Université de Montréal

Automates à contraintes semilinéaires

Automata with a semilinear constraint

par Michaël Cadilhac

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.) en informatique

Novembre 2012

Résumé

Cette thèse présente une étude dans divers domaines de l'informatique théorique de modèles de calculs combinant automates finis et contraintes arithmétiques. Nous nous intéressons aux questions de décidabilité, d'expressivité et de clôture, tout en ouvrant l'étude à la complexité, la logique, l'algèbre et aux applications. Cette étude est présentée au travers de quatre articles de recherche.

Le premier article, *Affine Parikh Automata*, poursuit l'étude de Klaedtke and Rueß des automates de Parikh et en définit des généralisations et restrictions. L'automate de Parikh est un point de départ de cette thèse; nous montrons que ce modèle de calcul est équivalent à *l'automate contraint* que nous définissons comme un automate qui n'accepte un mot que si le nombre de fois que chaque transition est empruntée répond à une contrainte arithmétique. Ce modèle est naturellement étendu à *l'automate de Parikh affine* qui effectue une opération affine sur un ensemble de registres lors du franchissement d'une transition. Nous étudions aussi *l'automate de Parikh sur lettres*: un automate qui n'accepte un mot que si le nombre de fois que chaque lettre y apparaît répond à une contrainte arithmétique.

Le deuxième article, *Bounded Parikh Automata*, étudie les langages *bornés* des automates de Parikh. Un langage est borné s'il existe des mots $w_1, w_2, \ldots, w_k$ tels que chaque mot du langage peut s'écrire $w_1 \cdots w_1 w_2 \cdots w_2 \cdots w_k \cdots w_k$. Ces langages sont importants dans des domaines applicatifs et présentent usuellement de bonnes propriétés théoriques. Nous montrons que dans le contexte des langages bornés, le déterminisme n'influence pas l'expressivité des automates de Parikh.

Le troisième article, *Unambiguous Constrained Automata*, introduit les automates contraints *non ambigus*, c'est-à-dire pour lesquels il n'existe qu'un chemin acceptant par mot reconnu par l'automate. Nous montrons qu'il s'agit d'un modèle combinant une meilleure expressivité et de meilleures propriétés de clôture que l'automate contraint déterministe. Le problème de déterminer si le langage d'un automate contraint non ambigu est régulier est montré décidable.

Le quatrième article, *Algebra and Complexity Meet Contrained Automata*, présente une étude des représentations algébriques qu'admettent les automates contraints et les automates de Parikh affines. Nous déduisons de ces caractérisations des résultats d'expressivité et de complexité. Nous montrons aussi que certaines hypothèses classiques en complexité computationelle sont reliées à des résultats de séparation et de non clôture dans les automates de Parikh affines.

La thèse est conclue par une ouverture à un possible approfondissement, au travers d'un certain nombre de problèmes ouverts.

Mots-clés : automates, automates de Parikh, ensembles semilinéaires, fonctions affines, decidabilité, langages bornés, langages réguliers, monoïdes typés, séries rationnelles, *model-checking*.

Abstract

This thesis presents a study from the theoretical computer science perspective of computing models combining finite automata and arithmetic constraints. We focus on decidability questions, expressiveness, and closure properties, while opening the study to complexity, logic, algebra, and applications. This thesis is presented through four research articles.

The first article, *Affine Parikh Automata*, continues the study of Klaedtke and Rueß on Parikh automata and defines generalizations and restrictions of this model. The Parikh automaton is one of the starting points of this thesis. We show that this model of computation is equivalent to the *constrained automaton* that we define as an automaton which accepts a word only if the number of times each transition is taken satisfies a given arithmetic constraint. This model is naturally extended to *affine Parikh automata*, in which an affine transformation is applied to a set of registers on taking a transition. We also study the *Parikh automaton on letters*, that is, an automaton which accepts a word only if the number of times each letter appears in the word verifies an arithmetic constraint.

The second article, *Bounded Parikh Automata*, focuses on the *bounded* languages of Parikh automata. A language is bounded if there are words $w_1, w_2, \dots, w_k$ such that every word in the language can be written as $w_1 \cdots w_1 w_2 \cdots w_2 \cdots w_k \cdots w_k$. These languages are important in applications and usually display good theoretical properties. We show that, over the bounded languages, determinism does not influence the expressiveness of Parikh automata.

The third article, *Unambiguous Constrained Automata*, introduces the concept of *unambiguity* in constrained automata. An automaton is unambiguous if there is only one accepting path per word of its language. We show that the unambiguous constrained automaton is an appealing model of computation which combines a better expressiveness and better closure properties than the deterministic constrained automaton. We show that it is decidable whether the language of an unambiguous constrained automaton is regular.

The fourth article, *Algebra and Complexity Meet Constrained Automata*, presents a study of algebraic representations of constrained automata and affine Parikh automata. We deduce expressiveness and complexity results from these characterizations. We also study how classical computational complexity hypotheses help in showing separations and nonclosure properties in affine Parikh automata.

The thesis is concluded by a presentation of possible future avenues of research, through several open problems.

Keywords: automata, Parikh automata, semilinear sets, affine functions, decidability, bounded languages, regular languages, typed monoids, rational series, model-checking.

Contents

List of Acronyms

AFL	Abstract Family of Languages.
APA	Affine Parikh Automaton.
BC	Boolean Closure.
BSL	Bounded Semilinear Languages.
CA	Constrained Automaton.
CFL	Context-Free Languages.
CQDD	Constrained Queue-content Decision Diagram.
CSL	Context-Sensitive Languages.
DetAPA	Deterministic Affine Parikh Automaton.
DetCA	Deterministic Constrained Automaton.
DetLPA	Deterministic Letter Parikh Automaton.
DetPA	Deterministic Parikh Automaton.
FM-(Det)APA	Finite-Monoid (Deterministic) Affine Parikh Automaton.
FO	First-Order logic.
LPA	Letter Parikh Automaton.

M-APA	Moving Affine Parikh Automaton.
MSO	Monadic Second-Order logic.
PA	Parikh Automaton.
RBCM	Reversal-Bounded multi-Counter Machine.
SLRE	Semilinear Regular Expression.
UnAPA	Unambiguous Affine Parikh Automaton.
UnCA	Unambiguous Constrained Automaton.

To my father,
who thought I'd be a damn good engineer.

aussitôt demain vidé de ses astres
tu rassembles tes affaires
pour les enfouir en secret dans la vase

n'ébruite pas la chose
tu ne saurais répondre
de la disparition des heures

— Olivier Labonté
in *Lointain écho de la petite histoire*

Acknowledgments

My foremost thanks go to Pierre McKenzie who gave me the opportunity to work with him. Doing so, he certainly took a chance and I am very grateful of his risk taking. Working with him was enriching not only in a scientific way, but also humanly speaking.

I am also deeply indebted to Alain Finkel, without whose constant legitimate and strong insistence on the finest details, this thesis would never have reached this degree of precision, formality, and clarity.

I want to thank Andreas Krebs, coauthor of Paper IV, for his irrational willingness to work with me, despite our differences in brain speed — he is the fast one of course. His work methods and sum of knowledge are an inspiration.

I am grateful to David A. Mix Barrington for agreeing to review this thesis. It is in his honor that the formatting of the main text is of bounded-width (5 inches, which is sufficient).

