative).} \end{cases}$$

Finally, for the natural numbers $k_1, \ldots, k_{m+1}$, we define

$$k_j := |k| l_i \,,$$

for each $i \in \{1, \ldots, m\}$, and

$$k_{m+1} := |l| \,. \qquad \square$$

Using Lemma 5.18 above, we now show the following lemma, which will be crucial to proving our main result; it asserts that the elements of any infinite, one-dimensional linear set, up to finitely many exceptions, can be generated by the greatest common divisor of the periods of the linear set. Moreover, this representation can effectively be found from the generators of the given linear set.

Lemma 5.19. *Let $A \subseteq \mathbb{N}$ be a one-dimensional linear set, whose constant vector is given by the natural number x_0, and whose periods are given by the positive integers $x_1, \ldots, x_m$, for some $m \geq 1$. Further, let g be the greatest common divisor of $x_1, \ldots, x_m$. Then, we can effectively find a natural number z_0 such that the set $\{x_0 + z_0 g + kg \mid k \in \mathbb{N}\}$ is subsumed by A. In addition, the set $A \setminus \{x_0 + z_0 g + kg \mid k \in \mathbb{N}\}$ is finite.*

Proof. According to Lemma 5.18, we can effectively find some natural numbers $k_1, \ldots, k_m$ and a set $I \subseteq \{1, \ldots, m\}$, such that

$$g = \sum_{i \in I} k_i x_i - \sum_{i \in \{1,\ldots,m\} \setminus I} k_i x_i \,. \tag{5.21}$$

For convenience, let

$$\xi := \sum_{i \in I} k_i x_i \quad \text{and} \quad \lambda := \sum_{i \in \{1,\ldots,m\} \setminus I} k_i x_i \,. \tag{5.22}$$

Using ξ and λ, we can restate (5.21) as

$$g = \xi - \lambda \,.$$

We now define

$$z_0 := \lambda^2 \,.$$

It remains to show that the set

$$C := \{x_0 + z_0 g + kg \mid k \in \mathbb{N}\}$$

is subsumed by A. In other words, we have to show that $x_0 + z_0 g + kg$ belongs to A, for each $k \in \mathbb{N}$.

Before we proceed, it is worth noting that ξ and λ of (5.22) above are each obtained by summing some nonnegative multiples of the periods of A. Thus, the sum of the constant vector x_0 of A and some nonnegative multiples of ξ and λ yields an element of A. In other words, the vector (or the number)

$$x_0 + p\xi + q\lambda \tag{5.23}$$

belongs to A, for any natural numbers p and q.

If λ (and thus also z_0) is zero, then we have $g = \xi$. In this case, $x_0 + z_0 g + kg$ can be expressed as $x_0 + k\xi$, for each natural number k. By the remark above, we then conclude that $x_0 + z_0 g + kg$ belongs to A. Hence, in the sequel we will assume that λ is nonzero.

First, suppose that k is a natural number with $k < \lambda$. Now, consider the term $x_0 + z_0 g + kg$. Replacing z_0 with λ^2, we obtain

$$x_0 + \lambda^2 g + kg\,.$$

Further, replacing the *second* g with $\xi - \lambda$, we obtain

$$x_0 + \lambda^2 g + k(\xi - \lambda)\,.$$

After doing some transformations, we obtain

$$x_0 + (\lambda g - k)\lambda + k\xi\,. \tag{5.24}$$

In (5.24) above, the term $(\lambda g - k)$ is positive since, by assumption, k is less than λ. Thus, the term in (5.24) has the form described in (5.23). Hence, we conclude that $x_0 + z_0 g + kg$ indeed belongs to A.

Second, suppose that k is a natural number with $k \geq \lambda$. For such a number k, we can effectively find some natural numbers $l \geq 1$ and $l' < \lambda$ such that

$$k = l\lambda + l'\,,$$

by computing the quotient l and the remainder l' of the division of k by λ (see, for example, [Big89, pages 14–16]). Then, after replacing k with $l\lambda + l'$ in the term $x_0 + z_0 g + kg$, we obtain

$$x_0 + z_0 g + (l\lambda + l')g\,,$$

and further, after doing some transformations, we obtain

$$x_0 + z_0 g + l' g + l g \lambda \,.$$

Since $l' < \lambda$, we can apply the case we have dealt with above to show that $x_0 + z_0 g + l' g$ belongs to A. Furthermore, the term lg, clearly, is not negative. Thus, the term $lg\lambda$ merely adds some nonnegative multiples of λ to $x_0 + z_0 g + l' g$, which already belongs to A. Then, by the remark above, $x_0 + z_0 g + l' g + lg\lambda$ belongs to A as well. Hence, we conclude that $x_0 + z_0 g + kg$ indeed belongs to A.

We turn to proving that the set $A \setminus C$ is finite. Our approach will be to show that the set A is subsumed by the set $\{x_0 + kg \mid k \in \mathbb{N}\}$. As a result, the set $A \setminus C$ is precisely given by the set

$$\{x_0 + kg \mid k \in \mathbb{N} \text{ and } 0 \leq k < z_0\} \,,$$

which is finite since there are only finitely many natural numbers between 0 and z_0. In other words, the set C precisely contains the elements of A from $x_0 + z_0 g$ onwards.

Now, suppose that x belongs to A. That is,

$$x = x_0 + k_1 x_1 + \cdots + k_m x_m \,,$$

for some natural numbers $k_1, \ldots, k_m$. Since g is the greatest common divisor of $x_1, \ldots, x_m$, we can restate the equation above as

$$x = x_0 + k_1 x'_1 g + \cdots + k_m x'_m g \,,$$

for some natural numbers $x'_1, \ldots, x'_m$. Hence, $x = x_0 + kg$, for some natural number k, namely

$$k := k_1 x'_1 + \cdots + k_m x'_m \,,$$

whereupon we conclude that x belongs to the set $\{x_0 + kg \mid k \in \mathbb{N}\}$. □

Now we are ready to prove the main result of the present subsection, namely that the intersection of a componentwise semi-linear set and a semi-polynomial set yields a semi-polynomial set of the same degree as the given semi-polynomial set. Consequently, it is decidable whether the intersection of a semi-polynomial set and a componentwise semi-linear set is decidable.

Theorem 5.20. *Let $n \geq 1$. Furthermore, let $A \subseteq \mathbb{N}^n$ be a componentwise semi-linear set, and let $B \subseteq \mathbb{N}^n$ be a semi-polynomial set of degree $d \geq 1$. Then, $A \cap B$ is a semi-polynomial set of degree d.*

Proof. Since intersection distributes over union, it suffices to consider the case that A is componentwise linear and B is polynomial (of degree d). We will construct a semi-polynomial representation of $A \cap B$ by a refinement process that successively covers the n components one by one. For the intersection of the first component of A and of B, we obtain a semi-polynomial set, which will then be made smaller by taking into account the remaining components one by one.

Case 1 ($n = 1$). Suppose that $A \subseteq \mathbb{N}$ is a (componentwise) linear set and $B \subseteq \mathbb{N}$ is a polynomial set of degree $d \geq 1$. That is, there are some natural numbers $x_0, \ldots, x_m$ such that

$$A = \{x_0 + k_1 x_1 + \ldots + k_m x_m \mid k_1, \ldots, k_m \in \mathbb{N}\},$$

and there are some natural numbers y_0 and $(y_{i,j})$, for $i = 1, \ldots, r$ and $j = 1, \ldots, d$, such that

$$\begin{aligned} B = \{&y_0 + k_1 y_{1,1} + k_1^2 y_{1,2} + \cdots + k_1^{d-1} y_{1,d-1} + k_1^d y_{1,d} \\ &+ \cdots + k_r y_{r,1} + k_r^2 y_{r,2} + \cdots + k_r^{d-1} y_{r,d-1} + k_r^d y_{r,d} \\ &\mid k_1, \ldots, k_r \in \mathbb{N}\}. \end{aligned}$$

To simplify notation, let $P\colon \mathbb{N}^r \to \mathbb{N}$ be defined by

$$\begin{aligned} P(k_1, \ldots, k_r) := {}& y_0 + k_1 y_{1,1} + k_1^2 y_{1,2} + \cdots + k_1^{d-1} y_{1,d-1} + k_1^d y_{1,d} \\ &+ \cdots + k_r y_{r,1} + k_r^2 y_{r,2} + \cdots + k_r^{d-1} y_{r,d-1} + k_r^d y_{r,d}, \end{aligned}$$

for each tuple $(k_1, \ldots, k_r) \in \mathbb{N}^r$. The function P is a polynomial of degree d in $k_1, \ldots, k_r$ with nonnegative integer coefficients, where the variables $k_1, \ldots, k_r$ may not be mixed (that is, each monomial in P contains only at most one of the variables $k_1, \ldots, k_r$).

Without loss of generality, we assume that the periods of A are all positive; otherwise, we could eliminate the periods that are zero without affecting the set A. Similarly, we assume, without loss of generality, that the periods of B satisfy $\sum_{j=1}^{d} y_{i,j} \geq 1$, for each $i = 1, \ldots, r$; that is, at least one of the numbers $y_{i,1}, \ldots, y_{i,d}$ is positive.

Furthermore, we may assume that, for both A and B, at least one period is positive (that is, $m \geq 1$ and $r \geq 1$). Otherwise, A or B is finite, and therefore, also the intersection of A and B is finite. We can effectively compute this finite set since the membership problem for semi-linear and for semi-polynomial sets are decidable. Moreover, this finite set is semi-linear since every finite set is semi-linear (see Remark 2.2(c)). Thus, as has been noted in Example 5.6(a), this finite set is a semi-polynomial set of degree d.

Let g be the greatest common divisor of $x_1, \ldots, x_m$. By Lemma 5.19, we can effectively find a natural number z_0 such that the set

$$C := \{x_0 + z_0 g + kg \mid k \in \mathbb{N}\}$$

is subsumed by A and the set $A \setminus C$ is finite. Thus, we can decompose $A \cap B$ as the union of the set $(A \setminus C) \cap B$ and the set $C \cap B$. As noted above, since the set $A \setminus C$ is finite, we can effectively compute the finite set $F := (A \setminus C) \cap B$.

We turn to the set $C \cap B$. By definition, a natural number x belongs to $C \cap B$ if and only if

$$x = x_0 + z_0 g + kg \quad \text{and} \quad x = P(k_1, \ldots, k_r)\,, \tag{5.25}$$

for some $k, k_1, \ldots, k_r \in \mathbb{N}$. By computing the remainder and the quotient of the division of $x_0 + z_0 g$ by g (see, for example, [Big89, pages 14–16]), we can effectively find some natural numbers $z_0' < g$ (the remainder) and z_0'' (the quotient) such that $x_0 + z_0 g = z_0' + z_0'' g$. As a result, we can replace (5.25) with

$$x = z_0' + z_0'' g + kg \quad \text{and} \quad x = P(k_1, \ldots, k_r)\,. \tag{5.26}$$

The condition (5.26), in turn, is equivalent to the following conditions:

(a) $x = P(k_1, \ldots, k_r)$, for some solution $k_1, \ldots, k_r \in \mathbb{N}$ of the congruence equation

$$z_0' \equiv P(k_1, \ldots, k_r) \pmod{g} \tag{5.27}$$

in natural numbers, and

(b) $x \geq z_0'' g$.

For (a), without loss of generality, we may assume that such a solution satisfies the condition $k_1, \ldots, k_r < g$; if there is a solution that does not satisfy the latter condition, then there is also a solution that satisfies it.

If (5.27) has no solutions, then the set $C \cap B$ is empty and thus $A \cap B = F$.

If (5.27) has a solution, say $\bar{s} = (s_1, \ldots, s_r) \in \{0, \ldots, g-1\}^r$, then this solution generates the elements of B each of which is of the form

$$x = P(k_1, \ldots, k_r)\,, \tag{5.28}$$

where

$$k_i = s_i + l_i g\,, \tag{5.29}$$

for some $l_i \in \mathbb{N}$, for each $i = 1, \ldots, r$. In other words, such $(k_1, \ldots, k_r) \in \mathbb{N}^r$ is also a solution of (5.27), and thus, $x = P(k_1, \ldots, k_r)$ satisfies (a).