My first encounter with automata theory is still a vivid memory and the causes of this vividness certainly contributed to my working in this field. Hence I am grateful to my first teachers on the theory, Akim Demaille and Jacques Sakarovitch.

The *laboratoire d'informatique théorique et quantique* at the Université de Montréal abounds with brilliant and kind people, some of them only pass by, others are there to stay. I thank them all for the nice atmosphere to which they all contribute. In particular, the following people were kind enough to work with me at some point: Laurent Beaudou, Michael Blondin, Yara Elias, Marc Kaplan, Philippe Lamontagne, Adrien Lemaître, David Pouliot, Samuel Ranellucci, Benno Salwey, Sébastien Tavenas, and Dave Touchette.

I moreover acknowledge the help of the following people who proofread parts of this document (I still keep full ownership of the remaining typos): Paul Khuong, Mathieu Lemoine, and Nicolas Widynski.

Finally, I want to thank all my friends for their continuing support in more ways than one. Financial support through Pierre McKenzie's Natural Sciences and Engineering Research Council of Canada Discovery Grant is also gratefully acknowledged.

This PhD was a great experience, and I am pretty confident in saying that if I had to do it all over again, I would buy a better chair.

A note on formatting

The formatting style of a thesis is often a touchy subject in the academics. I acknowledge the fact that the Université de Montréal does not restrict as much as some other universities the writing style and presentation. Also, I thank the editorial board of the Leibniz International Proceedings in Informatics for taking the bold and rare move of having a very appealing LaTeX style for their publications. I blatantly mimicked parts of the style they use.

Introduction

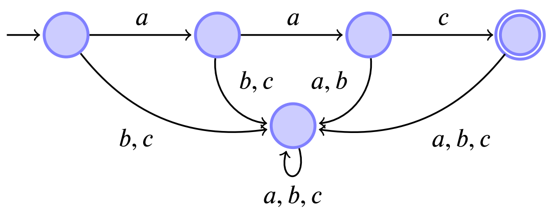
Scientists and engineers alike have tried for decades to automate the tasks humans do. Neuropsychologists on the one hand tried to model the human brain, so that a formal device could learn to think (and hence compute) like a person. On the other hand, in a complementary way, logicians and the first computer scientists worked on building machines that would compute specific *mental tasks* (additions, proving theorems, finding a shortest path). In this introduction, we present how to formalize the notion of *implementation* of a task, and how, given an implementation, one can verify that it fits a *specification*.

This introduction is partly inspired by the historic surveys of Greibach [Gre81] and Perrin [Per95, Per03].

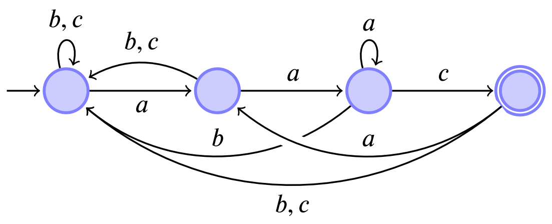
1 Models

The common root of the works on automation is the notion of *finite control structure*, which gave rise to the formalization of the concept of *algorithm*. A finite control structure is a finite set of *states* together with a finite set of *rules* describing how the device goes from one state to another. Indeed, when mimicking human computation, the foremost requirement is that the "recipe" for that computation be finite: one could imagine having an infinite supply of a particular ingredient (say, salt and pepper), but not having to follow an infinitely long recipe. This distinction between the control structure (the recipe) and the resources available (the ingredients) is at the core of the hierarchies that abound in the study of computer science.

At one end of the spectrum, we find the finite control structure *without* any additional resource: the finite automaton. This applies to the simplest tasks of mechanization, and we illustrate this with an example. Suppose we want to automate the ingrate task of waiting at a door to let only the people with the *password* in. We want to design a device which receives an input consisting of a finite sequence of letters, accepts if the input matches a prescribed password, and rejects otherwise. With the three letter alphabet $\Sigma = \{a, b, c\}$ and the password aac, this is described by the following diagram:



In this diagram, states are represented by circles and rules (or *transitions*) by edges. Each time a piece of input (a letter) is given to the automaton, it goes to the state prescribed by the *current state* and the input. Now the automaton *accepts* starting from the *initial state* (that is, finishes in the double-circled state starting from the one with an arrow) if the input is aac, and nothing more. This device has the major drawback that it has to be restarted each time someone makes a mistake; for instance, the word $aaac$, which ends with the password, is rejected. We may avoid this drawback using a slightly more complicated version of the previous automaton:



This is such that each time the input ends with aac, the automaton is in the accepting (double-circled) state.

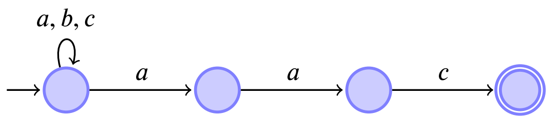
We see that these machines, albeit simple in appearance, already offer a certain challenge in implementing certain tasks. At the other end of the spectrum is the *Turing machine*, this is a finite control structure (as always) with an unbounded memory. The machine is presented with the input at the beginning of the memory tape, and can, in addition to the abilities of the finite automaton, move the read head on the tape and write on it. It may look dubious, at first, that such a simple device is "at the other end of the spectrum." This is known as the *Church-Turing thesis*: any mechanical process can be simulated using a Turing machine. Although this is not a formal statement (and it cannot be one), it has been supported by the last seventy years of research: no device which works by a discrete sequence of "small steps" has been found to exceed the power of the Turing machine. Now, it should not come as a surprise that finite automata are way less *expressive* than Turing machines; but what is the formal statement for such a property?

In the usual vocabulary, an *alphabet* Σ is a finite set of input symbols called *letters*. Naturally, a finite sequence of letters is a *word*, and we write Σ^* for the set of all words. Now a set of words (finite or not), that is, a subset of Σ^*, is a *language*. When studying the *expressive power* of a computing device, we are usually interested in the *languages* it decides, i.e., what are the languages for which there is a device that accepts if the input is in the language and rejects otherwise. This may look like a restriction on two accounts. As a first restriction, we do not consider *functions*, although most processes call for *finding* a solution. There are numerous ways to address this problem; say we want to compute a function from Σ^* to the integers, then we can consider languages L whose words are of the form (w, i) for $w \in \Sigma^*$ and i an integer. Here (w, i) is short for the word consisting of w followed by the binary representation of i. Thence we can see L as mapping $w \in \Sigma^*$ to an integer n whose i-th bit is 1 if $(w, i) \in L$ and 0 otherwise. As a second restriction, we are studying processes that eventually reach a stop — as opposed, for instance, to a critical system that should run indefinitely. In this case, the words under study should be *infinite* sequences of input letters. This fruitful field of computer science, which focuses on so-called *ω-languages*, i.e., languages of infinite words, is in fact a different branch that we do not consider here.

Using these terms, we say that the language of the first automaton depicted above is $\{aac\}$, while the language of the second is that of words ending with aac. Also, we may give a formal statement for the property that finite automata are less expressive than Turing machines: there are languages which are decided by Turing machines but by no finite automaton. Since we have asserted that the Turing machine is the most powerful machine, it may come as a surprise that there are languages which are not decided by any Turing machine. This is essentially the results of the works of Gödel, Turing, and Church in the 1930s on the *Entscheidungsproblem*: there is no mechanical way to determine, given a mathematical statement about the integers, whether it is true, thus the language of such true statements is not decided by a Turing machine.