For (b), we want to ensure that the number x obtained from (5.28) above satisfies $x \geq z_0'' g$. By our assumptions on the periods of B, it suffices to require that $k_i \geq z_0'' g$, for each $i = 1, \ldots, r$. Only finitely many members of $C \cap B$ do not meet this requirement; we can effectively collect them in the finite set $E_{\bar{s}}$. With this requirement, a solution $(s_1, \ldots, s_r) \in \{0, \ldots, g-1\}^r$ generates the elements of $C \cap B$ each of which is of the form

$$x = P(k_1, \ldots, k_r)\,, \tag{5.30}$$

where

$$k_i = s_i + (z_0'' + l_i)g\,, \tag{5.31}$$

for some $l_i \in \mathbb{N}$, for each $i = 1, \ldots, r$.

Now, by replacing each k_i in (5.30) with the right-hand side of (5.31), we obtain a function $Q\colon \mathbb{N}^r \to \mathbb{N}$ with

$$\begin{aligned} Q_{\bar{s}}(l_1, \ldots, l_r) &:= P((s_1 + (z_0'' + l_1)g), \ldots, (s_r + (z_0'' + l_r)g)) \\ &= y_0 + (s_1 + (z_0'' + l_1)g)y_{1,1} + (s_1 + (z_0'' + l_1)g)^2 y_{1,2} + \cdots \\ &\quad + (s_1 + (z_0'' + l_1)g)^{d-1} y_{1,d-1} + (s_1 + (z_0'' + l_1)g)^d y_{1,d} \\ &\quad + \cdots \\ &\quad + (s_r + (z_0'' + l_r)g)y_{r,1} + (s_r + (z_0'' + l_r)g)^2 y_{r,2} + \cdots \\ &\quad + (s_r + (z_0'' + l_r)g)^{d-1} y_{r,d-1} + (s_r + (z_0'' + l_r)g)^d y_{r,d}\,, \end{aligned}$$

for each tuple $(l_1, \ldots, l_r) \in \mathbb{N}^r$. Note that, as with P, the function $Q_{\bar{s}}$ is a polynomial of degree d in $l_1, \ldots, l_r$ with nonnegative integer coefficients, where the variables $l_1, \ldots, l_r$ are not mixed. Thus, we may use $Q_{\bar{s}}$ to represent a polynomial set of degree d.

To summarize, we obtain

$$C \cap B = \bigcup_{\substack{\bar{s} \in \{0,\ldots,g-1\}^r \\ \bar{s} \text{ is a solution of (5.27)}}} E_{\bar{s}} \cup \{Q_{\bar{s}}(l_1, \ldots, l_r) \mid l_1, \ldots, l_r \in \mathbb{N}\}\,,$$

where each of the functions $Q_{\bar{s}}$ satisfies the conditions mentioned above, and each of the sets $E_{\bar{s}}$ is finite. Consequently, the set $C \cap B$ is a finite union of semi-polynomial sets of degree d and thus also a semi-polynomial set of degree d.

The set $A \cap B$, being the union of the finite set F and the semi-polynomial set $C \cap B$, is also semi-polynomial of degree d, namely

$$A \cap B = F \cup \bigcup_{\substack{\bar{s} \in \{0,\ldots,g-1\}^r \\ \bar{s} \text{ is a solution of (5.27)}}} (E_{\bar{s}} \cup \{Q_{\bar{s}}(l_1, \ldots, l_r) \mid l_1, \ldots, l_r \in \mathbb{N}\})\,. \tag{5.32}$$

Since the emptiness problem for semi-polynomial sets is decidable, the emptiness problem for $A \cap B$ is decidable.

Case 2 ($n > 1$). Suppose that $A \subseteq \mathbb{N}^n$ is a componentwise linear set, say

$$A = A_1 \otimes \cdots \otimes A_n \,,$$

where each of the sets $A_1, \ldots, A_n$ is a one-dimensional linear set. Also, suppose that $B \subseteq \mathbb{N}^n$ is a polynomial set of degree d, say

$$\begin{aligned} B = \{ & \bar{y}_0 + k_1\bar{y}_{1,1} + k_1^2\bar{y}_{1,2} + \cdots + k_1^{d-1}\bar{y}_{1,d-1} + k_1^d\bar{y}_{1,d} \\ & + \cdots + k_r\bar{y}_{r,1} + k_r^2\bar{y}_{r,2} + \cdots + k_r^{d-1}\bar{y}_{r,d-1} + k_r^d\bar{y}_{r,d} \\ & \mid k_1, \ldots, k_r \in \mathbb{N} \} \,, \end{aligned}$$

for some vectors $\bar{y}_0$ and $\bar{y}_{i,j}$ ($1 \leq i \leq r$, $1 \leq j \leq d$) of natural numbers. For convenience, we also represent the set B as

$$B = \left\{ \begin{pmatrix} P_1(k_1, \ldots, k_r) \\ \cdots \\ P_n(k_1, \ldots, k_r) \end{pmatrix} \;\mid\; k_1, \ldots, k_r \in \mathbb{N} \right\} ,$$

where $P_1, \ldots, P_r$ are polynomials of degree d in $k_1, \ldots, k_r$ with nonnegative integer coefficients, and where the variables $k_1, \ldots, k_r$ are not mixed (this representation of semi-polynomial sets is, in fact, equivalent to the usual representation; see also Subsection 5.3.2 below).

First, we analyze the intersection $A \cap B$ for the first component as above, using A_1 and $(B)_1$. In other words, we consider the intersection $(A_1 \otimes \mathbb{N}^{n-1}) \cap B$, where only the first component matters. If this intersection is empty, then $A \cap B$ is empty as well, and we are done. Otherwise, we obtain a representation of $(A_1 \otimes \mathbb{N}^{n-1}) \cap B$ as a semi-polynomial set of degree d similar to (5.32). Note that we must replace *all* the polynomials in $k_1, \ldots, k_r$ with the corresponding polynomials in $l_1, \ldots, l_r$.

Now, for each of the polynomial sets of degree d constituting $(A_1 \otimes \mathbb{N}^{n-1}) \cap B$, say B', we apply the procedure described in Case 1 to the intersection of $(\mathbb{N} \otimes A_2 \otimes \mathbb{N}^{n-2})$ and B'. That is, only the second component matters this time.

After n steps of this kind our whole procedure terminates with a representation of $A \cap B$ as a semi-polynomial set, giving also the information whether $A \cap B$ is empty. □

Corollary 5.21. *The emptiness problem for the intersection of a semi-polynomial set and a componentwise semi-linear set is decidable.*

Proof. The assertion follows from Theorem 5.20 since the emptiness problem for semi-polynomial sets, by Proposition 5.13, is decidable. □

5.3.2 Semi-Polynomial Sets with Mixed Variables

In the spirit of (5.9) in the proof of Proposition 5.12, we can represent each polynomial set of dimension $n \geq 1$ as

$$\left\{ \begin{pmatrix} x_1 + P_{1,1}(k_1) + \cdots + P_{1,m}(k_m) \\ \cdots \\ x_n + P_{n,1}(k_1) + \cdots + P_{n,m}(k_m) \end{pmatrix} \mid k_1, \ldots, k_m \in \mathbb{N} \right\}, \tag{5.33}$$

where $x_1, \ldots, x_n \in \mathbb{N}$, and each $P_{i,j}$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$ is a polynomial in k_j with nonnegative integer coefficients and without constants. Note that each of these polynomials is a polynomial in *one* variable. We generalize the notion of polynomial sets by replacing the polynomials $P_{i,1}, \ldots, P_{i,m}$ with a polynomial $P_i(k_1, \ldots, k_m)$ with nonnegative integer coefficients in the variables $k_1, \ldots, k_m$, for each $i = 1, \ldots, n$, thus allowing polynomials in more than one variable where the variables may be mixed.

Definition 5.22. A subset A of $\mathbb{N}^n$, $n \geq 1$, is said to be a *polynomial set with mixed variables* if there are polynomials $P_1, \ldots, P_n$ in the variables $k_1, \ldots, k_m$, for some $m \geq 0$, such that

$$A = \left\{ \begin{pmatrix} P_1(k_1, \ldots, k_m) \\ \cdots \\ P_n(k_1, \ldots, k_m) \end{pmatrix} \mid k_1, \ldots, k_m \in \mathbb{N} \right\}. \tag{5.34}$$

The polynomials $P_1, \ldots, P_n$ are called the *generators* of A. The set A is said to be a *semi-polynomial set with mixed variables* if it is a finite union of polynomial sets with mixed variables.

Note that in Definition 5.22 we have omitted the notion of the 'degree' of a (semi-)polynomial set with mixed variables, in particular due to the following reasons. First, for polynomials in several variables, we would need to agree on whether we consider the degree of a polynomial with respect to a particular variable or the degree of a polynomial with respect to all variables. Second, for our result below, it does not matter how the degree of a polynomial in several variables is defined.

Example 5.23. The set $A_{\text{prod}} = \{(x, y, xy) \mid x, y \in \mathbb{N}\}$ of Proposition 5.12 is obviously a polynomial set with mixed variables.

By Proposition 5.12, the set A_{prod} is not semi-polynomial. It follows that the class of (semi-)polynomial sets with mixed variables strictly contains the class of (semi-)polynomial sets.

We will show that the emptiness problem for the intersection of a polynomial set with mixed variables and a linear set (and thus also for the intersection of a semi-polynomial set with mixed variables and a semi-linear set) is undecidable by means of a reduction of Hilbert's tenth problem[2]:

> Given a Diophantine equation $D_{\mathrm{L}}(k_1, \ldots, k_m) = D_{\mathrm{R}}(k_1, \ldots, k_m)$, where D_{L} and D_{R} are polynomials with nonnegative integer coefficients in the variables $k_1, \ldots, k_m$, and $m \geq 0$, does it have a solution in natural numbers?

Since this problem is known to be undecidable, the particular emptiness problem of our interest is also undecidable. For a more detailed account on Hilbert's tenth problem, the reader is referred to [Mat93].

Proposition 5.24. *The emptiness problem for the intersection of a polynomial set with mixed variables and a linear set is undecidable.*

Proof. We reduce Hilbert's tenth problem to the emptiness problem for the intersection of a polynomial set with mixed variables and a linear set as follows. Since the former problem is undecidable, the latter problem is also undecidable.

First of all, we define the set

$$B_{\mathrm{equal}} := \{(x, y) \in \mathbb{N}^2 \mid x = y\}\,.$$

Obviously, B_{equal} is a linear set; it coincides with the linear set

$$\{k(1, 1) \mid k \in \mathbb{N}\}\,.$$

Now, given a Diophantine equation

$$D_{\mathrm{L}}(k_1, \ldots, k_m) = D_{\mathrm{R}}(k_1, \ldots, k_m)\,, \tag{5.35}$$

where D_{L} and D_{R} are polynomials with nonnegative integer coefficients in the variables $k_1, \ldots, k_m$, and $m \geq 0$, we define a polynomial set with mixed variables

$$A_{D_{\mathrm{L}}, D_{\mathrm{R}}} := \left\{ \begin{pmatrix} D_{\mathrm{L}}(k_1, \ldots, k_m) \\ D_{\mathrm{R}}(k_1, \ldots, k_m) \end{pmatrix} \;\middle|\; k_1, \ldots, k_m \in \mathbb{N} \right\}.$$

We will show that the Diophantine equation (5.35) has a solution in natural numbers if and only if the set $A_{D_{\mathrm{L}}, D_{\mathrm{R}}} \cap B_{\mathrm{equal}}$ is not empty.

[2] To be precise, the statement of Hilbert's tenth problem presented here is a slight modification of the original statement. Anyhow, both statements are equivalent to each other (see [Mat93, page 1]).

For the 'only if' part, suppose that the sequence $k_1, \ldots, k_m$ of natural numbers is a solution of the Diophantine equation (5.35). In other words, we have

$$D_{\mathrm{L}}(k_1, \ldots, k_m) = D_{\mathrm{R}}(k_1, \ldots, k_m)\,,$$

for these particular natural numbers $k_1, \ldots, k_m$. It follows that, by the definition of B_{equal}, the vector $(D_{\mathrm{L}}(k_1, \ldots, k_m), D_{\mathrm{R}}(k_1, \ldots, k_m))$ belongs to B_{equal}. Further, by the definition of $A_{D_{\mathrm{L}},D_{\mathrm{R}}}$, this vector also belongs to $A_{D_{\mathrm{L}},D_{\mathrm{R}}}$. Thus, we conclude that the set $A_{D_{\mathrm{L}},D_{\mathrm{R}}} \cap B_{\mathrm{equal}}$ is not empty.

For the 'if' part, suppose that the set $A_{D_{\mathrm{L}},D_{\mathrm{R}}} \cap B_{\mathrm{equal}}$ is not empty. It follows that there are some natural numbers $k_1, \ldots, k_m$ such that the vector $(D_{\mathrm{L}}(k_1, \ldots, k_m), D_{\mathrm{R}}(k_1, \ldots, k_m))$ belongs to $A_{D_{\mathrm{L}},D_{\mathrm{R}}}$ and to B_{equal}. By the definition of B_{equal}, we have

$$D_{\mathrm{L}}(k_1, \ldots, k_m) = D_{\mathrm{R}}(k_1, \ldots, k_m)\,,$$

for these particular natural numbers $k_1, \ldots, k_m$. Hence, we conclude that the Diophantine equation (5.35) indeed has a solution in natural numbers. □

Since semi-polynomial sets with mixed variables generalize polynomial sets with mixed variables and semi-linear sets generalize linear sets, we immediately obtain the following undecidability result.

Corollary 5.25. *The emptiness problem for the intersection of a semi-polynomial set with mixed variables and a semi-linear set is undecidable.*

Summary and Concluding Remarks

In this chapter, we were concerned with 2-PDA's and a generalization of semi-linear sets that we referred to as semi-polynomial sets.

In Section 5.1 we gave a brief overview of HOPDA's, and then focussed our investigation on the special case of 2-PDA's. We observed that 2-PDA's are strictly more powerful than pushdown automata both with respect to recognized languages and with respect to Parikh images.

Then, in Section 5.2 we proposed the notion of semi-polynomial sets as a generalization of the notion of semi-linear sets. We showed that 2-PDA's are powerful enough to 'generate' these sets in the sense that each semi-polynomial set is the Parikh image of the language recognized by some 2-PDA. On the other hand, it turned out that the framework of semi-polynomial sets cannot capture the Parikh images of 2-PDA's. We showed that 2-PDA's are capable of generating sets with exponential and mixed terms, which we also showed to be not semi-polynomial.

In Section 5.3 we were concerned with the emptiness problem for the intersection of a semi-polynomial and a semi-linear set. We showed two partial results regarding this problem. First, weakening the notion of semi-linear sets, which results in the so-called componentwise semi-linear sets, we obtained a decidability result, which involved some number-theoretical analysis of one-dimensional semi-linear sets. Second, strengthening the notion of semi-polynomial sets by permitting polynomials with mixed variables in the definition of polynomial sets, we obtained an undecidability result by means of a reduction of Hilbert's tenth problem, which is known to be undecidable.

Regarding the second question posed in Chapter 1, we have only taken a first step toward answering these questions for 2-PDA's. We have not yet succeeded in characterizing the Parikh images of these automata. Hence, the quest to characterize their Parikh images is still an open problem and might be an interesting topic of research. In this spirit, it might also be interesting to investigate the Parikh images of HOPDA's in general.

Regarding the third question posed in Chapter 1, we have proposed a new framework of sets of vectors of natural numbers, which might serve as candidates for the constraint sets of Parikh automata. As has been noted at the beginning of Section 5.3, the application of this new framework in the context of Parikh automata might look as follows: we use a pushdown automaton for the automata component and a semi-polynomial set for the constraint set. In this setting, the use of semi-polynomial sets as constraint sets permits recognizing languages whose Parikh images are not semi-linear. For instance, it is not difficult to construct a Parikh (finite) automaton with a semi-polynomial constraint set that recognizes the language $L_{\text{quad}} = \{a^k b^{k^2} \mid k \in \mathbb{N}\}$, which is not PPDA-recognizable (see Corollary 4.14 on page 56).

As far as the general case is concerned, however, the question whether the emptiness problem for the resulting Parikh automaton is decidable remains unsolved. For the special case where the extended Parikh image of the automata component is componentwise semi-linear, we can apply our decidability result (see Theorem 5.20 and Corollary 5.21) to obtain the decidability of the emptiness problem for the resulting Parikh automaton. Still, to date the author is not aware of any automata model whose (extended) Parikh images can be characterized within the framework of componentwise semi-linear sets.

Chapter 6

Monotonic-Counter Extension of Infinite Graphs

A recent trend in research is the model-checking of infinite-state systems. As an approach to modeling such systems, many classes of infinite graphs have been developed and extensively studied in the literature.

With regard to the application in model-checking, however, infinite graphs under consideration must be finitely representable. Accordingly, in the sequel we will almost invariably refer to finitely representable, infinite graphs when speaking about infinite graphs.

Some classes of infinite graphs that can be found in the literature are the class of pushdown graphs, prefix-recognizable graphs, synchronized rational graphs, and rational graphs. Thomas [Tho02] provided a survey of these graph classes and their properties. Moreover, the use of infinite graphs from these classes as infinite automata was discussed. In this context, the vertices of an infinite graph are viewed as the states of the corresponding infinite automaton while the edges, which are labeled with symbols from a certain input alphabet, are viewed as the transitions.

As noted in Chapter 3 (see page 17), a Parikh automaton can be seen as an automaton that is equipped with some monotonic counters. The automaton may increment these counters while reading an input symbol. In this setting, a state of the Parikh automaton, in a broader sense, must remember not only the current state of the underlying automaton, but also the sum of the vectors of natural numbers that have been collected during a computation (that is, the values of the counters).

We combine the idea of infinite automata and the idea of Parikh automata as follows. Given an infinite automaton (that is, an infinite graph), we equip a state of this automaton (that is, a vertex of this graph) with finitely many

monotonic counters, which are represented by a vector of natural numbers. In this setting, a transition (that is, an edge) may increment the values of the counters while reading an input symbol.

A fair amount of definitions and results concerning classes of infinite graphs has to be built up in order to present our main results. The first section of this chapter is devoted to these definitions and results. In the second section, we apply the idea of Parikh automata to the graph classes mentioned above, resulting in new classes of infinite graphs. Then, we study some decision problems for the graphs in these new classes. Concluding this chapter, we also compare these new classes with the well-known classes mentioned above.

6.1 Preliminaries

In general, we consider graphs whose edges are labeled with symbols from an alphabet. Note that, as in Chapter 2, by an alphabet we always mean a *finite* alphabet.

Definition 6.1. Let Σ be an alphabet. A Σ-*labeled graph* is a structure $G := (V, (E_a)_{a\in\Sigma})$, where V is the set of *vertices*, and $E_a \subseteq V \times V$, for each $a \in \Sigma$, is the set of a-*labeled edges* of G. The set of all edges of G is denoted by $E := \bigcup_{a\in\Sigma} E_a$.

In this thesis, we do not distinguish Σ-labeled graphs up to isomorphism; we consider two Σ-labeled graphs to be the same if and only if they are isomorphic to each other.

In the sequel, we briefly describe the classes of infinite graphs we have mentioned above, namely the class of pushdown graphs, prefix-recognizable graphs, synchronized rational graphs, and rational graphs. We also present some results concerning these graph classes. Our description follows the definitions and results in [Tho02], where a more comprehensive account of these graph classes can be found.

6.1.1 Classes of Infinite Graphs

Pushdown Graphs

As the name can tell, a pushdown graph is the configuration graph of an ε-free pushdown automaton. Accordingly, each vertex of a pushdown graph represents a configuration of the corresponding pushdown automaton; such a configuration comprises the current state and the current stack content (see Section 2.1). If Q is the state set of the corresponding pushdown automaton and Γ is the stack alphabet, then a configuration is a pair (q, α) where $q \in Q$

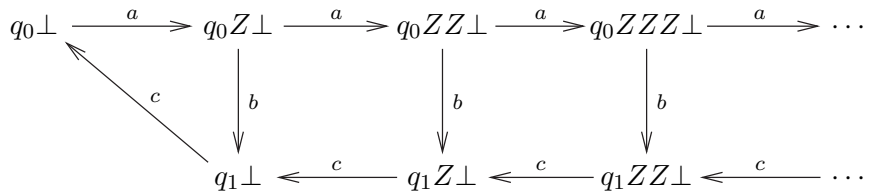


Figure 6.1: A pushdown graph

and $\alpha \in \Gamma^*$. For simplicity, we merge these two components, thus obtaining the word $q\alpha \in Q\Gamma^*$ as a configuration, which in turn represents a vertex in the pushdown graph under consideration. In this setting, there is an a-labeled edge from $p\gamma_1$ to $q\gamma_2$ if and only if $\gamma_1 = Z\alpha$, $\gamma_2 = \beta\alpha$, and $(q, \beta) \in \delta(p, a, Z)$ is a transition in the corresponding pushdown automaton.

Formally, a Σ-*labeled pushdown graph* is a graph $G = (V, (E_a)_{a\in\Sigma})$ where $V \subseteq Q\Gamma^*$ is regular, for some finite, nonempty sets Q and Γ, and

$$E_a := \{(pZ\alpha, q\beta\alpha) \in V \times V \mid (q, \beta) \in \delta(p, a, Z), \text{ where } p, q \in Q \text{ and } Z \in \Gamma \text{ and } \alpha, \beta \in \Gamma^*\},$$

for each $a \in \Sigma$, where δ is the transition function of an ε-free pushdown automaton over Σ with set of states Q and stack alphabet Γ (we suppress the role of the initial state, the initial stack symbol, and the set of final states since we do not need them in this context). The class of Σ-labeled pushdown graphs is denoted by $\mathcal{G}_{\mathrm{PD}}(\Sigma)$ or $\mathcal{G}_{\mathrm{PD}}$, whenever Σ is clear from the context.

Example 6.2. We consider the pushdown graph defined by the following ε-free pushdown automaton. Let $\Sigma := \{a, b, c\}$, $Q := \{q_0, q_1\}$, and $\Gamma := \{\bot, Z\}$. The set of vertices is given by the regular expression $V := QZ^*\bot$, and the set of edges is defined by the transition function δ with

$$\begin{aligned} \delta(q_0, a, \bot) &:= \{(q_0, Z\bot)\}, & \delta(q_1, c, \bot) &:= \{(q_0, \bot)\}, \\ \delta(q_0, a, Z) &:= \{(q_0, ZZ)\}, & \delta(q_1, c, Z) &:= \{(q_1, \varepsilon)\}, \\ \delta(q_0, b, Z) &:= \{(q_1, \varepsilon)\}. \end{aligned}$$

Figure 6.1 illustrates the resulting pushdown graph.

Prefix-Recognizable Graphs

The notion of prefix-recognizable graphs generalizes the notion of pushdown graphs as follows. An a-labeled edge $(pZ\alpha, q\beta\alpha)$ of a pushdown graph with

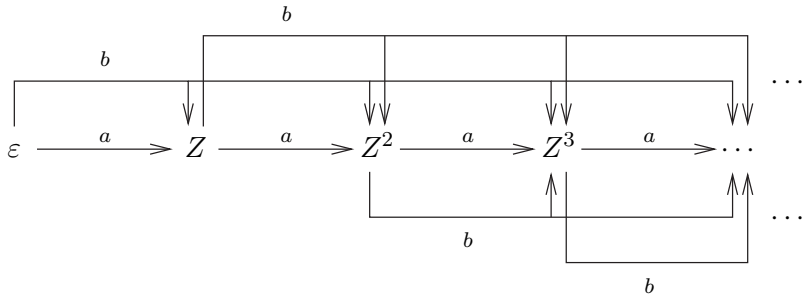


Figure 6.2: A prefix-recognizable graph

state set Q and stack alphabet Γ represents a prefix rewriting; the prefix pZ is rewritten to the word $q\beta$. Now, if we suppress the role of Q, then such an edge represents a prefix rewriting on words over Γ.

Let Γ be an alphabet. A (Σ-*labeled*) *prefix rewriting rule* over Γ is of the form $\alpha \xrightarrow{a} \beta$ where $\alpha, \beta \in \Gamma^*$ and $a \in \Sigma$. A (Σ-*labeled*) *generalized prefix rewriting rule* over Γ is of the form $R \xrightarrow{a} S$ where $R, S \subseteq \Gamma^*$ are regular and $a \in \Sigma$. A (Σ-*labeled*) *prefix rewriting system* over Γ is a *finite* set $\mathfrak{R}$ of generalized prefix rewriting rules over Γ. In the following, we will omit the reference to Σ whenever Σ is clear from the context.