Between finite automata and Turing machines lie two important hierarchies. The first hierarchy, devised on the impulse of the linguists, is due to Chomsky [Cho56]. It consists of 4 levels: the *recursively enumerable* languages are those recognized by Turing machines; the *context-sensitive* languages are those recognized by a Turing machine using a memory size *linear* in the length of its input; the *context-free* languages are those recognized by finite automata equipped with a stack; finally, the bottom of the hierarchy is formed by the *regular* languages, those recognized by finite automata. The second hierarchy is that of computational complexity, and is far more diverse. It consists in restraining a Turing machine to work with an amount of time and/or memory size which depends, in a given fashion, on the length of its input.

In both hierarchies, one critical property that is examined is the contribution of *nondeterminism*. We first present an example. The password example at the beginning of this section could have been more succinctly solved using the following automaton:



Clearly, for any word ending in *aac*, there is a path in this automaton going from the initial state and ending in the accepting state. However, the word *aac* is also read by a path which does not end in the accepting state. *Nondeterminism* is the property that a word is accepted if *at least one* path leads to acceptance. It is usually more succinct to present a regular language using a nondeterministic automaton, but checking manually that a word is accepted becomes more complicated. In the realm of finite automata, nondeterminism does not add expressive power, that is, the languages recognized by nondeterministic automata are those recognized by deterministic automata. It is compelling to study determinism in particular because a central problem of complexity theory is whether nondeterminism adds power to Turing machines restrained to work in a time polynomial in the length of their input (known as the P vs. NP problem [Coo71]).

2 The behavior of a model

As the building blocks of the specification of a task are put together, it becomes increasingly difficult to be convinced that their implementation is correct. Suppose for instance that our password checker is part of a vast application, buried deep and interleaved with different functions of the bigger software. This software is still a finite control structure (represented by its source code) with given resources. Now we want to check that the software *behaves* correctly: specifying a behavior and implementing it are two distinct tasks, and we may be confident that the specification of the behavior is correct. Thus what we want to do is run the software and check that each of its actions comply with the specification; an impossible task in the usual case. However, when checking critical parts of the system, as a password checker, we are not interested in the behavior of the whole software, but only that properties of the following form are satisfied:

- *The password is accepted only if it is correct,*
- *The correct password is always accepted.*

Checking *properties* of a system, rather than compliance with regards to a comprehensive specification, is the concern of *model-checking* [BBF+01, BK08]. Model-checking is a two-phase process. First, a model of the software has to be constructed; this is usually done using a finite automaton A with added synchronizing features. Second, the property ϕ to be verified has to be written in a formal language, usually a *temporal logic*, which are apt to specify assertions such as "If in the future we are in the state where no correct character of the password has been entered, then the next state will not be the password accepting state." Then model-checking offers, according to the choice of modeling framework and logic, ways to check that the automaton A verifies the property ϕ. In particular, if ϕ is specified using another automaton B, then checking that A verifies ϕ is equivalent to checking whether the language of A has an empty intersection with the complement of the language of B. This has to be an efficient procedure for model-checking to be tractable.

When presented with a (deterministic) automaton, computing an automaton which recognizes the complement language is simply a matter of turning the accepting states into nonaccepting states and *vice versa*. Given two automata, we can compute an automaton recognizing the intersection of the two languages quite easily: it is the automaton which moves in the two automata at the same time, that is, it has states of the form (p, q) where p is a state of the first automaton and q a state of the second. Next, checking that an automaton has an empty language amounts to checking if there is a path between the initial state and an accepting state. All of the operations being quite efficient, we can state that model-checking where both the model and the property are specified as automata can be done efficiently.

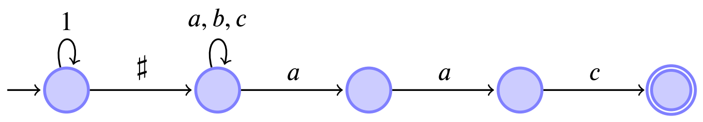
Now say we want to express the model with a more powerful device than the finite automaton, allowing access to a stack to be used as a temporary storage mechanism. One definite advantage of doing this is that extra power usually comes with a gain in the ease of modeling. Then, before asking whether we can model-check *efficiently* using this more powerful device, a fundamental question is whether we *can* model-check using it at all; that is, is there an algorithm (a Turing machine) which, given an automaton A with a stack and an automaton B, decides whether the language of A has an empty intersection with the complement of the language of B. In fact, it is known that such an intersection is again expressible as the language of an automaton with a stack, hence the fundamental question amounts to whether we can determine if an automaton with a stack has an empty language.

Parikh [Par66] gave a positive answer to this question for automata with stacks using the notion of *commutative image* of a word. The name stems from the fact that a word is seen as an element of a noncommutative structure: the words *ab* and *ba* are different. The commutative image of a word is the information that a word contains when we consider that the letters commute; this is the number of times each letter appears in the word. Thus define $\mathsf{Pkh}(w)$, for a word $w \in \Sigma^*$, to be the vector of size $|\Sigma|$ where the i-th component indicates the number of times the i-th letter of Σ appears in w. The notation $\mathsf{Pkh}(w)$ comes from the fact that the commutative image of a word is nowadays called the *Parikh image* of w. Now Parikh's theorem gives a characterization of the set of Parikh images of languages of automata with a stack which is as follows. We say that a set E of vectors of dimension d is *linear* if $E = \{\mathbf{v}_0 + k_1\mathbf{v_1} + \cdots + k_n\mathbf{v_n} \mid k_i \in \mathbb{N}\}$, with $\mathbf{v}_i$ vectors of dimension d. It is *semilinear* if it is a finite union of linear sets. Parikh's theorem states that if L is the language of an automaton with a stack, the set $\mathsf{Pkh}(L)$ of Parikh images of the words in L is *effectively* semilinear. The term "effectively" refers to the fact that we can find an explicit expression for $\mathsf{Pkh}(L)$. Now L is empty iff $\mathsf{Pkh}(L)$ is empty, and checking whether a (semi)linear set is empty is easy, thus one may check whether the language of an automaton with a stack is empty. Hence model-checking where the model is specified using an automaton with a stack and the property is specified using an automaton is doable — although efficiency here is a greater problem.

More generally, when a computing device is presented, it is of great interest to study three main properties: its expressiveness (what are the languages it can decide?), its closure properties (e.g., if two languages are decided by the device, is it also true for their intersection?), and decidability properties (e.g., can we decide whether the language of the device is empty? or expressible using a simpler model?). In particular, depending on the efficiency of the operations and on the fitness of the device to represent real-world implementations, one can then consider relying on it for model-checking.

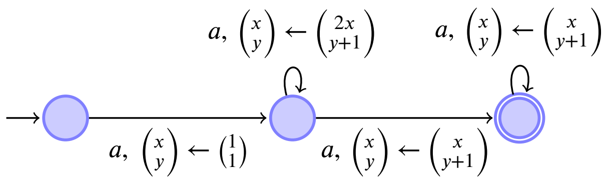
3 Contribution of this thesis

We continue the study of *Parikh automata*, introduced by Klaedtke and Rueß [KR03] in the context of model-checking. Let us first give the definition of an equivalent, arguably simpler model. A *constrained automaton* is a pair (A, C) where A is a finite automaton and C is a semilinear set. Its language is the set of inputs accepted by paths π in A for which $\mathsf{Pkh}(\pi) \in C$ — here, we see the transition set as an alphabet and the set of accepted paths as a language. Such a device is especially interesting when there is a need to specify a simple arithmetic constraint on the number of times a transition is taken; it is indeed an essential characterization of semilinear sets that they are the sets describable by a mathematical statement about the integers which only uses addition. We saw earlier that without this latter restriction, there is no Turing machine which decides the truth of a mathematical statement; with the restriction of using only addition, such a Turing machine exists, although its running time makes it usable only on small or simple formulas. We give an example of a constrained automaton. The following (nondeterministic) constrained automaton accepts the words which are a sequence of n letters 1, then a $\sharp$ followed by a word ending in *aac* but *no longer* than n. The automaton A is given by:



The set C is then specified so as to constrain the transition labeled 1 to occur more often than all the other transitions put together. Specifically, if x_1 is the number of occurrences of the transition labeled 1 and $x_2, x_3, \dots, x_8$ are the numbers of occurrences of the seven other transitions respectively, then C is specified by the inequality $x_1 \geq x_2 + x_3 + \cdots + x_8$, which describes a semilinear set of 8-tuples of integers.

One of the central definitions contributed by this thesis, and the main focus of the first paper to follow, is the *affine Parikh automaton*. An affine Parikh automaton is an automaton which uses a vector of integers as an additional resource. Upon taking a transition, the automaton applies an affine transformation to this vector. At the end of the computation, the input is accepted if the automaton accepts it and the vector belongs to a prescribed semilinear set. As an example, the following (nondeterministic) affine Parikh automaton of dimension 2 recognizes the language of words of lengths which are positive powers of two over the single letter a. The automaton and the affine transformations are given by:


$a, \binom{x}{y} \leftarrow \binom{2x}{y+1}$
$a, \binom{x}{y} \leftarrow \binom{x}{y+1}$
$a, \binom{x}{y} \leftarrow \binom{1}{1}$
$a, \binom{x}{y} \leftarrow \binom{x}{y+1}$

The constraint set then asserts that $x = y$. As for a word w, x can be any power of 2 less than $2^{|w|-1}$ and $y = |w|$, the language of this affine Parikh automaton is as claimed.

*
**

This thesis consists of an introductory technical chapter (Chapter 0) followed by four papers, each introduced and discussed in separate sections.

Chapter 0. This chapter contains the technical preliminaries common to all the papers. It also mentions facts from the literature which are not needed in this thesis but offer a broader picture.

Paper I: Affine Parikh Automata. We present new results on the expressive power of Parikh automata (which are proved equivalent to constrained automata) and introduce and study the affine Parikh automaton.

Paper II: Bounded Parikh Automata. We show, using in particular affine Parikh automata, that nondeterminism does not add expressiveness to Parikh automata over the bounded languages, i.e., the languages for which there exist words $w_1, w_2, \dots, w_k$ such that any word of the language can be written as $w_1 \cdots w_1 w_2 \cdots w_2 \cdots w_k \cdots w_k$.

Paper III: Unambiguous Constrained Automata. We study the constrained automata for which the underlying automaton is *unambiguous*, that is, has at most one accepting path per label. We show that this is an appealing model and that we can decide given an unambiguous constrained automaton whether its language is regular.

Paper IV: Algebra and Complexity Meet Constrained Automata. We give algebraic characterizations of constrained automata and affine Parikh automata, from which we derive expressiveness and complexity results. We use complexity assumptions to show that determinism impacts the expressive power of affine Parikh automata.

A note on the papers

The first three papers are the journal versions of conference papers that appeared in refereed conference proceedings. All three are invited submissions to the journal special issue of the related conference; Papers I and II are in print, while Paper III is still in preparation. Paper IV is in preparation and is not yet submitted to any venue.

The published work is integrally reproduced here. However, we merged the common preliminaries of the papers into Chapter 0; technical definitions which are local to only one paper are included in the specific paper. The margin notes included in the papers are added to clarify or reflect on particular points, and only appear in the thesis. We also homogenized the notation throughout the papers.

CHAPTER **0**

Definitions and Notations

Il n'y a que les mots qui comptent,
le reste n'est que bavardage
— Eugène Ionesco

In an effort to homogenize notation throughout the forthcoming articles and avoid repetition, we give in this section the essential notions of the present works. We also present the articles which form the basis of this research, and some related topics in order to put this thesis in its context.

The actual content of this section is larger than the union of the similar sections of the articles; we included more basic definitions that may help the reader not versed in language theory. It is however expected that basics of language theory — languages, regular languages, closure properties, automata — are known, and the goal of the presentation made here is essentially to fix notations.

Most readers may safely skim through this chapter, but should pay attention to the notions of semilinear sets, Parikh automata, and constrained automata. The sections are generally exempt from references, these being condensed in multiple "Bibliographical notes and comments." Those notes also contain related material which extends that of the main text but will not be needed in the articles.

1 Vectors, functions, definable and semilinear sets

We write $\mathbb{Z}$ for the integers, $\mathbb{N}$ for the nonnegative integers, and $\mathbb{Q}$ for the rational numbers; we follow these notations by a superscript + for the strictly positive subsets ($\mathbb{Z}^+ = \mathbb{N}^+, \mathbb{Q}^+$). We use $\mathbb{K}$ to denote any of $\mathbb{Z}$, $\mathbb{N}$, or $\mathbb{Q}$. When a dimension d is implicit, we write $\mathbf{e}_i \in \{0, 1\}^d$ for the vector having a 1 in position i and 0 elsewhere, and 0^d or $\mathbf{0}$ for the all-zero vector of dimension d. We usually see vectors as *column* vectors, but take liberties for the notation and explicitly indicate when the distinction is important.

Example 1

The semigroup $U_1 = (\{0, 1\}, \times)$ is a monoid with 1 as neutral element.

A *semigroup* is a set A equipped with an associative binary operation $\diamond\colon A \to A$, usually written in infix notation. It is a *monoid* if there exists a (unique) *neutral* element $e \in A$, that is, for all $x \in A$, $e \diamond x = x \diamond e = x$. For a set of elements G, we write G^+ for the semigroup generated by G and G^* for the monoid generated by G. If G consists of a single element x, then G^* can be written x^*. We see the sets $\mathbb{K}^d$ as the additive monoids $(\mathbb{K}^d, +)$, where:

$$(a_1, a_2, \dots, a_d) + (b_1, b_2 \dots, b_d) = (a_1 + b_1, a_2 + b_2, \dots, a_d + b_d) \ .$$

Example 2

The only morphisms from $\mathbb{Z}$ to U_1 are the constant functions.

A *(monoid) morphism* from a monoid $(M, \diamond)$ to a monoid $(N, \star)$ is a function $h\colon M \to N$ such that $h(m \diamond m') = h(m) \star h(m')$ for all $m, m' \in M$ and $h(e_M) = e_N$, with e_M, e_N the respective neutral elements of M and N. The morphism is *erasing* if h maps a nonneutral element of M to e_N.