A Σ-*labeled prefix-recognizable graph* is a graph $G = (V, (E_a)_{a\in\Sigma})$ where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and for each $a \in \Sigma$,

$$E_a := \{(\alpha\gamma, \beta\gamma) \in V \times V \mid \alpha \in R, \beta \in S, \text{ and } \gamma \in \Gamma^*, \text{ for some } R \xrightarrow{a} S \in \mathfrak{R}\},$$

for some prefix rewriting system $\mathfrak{R}$ over Γ. The class of Σ-labeled prefix-recognizable graphs is denoted by $\mathcal{G}_{\mathrm{PR}}(\Sigma)$ or $\mathcal{G}_{\mathrm{PR}}$, whenever Σ is clear from the context.

Example 6.3. We consider the following prefix-recognizable graph. Let $\Sigma := \{a, b\}$ and $\Gamma := \{Z\}$. Furthermore, let $\mathfrak{R} := \{\varepsilon \xrightarrow{a} Z, \varepsilon \xrightarrow{b} Z^+\}$ be a prefix-rewriting system. Then, the prefix-recognizable graph defined by $\mathfrak{R}$ is the graph $G = (V, E_a, E_b)$ depicted in in Figure 6.2 with

- $V = \Gamma^*$,
- $E_a = \{(Z^i, Z^{i+1}) \mid i \in \mathbb{N}\}$, and
- $E_b = \{(Z^i, Z^j) \mid i, j \in \mathbb{N} \text{ and } i < j\}$.

Synchronized Rational Graphs

The main concept concerning synchronized rational graphs is the notion of the so-called synchronized rational relations, which will be used to represent edges of graphs from this class.

Let Γ be an alphabet, and let $\square$ be a new symbol. We assign to two arbitrary words $\alpha = X_1 \cdots X_m$ and $\beta = Y_1 \cdots Y_n$ over Γ the word

$$\alpha \hat{} \beta := \begin{cases} \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} \cdots \begin{bmatrix} X_m \\ Y_m \end{bmatrix} & \text{if } m = n, \\ \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} \cdots \begin{bmatrix} X_m \\ Y_m \end{bmatrix} \begin{bmatrix} \square \\ Y_{m+1} \end{bmatrix} \cdots \begin{bmatrix} \square \\ Y_n \end{bmatrix} & \text{if } m < n, \\ \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} \cdots \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \begin{bmatrix} X_{n+1} \\ \square \end{bmatrix} \cdots \begin{bmatrix} X_m \\ \square \end{bmatrix} & \text{if } m > n, \end{cases}$$

over $(\Gamma \cup \{\square\}) \times (\Gamma \cup \{\square\})$. Furthermore, we assign to a relation $R \subseteq \Gamma^* \times \Gamma^*$ the language

$$L_R := \{\alpha \hat{} \beta \mid (\alpha, \beta) \in R\}$$

over $(\Gamma \cup \{\square\}) \times (\Gamma \cup \{\square\})$. The relation R is called *synchronized rational* if L_R is regular or recognizable by a finite automaton, equivalently. Intuitively, a finite automaton recognizing L_R can be seen as a one-way finite automaton which has two input tapes, each with its own input head, but these input heads may only be moved simultaneously (or in other words, synchronously). The special symbol $\square$ is needed to cover the case where the two words under consideration are of different length.

A Σ-*labeled synchronized rational graph* is a graph $G = (V, (E_a)_{a \in \Sigma})$ where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and E_a is synchronized rational, for each $a \in \Sigma$. The class of Σ-labeled synchronized rational graphs is denoted by $\mathcal{G}_{\mathrm{SR}}(\Sigma)$ or $\mathcal{G}_{\mathrm{SR}}$, whenever Σ is clear from the context.

Example 6.4. We consider the *infinite two-dimensional grid*, which is depicted in Figure 6.3. Let $\Sigma := \{a, b\}$ and $\Gamma := \{X, Y\}$.

- The set of vertices is defined by $V := X^*Y^*$.
- The set of a-labeled edges is defined by

$$E_a := \{(X^i Y^j, X^{i+1} Y^j) \mid i, j \in \mathbb{N}\},$$

  which is a synchronized rational relation since

$$L_{E_a} = \begin{bmatrix} X \\ X \end{bmatrix}^* \left(\begin{bmatrix} \square \\ X \end{bmatrix} + \begin{bmatrix} Y \\ X \end{bmatrix} \begin{bmatrix} Y \\ Y \end{bmatrix}^* \begin{bmatrix} \square \\ Y \end{bmatrix} \right).$$

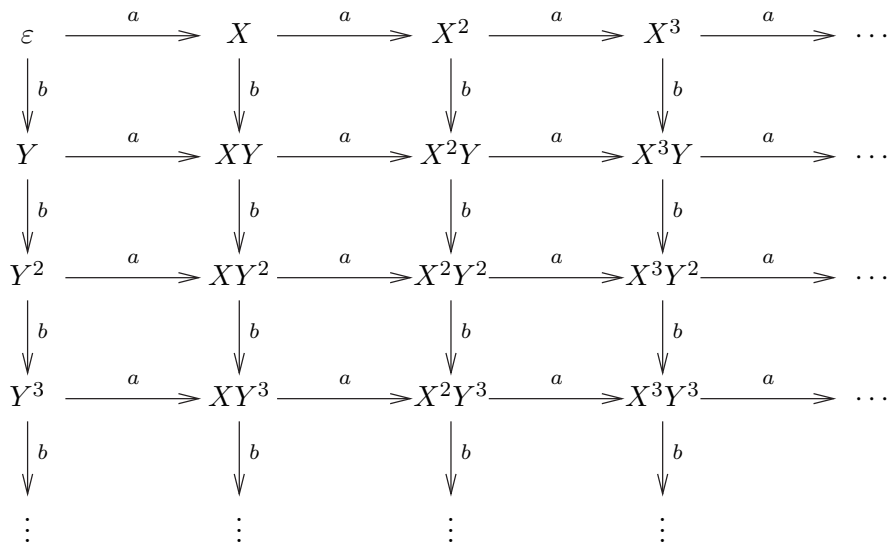


Figure 6.3: Infinite two-dimensional grid

- The set of b-labeled edges is defined by

$$E_b := \{(X^iY^j, X^iY^{j+1}) \mid i, j \in \mathbb{N}\},$$

which is a synchronized rational relation since

$$L_{E_b} = \begin{bmatrix} X \\ X \end{bmatrix}^* \begin{bmatrix} Y \\ Y \end{bmatrix}^* \begin{bmatrix} \square \\ Y \end{bmatrix}.$$

Rational Graphs

As we are not going to work with rational graphs directly, we will introduce rational graphs only informally.

The notion of rational relations generalizes the notion of synchronized rational relations. As mentioned before, a synchronized rational relation can be characterized by a one-way finite automaton with two input tapes, each with its own input head. Whereas for a synchronized rational relation the input heads may only be moved simultaneously (synchronously), for a rational relation the input heads may be moved asynchronously, that is, independently from each other.

A Σ-*labeled rational graph* is a graph $G = (V, (E_a)_{a\in\Sigma})$ where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and E_a is rational, for each $a \in \Sigma$. The class of

Σ-labeled rational graphs is denoted by $\mathcal{G}_{\mathrm{Rat}}(\Sigma)$ or $\mathcal{G}_{\mathrm{Rat}}$, whenever Σ is clear from the context.

6.1.2 Results on Classes of Infinite Graphs

First of all, it is worth noting that the class of Σ-labeled finite graphs (denoted by $\mathcal{G}_{\mathrm{Fin}}(\Sigma)$ or $\mathcal{G}_{\mathrm{Fin}}$, whenever Σ is clear from the context) and the classes of graphs presented above, in the order they were presented, form a strictly increasing hierarchy of graph classes (see [Tho02, Theorem 1]). In Section 6.4 we will extend this hierarchy to the new graph classes to be presented in Section 6.2.

Decision Problems

In the context of model-checking, an interesting decision problem concerning graphs is, of course, the reachability problem.

Further, more general questions concerning graphs may be formulated within a logical formalism. In this sense, a graph G is viewed as a relational structure. Then, the decision problem of interest is: "Given a logical formula φ, does φ hold in this structure?"

Regarding these aspects, we will focus on the following decision problems:

- Reachability: Given a graph G and two vertices α and β in G, is β reachable from α?
- First-order (FO) theory: Given a graph G and a first-order sentence φ, does φ hold in G?
- Monadic second-order (MSO) theory: Given a graph G and a monadic second-order sentence φ, does φ hold in G?

Table 6.1 summarizes the results concerning the classes of infinite graphs presented above regarding these decision problems (see [Tho02, Section 3]). In this table, for a graph class, with regard to a decision problem, the entry 'undecidable' means that *there exists* some graph in this class for which this problem is undecidable; it does *not* necessarily mean that this problem is undecidable *for all* graphs in this class

Later, we will need a result concerning the infinite two-dimensional grid of Example 6.4 (see also Figure 6.3), which we state in the following theorem. For a proof, the reader is referred to [Tho02, Theorem 10].

Theorem 6.5. *The monadic second-order theory of the infinite two-dimensional grid is undecidable.*

Table 6.1: Decision problems for infinite graphs (see [Tho02, Section 3])

Infinite graphs	Reachability	FO theory	MSO theory
$\mathcal{G}_{\mathrm{PD}}$	decidable	decidable	decidable
$\mathcal{G}_{\mathrm{PR}}$	decidable	decidable	decidable
$\mathcal{G}_{\mathrm{SR}}$	undecidable	decidable	undecidable
$\mathcal{G}_{\mathrm{Rat}}$	undecidable	undecidable	undecidable

Traces

One aspect that turns out to be useful in the analysis of graphs whose edges are labeled with symbols from a certain alphabet is the notion of traces.

In such a graph, each finite path forms a finite word over the alphabet of the edge labels of this graph. Thus, if we declare some vertices as initial and some as final, then the underlying graph can be seen as an *infinite automaton* on words over the alphabet of the edge labels. Such a word, called a *trace*, is accepted if and only if there is a finite path from an initial vertex to a final vertex that is labeled with this word.

The results on the traces of the graphs from the classes presented above are summarized in the following (see [Tho02, Theorem 5 and Theorem 6]):

- The traces of finite graphs precisely yield the regular languages.
- The traces of pushdown graphs and of prefix-recognizable graphs (with regular sets of initial and final vertices) precisely yield the context-free languages.
- The traces of synchronized rational graphs and of rational graphs (with regular sets of initial and final vertices) precisely yield the context-sensitive languages.

6.2 Monotonic-Counter Graphs

We apply the idea of Parikh automata, first, by using an extended alphabet of the form $\Sigma \times D$ instead of Σ for the labeling of edges, for some finite, nonempty set of vectors D. As a result, we obtain a $(\Sigma \times D)$-labeled graph. If we now take into consideration the vectors that are *accumulated along a path* and code them as monotonic counters (represented by vectors of natural numbers) in the vertices, the vector component of any edge label can be reconstructed from the vectors of the vertices that are incident on this edge. Thus, we may omit the vector component of the edge labels. In this way, we get back to a Σ-labeled graph.

Definition 6.6. Let Σ be an alphabet, and let D (called the *auxiliary set*) be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $G := (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$ be a $(\Sigma \times D)$-labeled graph. The Σ-*labeled monotonic-counter extension* (*of dimension* n) of G is the Σ-labeled graph $\tilde{G} := (\tilde{V}, (\tilde{E}_a)_{a\in\Sigma})$ with

$$\tilde{V} := V \times \mathbb{N}^n ,$$

and

$$\tilde{E}_a := \{((\alpha, \bar{x}), (\beta, \bar{y})) \in \tilde{V} \times \tilde{V} \mid (\alpha, \beta) \in E_{(a,\bar{d})} \text{ and } \bar{y} = \bar{x} + \bar{d}, \text{ for some } \bar{d} \in D\} ,$$

for each $a \in \Sigma$. The set of all edges of $\tilde{G}$ is denoted by $\tilde{E} := \bigcup_{a\in\Sigma} \tilde{E}_a$.