Vectors in M^d are noted in boldface, $\mathbf{v} \in M^d$, and their elements written in slanted roman with a subscript, $\mathbf{v} = (v_1, v_2, \dots, v_d)$. We often use, without mention, the isomorphism between $M^d \times M^{d'}$ and $M^{d+d'}$. When an ordered set of symbols $\Sigma = \{a_1, a_2, \dots, a_d\}$ is implicit, we refer to the components of $\mathbf{v} \in M^d$ by $v_{a_1}, v_{a_2}, \dots, v_{a_d}$, and v_a is understood as v_{a_i} where $a_i = a$.

For a map $f\colon A \to B$ and $E \subseteq A$, we let $f(E) = \{f(e) \mid e \in E\}$ and $f^{-1}(E) = \{x \mid f(x) \in E\}$. In addition, if $g\colon B \to C$ then $g \circ f$ is the function which maps $x \in A$ to $g(f(x)) \in C$.

A function $f\colon \mathbb{K}^d \to \mathbb{K}^d$ is an *affine function* of dimension d if there exist a matrix $M \in \mathbb{K}^{d\times d}$ and $\mathbf{v} \in \mathbb{K}^d$ such that for any $\mathbf{x} \in \mathbb{K}^d$, $f(\mathbf{x}) = M.\mathbf{x} + \mathbf{v}$. We note such a function $(M, \mathbf{v})$ and abusively write $f = (M, \mathbf{v})$. We let $\mathscr{F}_d^{\mathbb{K}}$, or $\mathscr{F}_d$ when $\mathbb{K}$ is clear, be the monoid of such functions under the operation $\diamond$ defined by $(f \diamond g)(\mathbf{x}) = g(f(\mathbf{x}))$, where the identity element is the identity function, i.e., $(Id, \mathbf{0})$ with Id the identity matrix of dimension d. Let U be a monoid morphism from Σ^* to $\mathscr{F}_d^{\mathbb{K}}$. For $w \in \Sigma^*$, we write U_w for $U(w)$, so that the application of $U(w)$ to a vector $\mathbf{v}$ is written $U_w(\mathbf{v})$, and U_ε is the identity function. We define $\mathscr{M}(U)$ as the multiplicative matrix monoid generated by the matrices used to define U, i.e., $\mathscr{M}(U) = \{M \mid (\exists a \in \Sigma)(\exists \mathbf{v})[U_a = (M, \mathbf{v})]\}^*$.

We now focus on the important notion of *definable set*, of which *semilinear set* is a particularly important special case. A subset $E \subseteq \mathbb{K}^d$ is $\mathbb{K}$-*definable* if it is expressible as a first-order formula which uses the function symbols $+$, λ_c with $c \in \mathbb{K}$ corresponding to the scalar multiplication by c, the order $<$, and constants. More precisely, a subset E of $\mathbb{K}^d$ is $\mathbb{K}$-definable iff there is such a formula ϕ with d free variables, with $\mathbf{x} \in E \Leftrightarrow \mathbb{K} \models \phi(\mathbf{x})$.

We will usually consider $\mathbb{K} = \mathbb{N}$. In this case, the logic together with the interpretation of the relations and symbols in $\mathbb{N}$ is the *Presburger arithmetic*, and definable sets coincide with the *semilinear sets*: A set $E \subseteq \mathbb{N}^d$ is *linear* if $E = \{\mathbf{b}\} + P^*$ for some $\mathbf{b} \in \mathbb{N}^d$ and $P \subseteq \mathbb{N}^d$ a finite set, it is *semilinear* if it is a finite union of linear sets. We will always favor the term "semilinear set" over "$\mathbb{N}$-definable set" in this document. One essential property of these sets, and a way to show they are decidable, is that they are exactly the sets expressible by a quantifier-free first-order formula with the same symbols as before, and added congruence relations ($\equiv_p$ for $p \in \mathbb{N}^+$, interpreted as $x \equiv_p y$ iff p divides $x - y$).

Example 3 — The many forms of semilinearity

Let

$$\phi(x, y, z) \equiv (\exists a) \left[x + \lambda_2(a) = y \land (\forall b) \left[a < b \rightarrow z < b\right]\right] \quad .$$

The subset E of $\mathbb{N}^3$ defined by ϕ is the set of triples (x, y, z) such that $x \leq y$, x and y have the same parity, and z is no greater than half the difference between x and y. It can be written as the quantifier free expression:

$$(x, y, z) \in E \Leftrightarrow x \equiv_2 y \land \lambda_2(z) + x \leq y \quad .$$

This is also the linear set:

$$E = \begin{pmatrix}0\\0\\0\end{pmatrix} + \left\{\begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\2\\1\end{pmatrix}, \begin{pmatrix}0\\2\\0\end{pmatrix}\right\}^* \quad .$$

*
**

The sets $\{x^2 \mid x \in \mathbb{N}\}$ and $\{2^x \mid x \in \mathbb{N}\}$ are not semilinear. Indeed, if a set of dimension 1 is semilinear, it contains an arithmetic progression, which is not the case of those sets. For more complex sets, one can use logical games or logical considerations to show that a set is not semilinear. As a simple example, the set

$$\{(x, y) \mid y < x \land (x \equiv_2 0 \rightarrow y \text{ is a power of } 2)\} \quad ,$$

is not semilinear. Indeed, if it were, it would be described by a formula $\phi(x, y)$, and the formula $\psi(y) \equiv \phi(\lambda_2(y), y)$ would describe the set of powers of two.

Example 4 — Quantifier removal in Presburger arithmetic

We mentioned that the logic with symbols $+$, $<$, and $\equiv_p$, $p \in \mathbb{N}^+$, admits quantifier elimination. We sketch this classical proof, which is done using a bottom-up induction on formulas. Careful syntactic work shows that any formula $(\exists x)[\beta_1 \wedge \beta_2 \wedge \cdots \wedge \beta_k]$, where each β_i is an atomic formula or the negation of one, can be written as:

$$(\exists x)\left[\bigwedge_{j<\ell} r_j - s_j < x \wedge \bigwedge_{i<k} x < t_i - u_i \wedge \bigwedge_{i<n} x \equiv_{m_i} v_i - w_i\right] ,$$

where r_i, s_i, t_i, u_i, v_i, and w_i are terms not containing x, and the formulas with a subtraction sign are abbreviations for the formulas without it, obtained by transposing terms.

Next, if there are no congruences ($n = 0$), then the formula can be written as the quantifier-free formula:

$$\bigwedge_{i<k} \bigwedge_{j<\ell} (r_j - s_j) + 1 < t_i - u_i \wedge \bigwedge_{i<k} 0 < t_i - u_i \enspace .$$

Otherwise, let M be the least common multiple of the m_i's, so that $a + M \equiv_{m_i} a$. Clearly, as a increases, the pattern of residues of a modulo $m_0, m_1, \ldots, m_{n-1}$ has period M. Hence it is enough to search for a solution to the congruences within M consecutive integers, starting from the lower bounds $r_j - s_j$. The formula is thus equivalent to:

$$\begin{aligned}\bigvee_{j<\ell} \bigvee_{1\leq c\leq M} \Bigg[& \bigwedge_{i<\ell} r_i - s_i < (r_j - s_j) + c \\ & \wedge \bigwedge_{i<k} r_j - s_j + c < (t_i - u_i) \\ & \wedge \bigwedge_{i<n} r_j - s_j + c \equiv_{m_i} v_i - w_i \Bigg] \enspace .\end{aligned}$$

When $\mathbb{K} = \mathbb{Q}$, definable sets coincide with the *semialgebraic set* (of degree 1): a set $E \subseteq \mathbb{Q}^d$ is $\mathbb{Q}$-definable iff it is a finite union of sets of the form:

$$\{\mathbf{x} \mid f_1(\mathbf{x}) = f_2(\mathbf{x}) = \cdots = f_p(\mathbf{x}) = 0 \wedge \\ g_1(\mathbf{x}) > 0 \wedge g_2(\mathbf{x}) > 0 \wedge \cdots \wedge g_q(\mathbf{x}) > 0\} \ , \qquad (1)$$

where $f_1, f_2, \ldots, f_p, g_1, g_2, \ldots, g_q \colon \mathbb{Q}^d \to \mathbb{Q}$ are polynomials of degree 1 over $\mathbb{Q}$; this shows in particular that over $\mathbb{Q}$ the formulas previously described admit quantifier elimination — this time, without requiring the addition of new relations.

In the context of a computational problem, we say that a $\mathbb{K}$-definable set is *effectively definable* (or effectively semilinear, effectively semialgebraic) if its description as a formula, or any equivalent form, can be computed from the input to the problem.

Bibliographical notes and comments

Any introductory book on algebra (e.g., [Jac09]) carries the basic definitions of monoid and morphism that we give here. As our focus is on language theory, the reader will probably find a better suited introduction in [Gin68].

The decidability of the arithmetic with only $+$ is due to Presburger [Pre27], hence the name “Presburger arithmetic.” The sketch given here is adapted from the proof in [End72]. Another line of attack is the technique of Büchi [Büc60] (see, e.g., [Str94, Ex. III.5]): any formula ϕ with d free variables can be encoded into a finite automaton over the alphabet $\{0, 1\}^d$ whose language is a binary encoding of the vectors true of ϕ — the converse is false, but we can decide, given an automaton, whether it encodes such a formula [Ler05]. Lastly, a body of research exists on *Presburger functions*, i.e., functions with semilinear graphs (see [FL08] for a review). The set of these functions is the closure under composition of projection functions, sum and positive subtraction, successor, conditional ($x, y \mapsto x$ if $y \neq 0$, and 0 otherwise), and division [IL81]; as a formula $\phi(\mathbf{x})$ can be seen as a Presburger function (with range $\{0, 1\}$) and the translation from formulas to the function composition tree is effective, this gives another proof of the decidability of Presburger arithmetic.

Semilinear sets as finite unions of linear sets, on the other hand, were introduced by Parikh [Par66] in his seminal work on the commutative image of context-free languages (see next section). These were investigated by Ginsburg and Spanier [GS66c] who showed their equivalence with $\mathbb{N}$-definable sets.

The semilinear sets are also called “rational subsets of $\mathbb{N}^d$,” this term being often found in the literature on rational power series. The reason is that the class of semilinear sets is the smallest class containing the finite sets and closed under union, complement, and starring (i.e., taking P^*), also known as the rational operators. See for instance [ES69].

Semialgebraic sets are finite unions of sets of the form of Equation (1) with the f_i's and g_i's being *polynomials*; the degree of a semialgrebraic set is the maximum degree of these polynomials. When this degree is 1, the polynomials are linear functions, and the sets defined are then called *semilinear* — we will however reserve this term for $\mathbb{N}$-definable sets. Semialgebraic sets arise in the study of o-minimal structures, which are generalizations of $\mathbb{K}$-definable sets over more exotic choices of $\mathbb{K}$ and of predicates, with the added constraint that the only sets of dimension 1 definable are the finite unions of intervals and points. We hinted that the formulas used for $\mathbb{Q}$-definable sets admit quantifier elimination; this is in fact true for any ordered F-linear space, with F an ordered field. A related important result is the Tarski–Seidenberg theorem, which states that the formulas admit quantifier elimination even if multiplication is allowed — in this case, the sets defined are all the semialgebraic sets. See [vdD98] for a modern account of these results.

2 Languages

Let Σ be an *alphabet*, i.e., a finite set of symbols called *letters*. The *free monoid* generated by Σ under the operation of concatenation is written Σ^* and has ε, the *empty word*, as neutral element — hence $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. The elements of Σ^* are the *words*, while subsets of Σ^* are the *languages*. It is usually assumed that $|\Sigma| \geq 2$, and the case $\Sigma = \{a\}$ is treated separately. The number of letters composing a word is called its *length*. For $w \in \Sigma^*$ and $a \in \Sigma$, $|w|_a$ is the number of occurrences of a in w. It is easily seen that the length operation or the letter count operations are morphisms from Σ^* to $(\mathbb{N}, +)$. For two words $u, v \in \Sigma^*$, we write their concatenation as either uv or $u \cdot v$; the word uu is written u^2, and more generally $u^n = uu^{n-1}$. For a word $w = w_1 w_2 \cdots w_n$, where $w_i \in \Sigma$, we let w^{R} be the *reversal* of w, that is:

$$w^{\mathrm{R}} = (w_1 w_2 \cdots w_n)^{\mathrm{R}} = w_n \cdots w_2 w_1 \ .$$

We naturally extend the operations of concatenation, powering, and reversal to languages, letting $L \cdot L' = LL' = \{uv \mid u \in L \wedge v \in L'\}$, $L^1 = L$, $L^n = LL^{n-1}$, and $L^{\mathrm{R}} = \{w^{\mathrm{R}} \mid w \in L\}$. For two languages L, L', we define the *left quotient* of L' by L as:

$$L^{-1}L' = \{v \mid \exists u \in L, uv \in L'\} \ ,$$

and similarly, the *right quotient* of L' by L as:

$$L'L^{-1} = \{u \mid \exists v \in L, uv \in L'\} \ .$$

Example 5

For any nonempty $L, L' \subseteq \Sigma^*$, $L^{-1}\Sigma^* = \Sigma^*$; $L^{-1}\{\varepsilon\} = \{\varepsilon\}$ iff $\varepsilon \in L$; $\varepsilon \in L^{-1}L'$ iff $L \cap L'$ is nonempty.

A morphism from Σ^* to some other monoid need only be defined on the letters. A *morphism between languages* is a monoid morphism h from Σ^* to T^* with Σ and T two alphabets; it is said to be *length-preserving* if $h(a) \in \mathrm{T}$ for all $a \in \Sigma$.

For a language $L \subseteq \Sigma^*$, define the *Nerode equivalence relation* $\equiv^{\mathrm{N}}_L$, or $\equiv^{\mathrm{N}}$ when L is clear, on Σ^* by $u \equiv^{\mathrm{N}}_L v$ iff for all $w \in \Sigma^*$, $uw \in L \leftrightarrow vw \in L$. Intuitively, if $u \equiv^{\mathrm{N}}_L v$, then if u appears as a prefix of a word, we may change this prefix to v without changing the membership of the word in the language. We write $[u]^{\mathrm{N}}_L$ for the equivalence class of u under $\equiv^{\mathrm{N}}_L$, or $[u]^{\mathrm{N}}$ when L is clear from the context.