The class of the Σ-labeled monotonic-counter extensions of Σ-labeled finite graphs is denoted by $\mathcal{G}_{\mathrm{FMC}}(\Sigma)$. Analogously, $\mathcal{G}_{\mathrm{PDMC}}(\Sigma)$, $\mathcal{G}_{\mathrm{PRMC}}(\Sigma)$, $\mathcal{G}_{\mathrm{SRMC}}(\Sigma)$, and $\mathcal{G}_{\mathrm{RMC}}(\Sigma)$ denote the class of the Σ-labeled monotonic-counter extensions of Σ-labeled pushdown graphs, prefix-recognizable graphs, synchronized rational graphs, and rational graphs, respectively. We will omit the reference to Σ whenever Σ is clear from the context.

In the sequel, we will refer to a graph that is obtained by a monotonic-counter extension simply as a 'monotonic-counter graph.'

Remark 6.7. (a) The graphs from the classes $\mathcal{G}_{\mathrm{FMC}}$, $\mathcal{G}_{\mathrm{PDMC}}$, $\mathcal{G}_{\mathrm{PRMC}}$, $\mathcal{G}_{\mathrm{SRMC}}$, and $\mathcal{G}_{\mathrm{RMC}}$ are finitely representable, namely by means of the finite representation of the underlying graph and by means of the finite auxiliary set of vectors.

(b) The class $\mathcal{G}_{\mathrm{Rat}}$ is contained in the class $\mathcal{G}_{\mathrm{RMC}}$; every rational graph can be represented as the monotonic-counter extension of a rational graph where the counters are never incremented. Analogously, we have $\mathcal{G}_{\mathrm{SR}} \subseteq \mathcal{G}_{\mathrm{SRMC}}$, $\mathcal{G}_{\mathrm{PR}} \subseteq \mathcal{G}_{\mathrm{PRMC}}$, $\mathcal{G}_{\mathrm{PD}} \subseteq \mathcal{G}_{\mathrm{PDMC}}$, and $\mathcal{G}_{\mathrm{Fin}} \subseteq \mathcal{G}_{\mathrm{FMC}}$.

(c) By definition, the classes of monotonic-counter graphs presented above constitute, in the order they were presented, an increasing chain of inclusions.

Example 6.8. Let $\Sigma := \{a, b\}$, $n = 2$, and $D := \{(1, 0), (0, 1)\}$. We define a $(\Sigma \times D)$-labeled finite graph $G_{\mathrm{grid}} = (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$ with

- $V := \{\bullet\}$,
- $E_{(a,(1,0))} = E_{(b,(0,1))} := \{(\bullet, \bullet)\}$, and
- $E_{(a,(0,1))} = E_{(b,(1,0))} := \emptyset$.

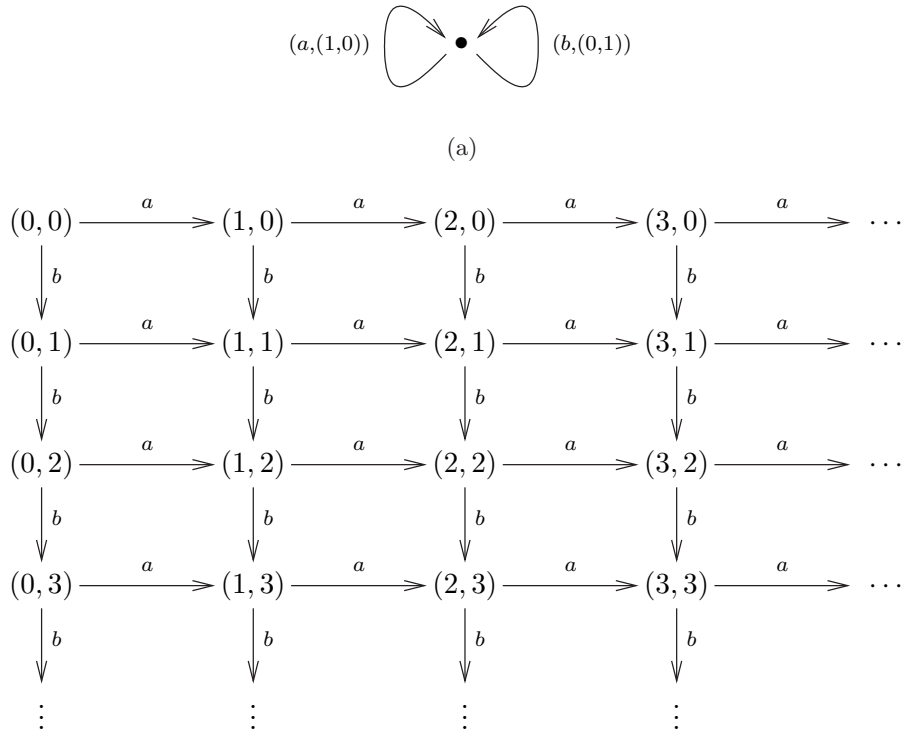


Figure 6.4: Graphs of Example 6.8

The monotonic-counter extension of G_{grid} is the graph $\tilde{G}_{\text{grid}} = (\tilde{V}, \tilde{E}_a, \tilde{E}_b)$ with

- $\tilde{V} := \{(\bullet, (i,j)) \mid i,j \in \mathbb{N}\}$,
- $E_a := \{((\bullet, (i,j)), (\bullet, (i+1,j))) \mid i,j \in \mathbb{N}\}$, and
- $E_b := \{((\bullet, (i,j)), (\bullet, (i,j+1))) \mid i,j \in \mathbb{N}\}$.

The graph G_{grid} is depicted in Figure 6.4(a), and the graph $\tilde{G}_{\text{grid}}$ is depicted in Figure 6.4(b), where, for better readability, the symbol $\bullet$ has been omitted.

6.3 Decision Problems

We consider the counterparts of the decision problems mentioned in Section 6.1 for classes of monotonic-counter graphs. Table 6.2 summarizes our results.

Table 6.2: Decision problems for monotonic-counter graphs

Monotonic-counter graphs	Reachability	FO theory	MSO theory
$\mathcal{G}_{\mathrm{FMC}}$	decidable	decidable	undecidable
$\mathcal{G}_{\mathrm{PDMC}}$	decidable	decidable	undecidable
$\mathcal{G}_{\mathrm{PRMC}}$	decidable	decidable	undecidable
$\mathcal{G}_{\mathrm{SRMC}}$	undecidable	decidable	undecidable
$\mathcal{G}_{\mathrm{RMC}}$	undecidable	undecidable	undecidable

Reachability

By Remark 6.7(b), the classes $\mathcal{G}_{\mathrm{SRMC}}$ and $\mathcal{G}_{\mathrm{RMC}}$ subsume the classes $\mathcal{G}_{\mathrm{SR}}$ and $\mathcal{G}_{\mathrm{Rat}}$, respectively. Consequently, we immediately obtain an undecidability result concerning the reachability problem for graphs in the former two classes.

Proposition 6.9. *There is a graph in the class $\mathcal{G}_{\mathrm{SRMC}}$ (and thus also in the class $\mathcal{G}_{\mathrm{RMC}}$) for which the reachability problem is undecidable.*

For the graphs in the class $\mathcal{G}_{\mathrm{PRMC}}$ (and thus also in the classes $\mathcal{G}_{\mathrm{PDMC}}$, and $\mathcal{G}_{\mathrm{FMC}}$), we obtain a decidability result for the following, more general reachability problem.

The reachability problem stated in Section 6.1 can easily be adapted to the case of the monotonic-counter extensions of prefix-recognizable graphs. Let Σ and Γ be alphabets. Furthermore, let $n \geq 1$, and let D be a finite, nonempty subset of $\mathbb{N}^n$. Let G be a $(\Sigma \times D)$-labeled prefix-recognizable graph with regular set $V \subseteq \Gamma^*$ of vertices. Then, the reachability problem for $\tilde{G}$, the monotonic-counter extension G, is the question:

> Given two regular sets $U, U' \subseteq V$ of vertices in G, and given two semi-linear sets $C, C' \subseteq \mathbb{N}^n$, are there vertices $(\alpha, \bar{x}) \in U \times C$ and $(\beta, \bar{y}) \in U' \times C'$ in $\tilde{G}$ such that $(\beta, \bar{y})$ is reachable from $(\alpha, \bar{x})$?

Toward our result, we will need the following lemma, which provides a *constructive* proof of the fact that the traces of prefix-recognizable graphs are context-free. This fact was stated without proof in Section 6.1 and, also without proof, in [Tho02]. However, for our decidability result, we need an effective procedure which, given a prefix-recognizable graph, returns a pushdown automaton or a context-free grammar for the traces of this graph as its output.

As a remark, the following lemma is actually a corollary of the fact that each prefix-recognizable graph is the ε-closure of the configuration graph of a pushdown automaton. Leucker [Leu02] pointed out that the latter result is

due to Stirling [Sti00] and provided a constructive proof of this result. In fact, the method used in the following lemma is very similar to the proof provided by Leucker.

Lemma 6.10. *Let Σ and Γ be alphabets, and let $G = (V, (E_a)_{a\in\Sigma})$ be a Σ-labeled prefix-recognizable graph defined by the prefix rewriting system $\mathfrak{R}$ over Γ. Furthermore, let $Init, Fin \subseteq \Gamma^*$ be regular sets of vertices in G. Then, the traces of G, with $Init$ as the set of initial vertices and Fin as the set of final vertices, yield effectively a context-free language.*

Proof. Suppose that each $U \in \{R, S \mid R \xrightarrow{a} S \in \mathfrak{R}\} \cup \{Init, Fin\}$ is given by a finite automaton $\mathfrak{A}^U := (Q^U, \Gamma, \delta^U, q_0^U, F^U)$. Without loss of generality, we assume that all these automata have pairwise disjoint sets of states.

We will construct a pushdown automaton $\mathfrak{A}$ over Σ that recognizes the traces of G, with *Init* as the set of initial vertices and *Fin* as the set of final vertices. We use the state set

$$Q := \bigcup_{R\xrightarrow{a}S\in\mathfrak{R}} (Q^R \cup Q^S) \cup Q^{Init} \cup Q^{Fin} \cup \{q_0, q_{\mathrm{run}}, q_{\mathrm{acc}}\},$$

the stack alphabet $\Gamma \cup \{\bot\}$, the initial state q_0, the initial stack symbol $\bot$, and the set $\{q_{\mathrm{acc}}\}$ of final states. Note that q_0, q_{run}, and q_{acc} are new states and that $\bot$ is a new stack symbol. Intuitively, we will use $\mathfrak{A}$ to simulate the prefix rewriting system $\mathfrak{R}$:

- Given an input word $w := a_1 \cdots a_m \in \Sigma^*$, we must first guess an initial vertex $\alpha \in Init$ and store it in the stack as $\alpha\bot$. Note that, for the guessing, we must simulate $\mathfrak{A}^{Init}$ *backwards* since the leftmost symbol of $\alpha\bot$ is considered as the topmost symbol of the stack.
- Now, for the first input symbol $a_1 \in \Sigma$, we guess a generalized prefix rewriting rule $R \xrightarrow{a_1} S$ and apply it to α (which is stored in the stack as $\alpha\bot$) as follows. Suppose that $\alpha = \alpha'\gamma$, for some $\alpha' \in R$ and $\gamma \in \Gamma^*$. Simulating $\mathfrak{A}^R$, we remove the prefix α' from the stack, resulting in the stack $\gamma\bot$. Then, we guess some word $\alpha'' \in S$ and store it in the stack, resulting in the stack $\alpha''\gamma\bot$. Again, for the guessing, we must simulate $\mathfrak{A}^S$ *backwards*.
- We repeat the latter steps for each of the remaining input symbols $a_2, \ldots, a_m$ successively.
- Suppose that at some point of the computation the input symbols have been used up and the stack content is $\beta\bot$, for some $\beta \in \Gamma^*$. We now verify that β belongs to *Fin*, by simulating $\mathfrak{A}^{Fin}$, and then go to an accepting configuration.

Roughly speaking, during the computation the stack, which contains a word over Γ, serves as a pseudo-input tape for the finite automaton $\mathfrak{A}^U$ above, for $U \in \{R, S \mid R \xrightarrow{a} S \in \mathfrak{R}\} \cup \{Init, Fin\}$.

Formally, we define the transition function δ of $\mathfrak{A}$ as follows.

Starting in configuration $(q_0, \bot)$, we first guess a final state $q \in F^{Init}$ of $\mathfrak{A}^{Init}$, by setting

$$(q, \bot) \in \delta(q_0, \varepsilon, \bot)\,,$$

for each $q \in F^{Init}$, and then simulate $\mathfrak{A}^{Init}$ *backwards*, by setting

$$(p, XZ) \in \delta(q, \varepsilon, Z) \quad \text{if} \quad q \in \delta^{Init}(p, X)\,,$$

for each $p, q \in Q^{Init}$, $X \in \Gamma$, and $Z \in \Gamma \cup \{\bot\}$. After seeing the initial state q_0^{Init} of $\mathfrak{A}^{Init}$ we go to state q_{run}, by setting

$$(q_{\mathrm{run}}, Z) \in \delta(q_0^{Init}, \varepsilon, Z)\,,$$

for each $Z \in \Gamma \cup \{\bot\}$.

Being in state q_{run}, we now guess a generalized prefix rewriting rule $R \xrightarrow{a} S \in \mathfrak{R}$ and add the following transitions to δ (note that we have to add the following transitions for *each* rule $R \xrightarrow{a} S \in \mathfrak{R}$). First, we go to the initial state q_0^R of $\mathfrak{A}^R$, by setting

$$(q_0^R, Z) \in \delta(q_{\mathrm{run}}, \varepsilon, Z)\,,$$

for each $Z \in \Gamma \cup \{\bot\}$, and simulate $\mathfrak{A}^R$, by setting

$$(q, \varepsilon) \in \delta(p, \varepsilon, X) \quad \text{if} \quad q \in \delta^R(p, X)\,,$$

for each $p, q \in Q^R$ and $X \in \Gamma$. If a final state of $\mathfrak{A}^R$ has been reached, we guess a final state $q \in F^S$ of $\mathfrak{A}^S$, by setting

$$(q, Z) \in \delta(p, \varepsilon, Z)\,,$$

for each $p \in F^R$ and $q \in F^S$, and then simulate $\mathfrak{A}^S$ *backwards*, by setting

$$(p, XZ) \in \delta(q, \varepsilon, Z) \quad \text{if} \quad q \in \delta^S(p, X)\,,$$

for each $p, q \in Q^S$, $X \in \Gamma$, and $Z \in \Gamma \cup \{\bot\}$. If the initial state q_0^S of $\mathfrak{A}^S$ has been reached, we go back to state q_{run}, while reading the input symbol $a \in \Sigma$, thus finishing the application of the rule $R \xrightarrow{a} S$; we set

$$(q_{\mathrm{run}}, Z) \in \delta(q_0^S, a, Z)\,,$$

for each $Z \in \Gamma \cup \{\bot\}$.

Now, at some point of the computation we may guess that the input symbols have been used up. In this case, from state q_{run} we go to the initial state q_0^{Fin} of $\mathfrak{A}^{Fin}$, by setting

$$(q_0^{Fin}, Z) \in \delta(q_{\mathrm{run}}, \varepsilon, Z)\,,$$

for each $Z \in \Gamma \cup \{\bot\}$. Then, simulating $\mathfrak{A}^{Fin}$, we empty the stack except for the initial stack symbol $\bot$, by setting

$$(q, \varepsilon) \in \delta(p, \varepsilon, X) \quad \text{if} \quad q \in \delta^{Fin}(p, X)\,,$$

for each $p, q \in Q^{Fin}$ and $X \in \Gamma$. If a final state of $\mathfrak{A}^{Fin}$ has been reached and at this point the stack only contains the initial stack symbol $\bot$, then we go to the final state q_{acc}, by setting

$$(q_{\mathrm{acc}}, \varepsilon) \in \delta(q, \varepsilon, \bot)\,,$$

for each $q \in F^{Fin}$, and thus accepts. □

We are now ready to prove the decidability of the reachability problem for the monotonic-counter extensions of prefix-recognizable graphs. Our approach will be to reduce this problem to the emptiness problem for semi-linear sets, which is known to be decidable.

Theorem 6.11. *The reachability problem for the graphs in the class $\mathcal{G}_{\mathrm{PRMC}}$ (and thus also in the classes $\mathcal{G}_{\mathrm{FMC}}$ and $\mathcal{G}_{\mathrm{PDMC}}$) is decidable.*

Proof. Let Σ be an alphabet. Furthermore, let $n \geq 1$, and let D be a finite, nonempty subset of $\mathbb{N}^n$. Let $\tilde{G}$ be the monotonic-counter extension of the prefix-recognizable graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$, where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ.

Now, let $U, U' \subseteq \Gamma^*$ be regular sets of vertices in G, and let $C, C' \subseteq \mathbb{N}^n$ be semi-linear. By Lemma 6.10 the traces of G, with U as the set of initial vertices and U' as the set of final vertices, yield *effectively* a context-free language L over $\Sigma \times D$. Then, by Parikh's theorem and Lemma 3.19, the extended Parikh image of L (denoted by $\tilde{\Phi}(L)$) is effectively semi-linear.

We will show that

$$\begin{aligned}&\text{there are some vertices } (\alpha, \bar{x}) \in U \times C \text{ and } (\beta, \bar{y}) \in U' \times C' \\ &\text{in } \tilde{G} \text{ such that } (\beta, \bar{y}) \text{ is reachable from } (\alpha, \bar{x})\end{aligned} \tag{6.1}$$

if and only if

$$(C + \tilde{\Phi}(L)) \cap C' \neq \emptyset\,. \tag{6.2}$$

By the effective closure of the class of semi-linear sets under addition and intersection (see Chapter 2), the set $(C + \tilde{\Phi}(L)) \cap C'$ is effectively semi-linear, and thus, the emptiness problem for this set is decidable. Consequently, it is decidable whether the statement (6.1) holds, which in turn implies the decidability of the reachability problem for $\tilde{G}$.

Suppose that the statement (6.1) holds. In other words, there is a Σ-labeled path in $\tilde{G}$ from $(\alpha, \bar{x})$ to $(\beta, \bar{y})$. Formally, there are some edge labels $a_1, \ldots, a_m \in \Sigma$ and some vertices $(\alpha_0, \bar{x}_0), \ldots, (\alpha_m, \bar{x}_m)$ in $\tilde{G}$, for some $m \geq 0$, such that

- $(\alpha_0, \bar{x}_0) = (\alpha, \bar{x})$,
- $((\alpha_{i-1}, \bar{x}_{i-1}), (\alpha_i, \bar{x}_i)) \in \tilde{E}_{a_i}$, for $i = 1, \ldots, m$, and
- $(\alpha_m, \bar{x}_m) = (\beta, \bar{y})$.

Then, by the definition of $\tilde{G}$, there are some vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ such that

$$(\alpha_{i-1}, \alpha_i) \in E_{(a_i, \bar{d}_i)} \quad \text{and} \quad \bar{x}_{i-1} + \bar{d}_i = \bar{x}_i \,, \tag{6.3}$$

for $i = 1, \ldots, m$. That is, there is a path in G from α to β, and the edge labels of this path constitute the word

$$w := (a_1, \bar{d}_1) \cdots (a_m, \bar{d}_m) \in (\Sigma \times D)^* \,.$$

Since $\alpha \in U$ and $\beta \in U'$, the word w is a trace in G that belongs to the language L. It follows that $\tilde{\Phi}(w)$ belongs $\tilde{\Phi}(L)$, and since $\bar{x}$ belongs to C, $\bar{x} + \tilde{\Phi}(w)$ belongs to $C + \tilde{\Phi}(L)$. Further, by the fact that $\bar{x} = \bar{x}_0$ and $\tilde{\Phi}(w) = \bar{d}_1 + \cdots + \bar{d}_m$ and by (6.3), we obtain

$$\begin{aligned} \bar{x} + \tilde{\Phi}(w) &= \bar{x}_0 + \bar{d}_1 + \cdots + \bar{d}_m \\ &\overset{(6.3)}{=} \bar{x}_1 + \bar{d}_2 + \cdots + \bar{d}_m \\ &\overset{(6.3)}{=} \bar{x}_2 + \bar{d}_3 + \cdots + \bar{d}_m \\ &\overset{(6.3)}{=} \cdots \\ &\overset{(6.3)}{=} \bar{x}_{m-1} + \bar{d}_m \\ &\overset{(6.3)}{=} \bar{x}_m \,. \end{aligned} \tag{6.4}$$

Since $\bar{x}_m = \bar{y}$ and $\bar{y}$ belongs to C', it follows that $\bar{x} + \tilde{\Phi}(w)$ belongs to C'. Hence, we conclude that the set $(C + \tilde{\Phi}(L)) \cap C'$ is not empty.

Conversely, suppose that $(C + \tilde{\Phi}(L)) \cap C'$ is not empty. Then, there are some vectors $\bar{x} \in C$ and $\bar{y} \in C'$ and some word $w := (a_1, \bar{d}_1) \cdots (a_m, \bar{d}_m) \in L$

such that $\bar{x} + \tilde{\Phi}(w) = \bar{y}$. Since $w \in L$, there is a path from some vertex $\alpha \in U$ to some vertex $\beta \in U'$, and the edge labels of this path constitute w. That is, there are some vertices $\alpha_0, \ldots, \alpha_m$ in G such that

- $\alpha_0 = \alpha$,
- $(\alpha_{i-1}, \alpha_i) \in E_{(a_i, \bar{d}_i)}$, for $i = 1, \ldots, m$, and
- $\alpha_m = \beta$.

We will construct a path in $\tilde{G}$ from $(\alpha, \bar{x})$ to $(\beta, \bar{y})$, thus showing (6.1). Now, we define the vectors

$$\bar{x}_0 := \bar{x} \quad \text{and} \quad \bar{x}_i := \bar{x}_{i-1} + \bar{d}_i \,,$$

for $i = 1, \ldots, m$. From this definition, it is not difficult to verify that $\bar{x}_m = \bar{y}$ (by using an analysis similar to (6.4) above). Further, by the definition of $\tilde{G}$, we have

$$((\alpha_{i-1}, \bar{x}_{i-1}), (\alpha_i, \bar{x}_i)) \in \tilde{E}_{a_i} \,,$$

for each $i = 1, \ldots, m$. It is now straightforward to see that the vertices $(\alpha_0, \bar{x}_0), \ldots, (\alpha_m, \bar{x}_m)$ in $\tilde{G}$ constitute a path from $(\alpha, \bar{x})$ to $(\beta, \bar{y})$. □

First-Order and Monadic Second-Order Theory

It should be clear that the graph $\tilde{G}_{\text{grid}}$ of Example 6.8 is isomorphic to the infinite two-dimensional grid of Example 6.4. Since the monadic second-order theory of the infinite two-dimensional grid is undecidable (see Theorem 6.5), we immediately obtain the following result.

Proposition 6.12. *The class $\mathcal{G}_{\text{FMC}}$ (and thus also the classes $\mathcal{G}_{\text{PDMC}}$, $\mathcal{G}_{\text{PRMC}}$, $\mathcal{G}_{\text{SRMC}}$, and $\mathcal{G}_{\text{RMC}}$) contains a graph whose monadic second-order theory is undecidable.*

By Remark 6.7(b), the class $\mathcal{G}_{\text{RMC}}$ subsumes the class $\mathcal{G}_{\text{Rat}}$. Thus, the undecidability result on the first-order theory of the class $\mathcal{G}_{\text{Rat}}$ carries over to the class $\mathcal{G}_{\text{RMC}}$.

Proposition 6.13. *There is graph in the class $\mathcal{G}_{\text{RMC}}$ whose first-order theory is undecidable.*

In the following section we will show that the monotonic-counter extensions of synchronized rational graphs are again synchronized rational (see Theorem 6.15). Consequently, the graphs in the classes $\mathcal{G}_{\text{SRMC}}$, $\mathcal{G}_{\text{PRMC}}$, $\mathcal{G}_{\text{PDMC}}$, and $\mathcal{G}_{\text{FMC}}$ are also synchronized rational and therefore have a decidable first-order theory.