Example 6

The language $\{\ell bw \mid \ell \in \{a, b\} \wedge w \in \{a, b\}^*\}$ has four Nerode equivalence classes, $[\varepsilon]^{\mathrm{N}}$, $[a]^{\mathrm{N}} = [b]^{\mathrm{N}}$, $[\ell b]^{\mathrm{N}}$, $[\ell a]^{\mathrm{N}}$. The operation mapping $([u]^{\mathrm{N}}, [v]^{\mathrm{N}})$ to $[uv]^{\mathrm{N}}$ is not well-defined.

The *Parikh image* (or *commutative image*) of a word is the vector of its letter counts. More precisely, for $w \in \Sigma^*$, with $\Sigma = \{a_1, a_2, \ldots, a_n\}$, we define:

$$\mathsf{Pkh}(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n}) \ .$$

Equivalently, $\mathsf{Pkh}(\cdot)$ is a morphism from Σ^* to $\mathbb{N}^n$ defined by $\mathsf{Pkh}(a_i) = \mathbf{e}_i$ for $1 \leq i \leq n$. A language L is said to be *semilinear* if $\mathsf{Pkh}(L)$ is semilinear. The *commutative closure* of a language L is $c(L) = \mathsf{Pkh}^{-1}(\mathsf{Pkh}(L))$, that is, the set of words whose Parikh image is the Parikh image of a word in L.

A language $L \subseteq \Sigma^*$ is *bounded* if there exist words $w_1, w_2, \ldots, w_n \in \Sigma^+$ such that:

$$L \subseteq w_1^* w_2^* \cdots w_n^* \ .$$

The tuple $(w_1, w_2, \ldots, w_n)$ is called a *socle*[1] of L. The concatenation, union, or intersection of two bounded languages is bounded, as is any subset of a bounded language. The canonical example of a nonbounded language is Σ^* (whenever $|\Sigma| \geq 2$, which is our usual assumption). This implies that the complement of a bounded language is never bounded, as $\Sigma^* = L \cup \overline{L}$ for any L. The *iteration set* of a bounded language L w.r.t. a socle $(w_1, w_2, \ldots, w_n)$ is the set:

$$\mathsf{Iter}_{w_1,w_2,\ldots,w_n}(L) = \{(i_1, i_2, \ldots, i_n) \in \mathbb{N}^n \mid w_1^{i_1} w_2^{i_2} \ldots w_n^{i_n} \in L\} \ .$$

Note that if a word $u \in L$ can be written as $w_1^{i_1} w_2^{i_2} \cdots w_n^{i_n}$ and $w_1^{j_1} w_2^{j_2} \cdots w_n^{j_n}$, then both $\mathbf{i}$ and $\mathbf{j}$ belong to $\mathsf{Iter}_{\mathbf{w}}(L)$. Bounded languages generally exhibit better decidability proprieties within a given language class (e.g., given two context-free languages, one of them bounded, it is decidable whether one is included in the other), while still offering a useful expressiveness. We write BOUNDED for the class of bounded languages and BSL for the class of bounded languages L for which there is a socle $\mathbf{w}$ such that $\mathsf{Iter}_{\mathbf{w}}(L)$ is semilinear.

Example 7

The following are bounded languages: a^*b^*, a regular language, $\{a^nb^n\}$, a nonregular context-free language, $\{a^nb^nc^n\}$, a noncontext-free context-sensitive language, $\{a^n \mid n \text{ is the code of an halting Turing machine}\}$, a nonrecursive recursively enumerable language.

Example 8 — Parikh images

The Parikh image of a word is a fundamental concept for this work. Its name stems from *Parikh's theorem*, which asserts that any context-free language is semilinear. Parikh's theorem implies, in particular, that over a unary alphabet, context-free languages are the same as regular languages — proofs of this fact appeared before Parikh's theorem, however.

Let us now consider a few languages and their Parikh image. Set $\Sigma = \{a, b\}$.

For $\{ww^{\mathrm{R}} \mid w \in \Sigma^*\}$, we have that each letter appears twice, once in w and once in w^{R}. Thus its Parikh image is $(2\mathbb{N})^2$. For $\{aa, bb\}^*$, the same holds. More generally, it is easy to show that any semilinear set is the Parikh image of a regular language. Moreover, the fact that regular languages all have a semilinear Parikh image is not hard to show (it is in particular easier than Parikh's theorem), as we sketch here. Let L be a regular language, we rewrite L, using a bottom-up induction,

[1]The term *socle* was introduced in Paper II.

as $\bigcup_{i=1}^{n} w_i W_i^*$ with w_i words and W_i finite sets of words, while keeping its Parikh image. No work is needed if L is a finite language or if $L = L_1 + L_2$, by induction. Now if $L = L_1 L_2$, with $L_1 = \bigcup_{i=1}^{m} u_i U_i^*$ and $L_2 = \bigcup_{j=1}^{n} v_j V_j^*$, then rewrite L as $\bigcup_{i=1}^{m} \bigcup_{j=1}^{n} u_i v_i (U_i \cup V_i)^*$, which has the same Parikh image as L. If $L = (L')^*$, with $L' = \bigcup_{i=1}^{n} w_i W_i^*$, then rewrite L as

$$\bigcup_{I \subseteq \{1,2,\ldots,n\}} \left(\prod_{i \in I} w_i\right) \left(\bigcup_{i \in I} (W_i \cup \{w_i\})\right)^* ,$$

which has the same Parikh image as L (if $I = \emptyset$ then the expression values to $\{\varepsilon\}$). Now the Parikh image of a language wW^* is simply $\mathsf{Pkh}(w) + \{\mathsf{Pkh}(u) \mid u \in W\}^*$, a linear set, and thus L has a semilinear Parikh image.

Parikh's theorem is thus equivalent to: for any context-free language L there is a *regular* language L' such that $\mathsf{Pkh}(L) = \mathsf{Pkh}(L')$.

Let us consider the nonbounded regular language $R = \{a \cdot u \mid u \in \{a,b\}^* \wedge |u| \text{ is even}\}^*$ (mind the ending star). Then if $w \in R$ has only one a, $|w|_b$ should be even. If it has more than one a, then no restriction is put on the number of b's. Hence:

$$\mathsf{Pkh}(R) = \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\} \cup \left(\begin{pmatrix}1\\0\end{pmatrix} + \left\{\begin{pmatrix}0\\2\end{pmatrix}\right\}^*\right) \cup \left(\begin{pmatrix}2\\0\end{pmatrix} + \left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right\}^*\right) .$$

Note that this is also the Parikh image of the language:

$$\{\varepsilon\} \cup \{a\}^+ \cdot \{ba, bb\}^* ,$$

which is a bounded regular language *within* R.

Finally, note that $\{a^n b^n\}$ is nonregular context-free and $\{a^n b^n c^n\}$ is noncontext-free, thus bounded regular languages form a strict subset of bounded context-free languages, themselves a strict subset of BSL.