Proposition 6.14. *The first order-theory of any graph in the classes $\mathcal{G}_{\text{FMC}}$, $\mathcal{G}_{\text{PDMC}}$, $\mathcal{G}_{\text{PRMC}}$, and $\mathcal{G}_{\text{SRMC}}$ is decidable.*

6.4 Hierarchy of Graph Classes

In this section, we relate the classes of monotonic-counter graphs with the classes of graphs presented in Section 6.1.

To begin with, we show that the monotonic-counter extension of a synchronized rational graph is synchronized rational. Together with Remark 6.7(b), this result implies that the class $\mathcal{G}_{\mathrm{SRMC}}$ is redundant; it is the same as the class of synchronized rational graphs.

Theorem 6.15. *The monotonic-counter extension of a synchronized rational graph is a synchronized rational graph.*

Proof. Let Σ be an alphabet. Furthermore, let $n \geq 1$ and let D be a finite, nonempty subset of $\mathbb{N}^n$. Let $\tilde{G} = (\tilde{V}, (\tilde{E}_a)_{a\in\Sigma})$ be the monotonic-counter extension of the synchronized rational graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$, where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and $E_{(a,\bar{d})}$ is synchronized rational, for each $(a, \bar{d}) \in \Sigma \times D$; that is, each $L_{E_{(a,\bar{d})}}$ is recognizable by a one-way finite automaton with two input tapes and synchronously moving input heads (as described in Section 6.1 on page 101), say by $\mathfrak{A}_{(a,\bar{d})}$.

In order to show that $\tilde{G}$ is synchronized rational, we need to find a synchronized rational graph that is isomorphic to $\tilde{G}$. First of all, we need to code the vertices of $\tilde{G}$, each of which consists of a word over Γ and a vector of dimension n, properly. Let $\Pi := \Gamma \cup \{|_1, \ldots, |_n\}$, where $|_1, \ldots, |_n$ are new symbols. These new symbols will be used to code vectors of dimension n. We define the coding function $f\colon \mathbb{N}^n \to |_1^* \cdots |_n^*$ for vectors of dimension n by setting

$$f(x_1, \ldots, x_n) := |_1^{x_1} \cdots |_n^{x_n} ,$$

for each $(x_1, \ldots, x_n) \in \mathbb{N}^n$. Using this coding, we can now define a coding function for the vertices of $\tilde{G}$. Let $h\colon V \times \mathbb{N}^n \to (|_1^* \cdots |_n^*)V$ be defined by

$$h(\alpha, \bar{x}) := f(\bar{x})\alpha ,$$

for each $\alpha \in V$ and $\bar{x} \in \mathbb{N}^n$. Clearly, both f and h are bijective, and thus, the graph $G' := (V', (E'_a)_{a\in\Sigma})$ with

$$V' := (|_1^* \cdots |_n^*)V$$

and

$$E'_a := \{(h(\alpha, \bar{x}), h(\beta, \bar{y})) \in V' \times V' \mid ((\alpha, \bar{x}), (\beta, \bar{y})) \in \tilde{E}_a\}$$

is isomorphic to $\tilde{G}$. The task is now to show that G' is synchronized rational.

Clearly, the set V' of vertices of G' is regular since V is regular. It remains to show that the edge relation E'_a is synchronized rational, for each $a \in \Sigma$.

For each edge relation E'_a, we define a one-way finite automaton $\mathfrak{A}'_a$ with two input tapes and synchronously moving input heads (as described in Section 6.1) as follows. Let two vertices $f(\bar{x})\alpha$ and $f(\bar{y})\beta$ of G' be given.

- First, we guess a vector $\bar{d} \in D$.
- Then, we check whether $\bar{x} + \bar{d} = \bar{y}$. Note that, for this purpose, the automaton must be able to perform a kind of 'shifted reading' since the input heads may only be moved simultaneously on both input tapes. To illustrate this, consider the following example. Let $u := ab$, $v := b$, $\bar{x} := (1, 2)$, and $\bar{y} := (3, 3)$. Then, we have

$$h(u, \bar{x})\hat{}\,h(v, \bar{y}) = \begin{bmatrix} |_1 \\ |_1 \end{bmatrix} \begin{bmatrix} |_2 \\ |_1 \end{bmatrix} \begin{bmatrix} |_2 \\ |_1 \end{bmatrix} \begin{bmatrix} a \\ |_2 \end{bmatrix} \begin{bmatrix} b \\ |_2 \end{bmatrix} \begin{bmatrix} \square \\ |_2 \end{bmatrix} \begin{bmatrix} \square \\ a \end{bmatrix}.$$

  In this example, while we are still working on the first component of the vectors (represented by sequences of $|_1$'s), we must already deal with the second component of the vectors (represented by sequences of $|_2$'s), and similarly, while we are still working with the second component of the vectors, we must already deal with the Γ-components. At first glance, this circumstance might seem to be an obstacle. Anyhow, since D and Γ are finite (and n is fixed), we can tackle this problem by using a finite memory remembering some part of the input tapes. In particular, we must remember the *differences* between $\bar{x}$ and $\bar{y}$ (that is, the vector $\bar{d}$) in each components.
- Finally, after verifying that $\bar{x} + \bar{d} = \bar{y}$, we just simulate the automaton $\mathfrak{A}_{(a,\bar{d})}$, checking whether $(u, v) \in E_{(a,\bar{d})}$.

It is now straightforward to see that $\mathfrak{A}'_a$ indeed recognizes $L_{E'_a}$. Thus, E'_a is synchronized rational, for each $a \in \Sigma$, and consequently, G' (and thus also $\tilde{G}$) is indeed a synchronized rational graph. □

The proof idea of Theorem 6.15 carries over to the case of the monotonic-counter extensions of rational graphs: We can use the same coding techniques as with the monotonic-counter extensions of synchronized rational graphs. The construction of the automata for the edge relations is even easier since the input heads may be moved asynchronously. As with $\mathcal{G}_{\mathrm{SRMC}}$, it follows that the class $\mathcal{G}_{\mathrm{RMC}}$ is the same as the class of rational graphs.

Corollary 6.16. *The monotonic-counter extension of a rational graph is a rational graph.*

Toward separating the class of pushdown graphs from the class $\mathcal{G}_{\mathrm{FMC}}$, we develop the notion of *repetition-free cycles* and show that every graph in the

class $\mathcal{G}_{\mathrm{FMC}}$ only contains repetition-free cycles of *bounded length*. Then, we show that there is a pushdown graph that does not have this property.

Definition 6.17. Let Σ be an alphabet. Further, let $G = (V, (E_a)_{a\in\Sigma})$ be a Σ-labeled graph, and let E be the set of all edges in G. A *repetition-free cycle* in G is a nonempty sequence of vertices $\alpha_1, \ldots, \alpha_m$ in G such that

- the vertices $\alpha_1, \ldots, \alpha_m$ are pairwise different,
- $(\alpha_i, \alpha_{i+1}) \in E$, for $i = 1, \ldots, m-1$, and
- $(\alpha_m, \alpha_1) \in E$.

For such a cycle, the number m is called the *length* of the cycle.

Lemma 6.18. *For each graph in the class $\mathcal{G}_{\mathrm{FMC}}$, there is a natural number $k_0 \geq 0$ such that the length of any repetition-free cycle in this graph does not exceed k_0.*

Proof. Let Σ be an alphabet. Furthermore, let $n \geq 1$ and let D be a finite, nonempty subset of $\mathbb{N}^n$. Let $\tilde{G} = (\tilde{V}, (\tilde{E}_a)_{a\in\Sigma})$ be the monotonic-counter extension of the finite graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$, where V and $E_{(a,\bar{d})}$ are finite, for each $(a, \bar{d}) \in \Sigma \times D$.

Let k_0 be the number of vertices in G, that is, $k_0 := |V|$. We show that the length of any repetition-free cycle in $\tilde{G}$ is bounded by k_0.

Toward a contradiction, we assume that there is a sequence $(\alpha_1, \bar{x}_1)$, $\ldots$, $(\alpha_m, \bar{x}_m)$ of vertices in $\tilde{G}$, for some $m > k_0$, which forms a repetition-free cycle; that is,

- all the vertices $(\alpha_1, \bar{x}_1), \ldots, (\alpha_m, \bar{x}_m)$ are pairwise different,
- $((\alpha_i, \bar{x}_i), (\alpha_{i+1}, \bar{x}_{i+1})) \in \tilde{E}$, for $i = 1, \ldots, m-1$, and
- $((\alpha_m, \bar{x}_m), (\alpha_1, \bar{x}_1)) \in \tilde{E}$.

Since $\tilde{G}$ is the monotonic-counter extension of G, it follows that there are some vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ such that

- $\bar{x}_i + \bar{d}_i = \bar{x}_{i+1}$, for $i = 1, \ldots, m-1$, and
- $\bar{x}_m + \bar{d}_m = \bar{x}_1$.

Now, considering the latter item, we have

$$\bar{x}_1 = \bar{x}_m + \bar{d}_m \,.$$

By replacing $\bar{x}_m$ with $\bar{x}_{m-1} + \bar{d}_{m-1}$, we obtain

$$\bar{x}_1 = \bar{x}_{m-1} + \bar{d}_{m-1} + \bar{d}_m \,.$$

Continuing in this way, we finally obtain

$$\bar{x}_1 = \bar{x}_1 + \bar{d}_1 + \cdots + \bar{d}_m \,.$$

The latter equation implies

$$\bar{d}_1 = \cdots = \bar{d}_m = \bar{0} \,,$$

which, in turn, implies

$$\bar{x}_1 = \cdots = \bar{x}_m \,.$$

Consequently, the statement that all the vertices $(\alpha_1, \bar{x}_1), \ldots, (\alpha_m, \bar{x}_m)$ in $\tilde{G}$ are pairwise different can only be true if all the vertices $\alpha_1, \ldots, \alpha_m$ in G are pairwise different. The latter statement, however, would imply that G contains at least m vertices; that is, $k_0 \geq m$. Contradiction. □

Lemma 6.19. *There is a pushdown graph that contains repetition-free cycles of unbounded length.*

Proof. Let G be the pushdown graph of Example 6.2. We will show that for each natural number k_0 there is a repetition-free cycle in G whose length exceeds k_0.

Actually, this fact can directly be seen from Figure 6.1. To illustrate this, consider those paths in G that start from the vertex $q_0\bot$ and whose edges are labeled with the words *abc*, *aabcc*, *aaabccc*, and so on. It is easy to see that each of these paths forms a repetition-free cycle. Moreover, the length of these cycles is unbounded; for any given $k_0 \geq 0$, the path starting from $q_0\bot$ whose edges are labeled with the word $a^{k_0+1}bc^{k_0+1}$ is a repetition-free cycle of length $2k_0 + 3 > k_0$.

Formally, suppose that $k_0 \geq 0$ is given. Then, it is straightforward to verify that the sequence of vertices

$$q_0\bot, q_0Z\bot, \ldots, q_0Z^{k_0+1}\bot, q_1Z^{k_0}, \ldots, q_1Z\bot, q_1\bot$$

is a repetition-free cycle of length $2k_0 + 3 > k_0$ in G. □

We are now ready to prove the main result of this section. We extend the hierarchy of the graph classes mentioned in Section 6.1.

Theorem 6.20. *(a) The classes $\mathcal{G}_{\mathrm{FMC}}$ and $\mathcal{G}_{\mathrm{PD}}$ are not comparable to each other.*

(b) The classes $\mathcal{G}_{\mathrm{PDMC}}$ and $\mathcal{G}_{\mathrm{PR}}$ are not comparable to each other.

(c) The classes $\mathcal{G}_{\mathrm{FMC}}$ and $\mathcal{G}_{\mathrm{PR}}$ are not comparable to each other.

(d) $\mathcal{G}_{\mathrm{PD}} \subsetneq \mathcal{G}_{\mathrm{PDMC}}$.

(e) $\mathcal{G}_{\mathrm{PR}} \subsetneq \mathcal{G}_{\mathrm{PRMC}}$.
(f) $\mathcal{G}_{\mathrm{Fin}} \subsetneq \mathcal{G}_{\mathrm{FMC}} \subsetneq \mathcal{G}_{\mathrm{PDMC}} \subsetneq \mathcal{G}_{\mathrm{PRMC}} \subsetneq \mathcal{G}_{\mathrm{SR}}$.

Proof. Part (a). By Proposition 6.12, the class $\mathcal{G}_{\mathrm{FMC}}$ contains a graph whose monadic second-order theory is undecidable. In contrast, the class $\mathcal{G}_{\mathrm{PD}}$ contains only graphs whose monadic second-order theory is decidable (see Table 6.1). Thus, the class $\mathcal{G}_{\mathrm{FMC}}$ is not contained in the class $\mathcal{G}_{\mathrm{PD}}$.

Conversely, by Lemma 6.18 and Lemma 6.19, there is a pushdown graph that does not belong to the class $\mathcal{G}_{\mathrm{FMC}}$. Hence, the class $\mathcal{G}_{\mathrm{PD}}$ is not contained in the class $\mathcal{G}_{\mathrm{FMC}}$.

Part (b). As with the class $\mathcal{G}_{\mathrm{FMC}}$, the class $\mathcal{G}_{\mathrm{PDMC}}$ contains a graph whose monadic second-order theory is undecidable. Thus, the latter class is not contained in the class $\mathcal{G}_{\mathrm{PR}}$, which contains only graphs whose monadic second-order theory is decidable (see Table 6.1).

For the converse, we observe that every graph in the class $\mathcal{G}_{\mathrm{PDMC}}$, by definition, contains only vertices of finite degree (the number of ingoing and outgoing edges); the sets Σ and D in Definition 6.6 are finite, and every pushdown automaton has only finitely many transitions. In contrast, the prefix-recognizable graph of Example 6.3 contains vertices of infinite outdegree, and thus, it does not belong to the class $\mathcal{G}_{\mathrm{PDMC}}$. We conclude that the class $\mathcal{G}_{\mathrm{PR}}$ is not contained in the class $\mathcal{G}_{\mathrm{PDMC}}$.

Part (c). As noted above, the class $\mathcal{G}_{\mathrm{FMC}}$ contains a graph with an undecidable monadic second-order theory, and therefore, this class is not contained in the class $\mathcal{G}_{\mathrm{PR}}$.

Moreover, the class $\mathcal{G}_{\mathrm{PR}}$ is not contained in the class $\mathcal{G}_{\mathrm{FMC}}$; otherwise, the class $\mathcal{G}_{\mathrm{PR}}$ would be contained in the class $\mathcal{G}_{\mathrm{PDMC}}$ since, by Remark 6.7(c), the class $\mathcal{G}_{\mathrm{FMC}}$ is contained in the class $\mathcal{G}_{\mathrm{PDMC}}$. Contradiction.

Part (d) and (e). The inclusions follow from Remark 6.7(b). The strictness of the inclusions follow from Proposition 6.12 and Table 6.1: The classes $\mathcal{G}_{\mathrm{PDMC}}$ and $\mathcal{G}_{\mathrm{PRMC}}$ contain a graph whose monadic second-order theory is undecidable whereas the classes $\mathcal{G}_{\mathrm{PD}}$ and $\mathcal{G}_{\mathrm{PR}}$ contain only graphs whose monadic second-order theory is decidable.

Part (f). All the inclusions follow from Remark 6.7(b) and Remark 6.7(c).

The strictness of the inclusion $\mathcal{G}_{\mathrm{Fin}} \subsetneq \mathcal{G}_{\mathrm{FMC}}$ is obvious; the class $\mathcal{G}_{\mathrm{FMC}}$, of course, also contains infinite graphs.

The strictness of the inclusions $\mathcal{G}_{\mathrm{FMC}} \subsetneq \mathcal{G}_{\mathrm{PDMC}}$ and $\mathcal{G}_{\mathrm{PDMC}} \subsetneq \mathcal{G}_{\mathrm{PRMC}}$ follow from part (a) and (d) above, and from part (b) and (e) above, respectively.

The strictness of the inclusion $\mathcal{G}_{\mathrm{PRMC}} \subsetneq \mathcal{G}_{\mathrm{SR}}$ follows from the fact that the decision properties of these classes differ from each other, with respect to the reachability problem (see Table 6.1 and Theorem 6.11). □

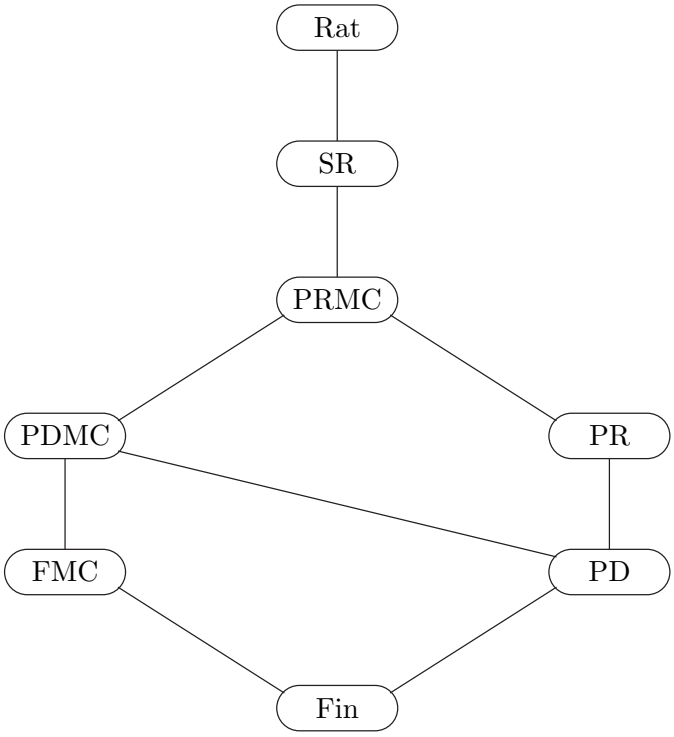


Figure 6.5: Hierarchy of graph classes

The results of this section are summarized in Figure 6.5, where each upward, solid edge indicates a strict inclusion. For simplicity, we write, for instance, FMC instead of $\mathcal{G}_{\mathrm{FMC}}$.

Summary and Concluding Remarks

In this chapter, we explored the application of the idea of Parikh automata to infinite graphs.

In Section 6.2 we introduced the notion of monotonic-counter graphs, which were obtained by extending the vertices of an underlying graph with vectors of natural numbers. Using this extension, we obtained new classes of infinite graphs from other existing classes of graphs.

Then, in Section 6.3 we investigated some decision problems concerning monotonic-counter graphs. Our main result was the decidability of the reachability problem for the monotonic-counter extensions of prefix-recognizable graphs, which was proven by means of a reduction to the emptiness problem for semi-linear sets.

In Section 6.4 we compared these new classes with some well-known classes of infinite graphs. We extended the hierarchy of the well-known graph classes to the classes of monotonic-counter graphs.

One aspect that remains unexplored is concerned with the traces of monotonic-counter graphs. Of course, the fact that our starting point was the idea of Parikh automata may suggest that the traces of monotonic-counter graphs yield languages that are recognized by Parikh automata. Still, this aspect deserves further study.

Bibliography

[Aho68] Alfred V. Aho. Indexed grammars – an extension of context-free grammars. *Journal of the Association for Computing Machinery*, 15(4):647–671, 1968. [63]

[BDR03] Véronique Bruyére, Emmanuel Dall'Olio, and Jean-François Raskin. Durations, parametric model-checking in timed automata with Presburger arithmetic. In *Proceedings of STACS 2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 687–698. Springer, 2003. [2]

[BH96] Ahmed Bouajjani and Peter Habermehl. Constrained properties, semi-linear systems, and Petri nets. In *Proceedings of CONCUR 1996*, volume 1119 of *Lecture Notes in Computer Science*, pages 481–497. Springer, 1996. [2]

[Big89] Norman L. Biggs. *Discrete Mathematics*. Oxford University Press, revised edition, 1989. [33, 69, 87, 90]

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960. [11]

[Cau02] Didier Caucal. On infinite terms having a decidable monadic theory. In *Proceedings of MFCS 2002*, volume 2420 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2002. [63]

[CJ98] Hubert Comon and Yan Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *Proceedings of CAV 1998*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998. [2]

[CW03] Arnaud Carayol and Stefan Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata.

In *Proceedings of FSTTCS 2003*, volume 2914 of *Lecture Notes in Computer Science*, pages 112–123. Springer, 2003. [63]

[DG82] Werner Damm and Andreas Goerdt. An automata-theoretic characterization of the OI-hierarchy. In *Proceedings of ICALP 1982*, volume 140 of *Lecture Notes in Computer Science*, pages 141–153. Springer, 1982. [63]

[DIB+00] Zhe Dang, Oscar H. Ibarra, Tevfik Bultan, Richard A. Kemmerer, and Jianwen Su. Binary reachability analysis of discrete pushdown timed automata. In *Proceedings of CAV 2000*, volume 1855 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2000. [2]

[DL03] Silvano Dal Zilio and Denis Lugiez. XML schema, tree logic and sheaves automata. In *Proceedings of RTA 2003*, volume 2706 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2003. [2]

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer, second edition, 1994. [9]

[Eng83] Joost Engelfriet. Iterated pushdown automata and complexity classes. In *Proceedings of STOC 1983*, pages 365–373. ACM Press, 1983. [63]

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, second edition, 1994. [83]

[GS64] Seymour Ginsburg and Edwin H. Spanier. Bounded ALGOL-like languages. *Transactions of the American Mathematical Society*, 113:333–368, 1964. [1, 9]

[GS66] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966. [1, 11]

[Har78] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978. [5, 54]

[HIKS02] Tero Harju, Oscar Ibarra, Juhani Karhumäki, and Arto Salomaa. Some decision problems concerning semilinearity and commutation. *Journal of Computer and System Sciences*, 65:278–294, 2002. [13]

[HU79] John E. Hocpcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979. [5, 42, 49, 53, 54, 57, 58]

[Iba78] Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the Association for Computing Machinery*, 25(1):116–133, 1978. [1, 44, 59]

[IBS00] Oscar H. Ibarra, Tevfik Bultan, and Jianwen Su. Reachability analysis for some models of infinite-state transition systems. In *Proceedings of CONCUR 2000*, volume 1877 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2000. [2]

[Kla04] Felix Klaedtke. *Automata-Based Decision Procedures for Weak Arithmetics.* PhD thesis, Albert-Ludwigs-Universität Freiburg, Fakultät für Angewandte Wissenschaften, Institut für Informatik, Freiburg im Breisgau, 2004. [11, 31]

[KNU02] Teodor Knapik, Damian Niwiński, and Pawel Urzyczyn. Higher-order pushdown trees are easy. In *Proceedings of FOSSACS 2002*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002. [63]

[KR02] Felix Klaedtke and Harald Rueß. Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science at Freiburg University, 2002. [2, 15, 16, 20, 31, 43, 44, 45, 53, 59]

[KR03] Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *Proceedings of ICALP 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 681–696. Springer, 2003. [2, 15, 16, 43]

[Leu02] Martin Leucker. Prefix-recognizable graphs and monadic logic. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, chapter 15, pages 263–283. Springer, 2002. [107]

[LK03] L. P. Lisovik and T. A. Karnaukh. A class of functions computable by index grammars. *Cybernetics and Systems Analysis*, 39(1):91–96, 2003. [81]

[Mas76] A.N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 12(1):38–42, 1976. [63]

[Mat93] Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993. [94]

[Par66] Rohit J. Parikh. On context-free languages. *Journal of the Association for Computing Machinery*, 13(4):570–581, 1966. [1, 7, 12]

[Pre30] Mojzesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Sprawozdanie z I Kongresu matematyków krajów slowiańskich (Comptes Rendus du I^{er} Congrès des Mathématiciens des Pays Slaves), Warszawa 1929*, pages 92–101 and 395, 1930. [10]

[Sal73] Arto Salomaa. *Formal Languages*. Academic Press, 1973. [54]

[SSM03] Helmut Seidl, Thomas Schwentick, and Anca Muscholl. Numerical document queries. In *Proceedings of PODS 2003*, pages 155–166. ACM Press, 2003. [2]

[Sti00] Colin Stirling. Decidability of bisimulation equivalence for pushdown processes. Technical Report EDI-INF-RR-0005, School of Informatics, University of Edinburgh, Scottland, 2000. [108]

[Tho02] Wolfgang Thomas. A short introduction to infinite automata. In *Proceedings of DLT 2001*, volume 2295 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2002. [97, 98, 103, 104, 107]

[Wey02] Mark Weyer. Decidability of S1S and S2S. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, chapter 12, pages 207–230. Springer, 2002. [11]