Bibliographical notes and comments

The study of languages and words independently of a model of computation is usually focused on combinatorics, in a field known as *combinatorics on words* (a classic textbook is that of the Lothaire group [Lot97]). We will not go deep into this field, but will make good use of it in Paper II. The presentation adopted here is somewhat specific to our needs. Basic language theory is covered in classic textbooks such as [HU79, Sip97], which go through finite automata and the Chomsky hierarchy. As mentioned previously, the Parikh map, or commutative map, has been considered by Parikh [Par66] to show that the Parikh image of a context-free language is effectively semilinear (see also [Gin66] for a proof). As it is checkable whether a semilinear set is empty or finite — this is particularly easy when presented with the finite union of linear sets $\{\mathbf{b}\} + P^*$, each of them given by $\mathbf{b}$ and P — this implies that one can check whether a context-free language is empty or finite.

Bounded languages were heavily studied in the 1960s by Ginsburg and Spanier (see [Gin66, Chap. 5] for a self-contained presentation of the results of this period). In particular, they characterized the iteration sets of bounded regular and context-free languages. We will use one such characterization in Paper III: call unary a semilinear set C that can be written as $\bigcup_{i=1}^{n}(\{\mathbf{b}_i\}+P_i^*)$ where the vectors of the P_i's have only one nonzero component, then a bounded language is regular iff one of its iteration sets is unary. A result of Restivo [Res75] characterizes the regular bounded languages with a combinatorics on words flavor: a regular language is bounded iff for any p, the number of words of the language with no factor repeated p times is finite. A class $\mathscr{C}$ having the property that any language $L \in \mathscr{C}$ has a subset which is a bounded language in $\mathscr{C}$ with the same Parikh image as L is said to be *Parikh-bounded*. The regular and context-free languages have this property ([Lat78] and [BL81], respectively); for a recent account in the case of context-free languages, see [GMM10].

3 Models of computation

We consider in this work several different models of computations, some of them being an integral part of the contribution of this thesis. In this section we present these models: first the usual finite automaton, then the different constrained versions of automata under study, and finally the reversal-bounded counter machine.

3.1 Regular languages and automata

The class of *regular languages*, REG, is the smallest class of languages containing the finite languages and closed under union, concatenation, and starring (i.e., taking L^* from L). It is closed under a wealth of other operations, e.g., complement, intersection, morphism, reversal, and quotient. The class of regular languages is precisely the class of languages having a finite number of Nerode equivalence classes.

Regular languages are also those recognized by finite automata. A *finite automaton*, more simply, an automaton, is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. For a transition $t = (q, a, q') \in \delta$, we write $t = q \bullet\!\!-a\!\!\rightarrow q'$ and define $\mathsf{From}(t) = q$ and $\mathsf{To}(t) = q'$. We define $\mu_A : \delta^* \to \Sigma^*$ as the length-preserving morphism given by $\mu_A(t) = a$, with, in particular, $\mu_A(\varepsilon) = \varepsilon$, and write μ when A is clear from the context. A *path* π on A is a word $\pi = t_1 t_2 \cdots t_n \in \delta^*$ such that $\mathsf{To}(t_i) = \mathsf{From}(t_{i+1})$ for $1 \leq i < n$; we extend From and To to paths, letting $\mathsf{From}(\pi) = \mathsf{From}(t_1)$ and $\mathsf{To}(\pi) = \mathsf{To}(t_n)$. We say that $\mu(\pi)$ is the *label* of π. A path π is *initial* if $\mathsf{From}(\pi) = q_0$, *final* if $\mathsf{To}(\pi) \in F$, and *accepting* if it is both initial and final; we write $\mathsf{Run}(A)$ for the language over δ of accepting paths (or *runs*) on A. We write $L(A)$ for the language of A, i.e., the labels of the accepting paths. The automaton A is *deterministic* if $(p \bullet\!\!-a\!\!\rightarrow q \in \delta \wedge p \bullet\!\!-a\!\!\rightarrow q' \in \delta)$ implies $q = q'$.

An ε-automaton is an automaton $A = (Q, \Sigma, \delta, q_0, F)$ as above, except with $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ so that in particular μ_A becomes an erasing morphism.

An (ε-)automaton A is said to be *unambiguous* if each word in $L(A)$ is the label of only one path in $\mathsf{Run}(A)$.

3.2 Parikh automata

The main focus of this thesis is models of computation in which a semilinear constraint is applied to the acceptance of a path in an automaton. In this section and the next, we present the models and notions at play.

A Parikh Automaton (PA) of dimension d over Σ is a pair (A, C) where $C \subseteq \mathbb{N}^d$ is a semilinear set and A is an automaton over an alphabet $\Sigma \times D$ where D is a finite subset of $\mathbb{N}^d$. Let $\Psi\colon (\Sigma \times D)^* \to \Sigma^*$ and $\Phi\colon (\Sigma \times D)^* \to \mathbb{N}^d$ be the morphisms defined by:

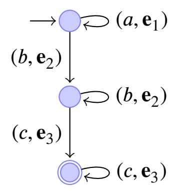
$$\Psi((a, \mathbf{x})) = a \ ,$$
$$\Phi((a, \mathbf{x})) = \mathbf{x} \ .$$

Recall that $\mathbb{N}^d$ is seen as the monoid $(\mathbb{N}^d, +)$, and thus $\Phi(\,(a, \mathbf{x})(b, \mathbf{y})\,) = \mathbf{x} + \mathbf{y}$, with the addition being done component-wise. The morphism Ψ is called the *projection on* Σ and Φ the *extended Parikh image*. Now the language of the PA is defined as:

$$L(A, C) = \{\Psi(\omega) \mid \omega \in L(A) \wedge \Phi(\omega) \in C\} \ .$$

Example 9

The DetPA of dimension 3 given by

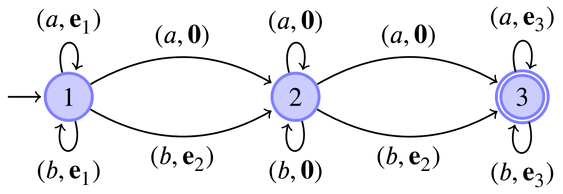


and the set $\{(n, n, n) \mid n \in \mathbb{N}\}$ recognizes $\{a^n b^n c^n \mid n \geq 1\}$.

The PA is said to be *deterministic* (*DetPA*) if for every state q of A and every $a \in \Sigma$, there exists at most one pair $(q', \mathbf{v})$ with q' a state and $\mathbf{v} \in D$, such that $(q, (a, \mathbf{v}), q')$ is a transition of A. The class of languages recognized by PA (resp. DetPA) is written $\mathscr{L}_{\text{PA}}$ (resp. $\mathscr{L}_{\text{DetPA}}$).

Example 10 — Complement of palindromes

Example 9 shows that PA are able to *count* more than context-free grammars. Here, we give a PA of dimension 3 for the nonpalindromes, NoPAL $= \{u \cdot v^{\text{R}} \mid u, v \in \{a, b\}^* \wedge |u| = |v| \wedge u \neq v\}$. The PA is given by the automaton A:



Here, $\mathbf{e}_i$ can be seen as *incrementing* the i-th register while $\mathbf{0}$ is a no-op. The automaton counts i letters in the first register, chooses nondeterministically to go to the second state, then the third, and counts j letters from the suffix in the third register. The second register is 1 iff the letter read while going from state 1 to state 2 is different from the one read while going from state 2 to state 3. Thus, a word is in NoPAL iff such guesses lead to $i = j$ and the second register is equal to 1, as this means that the word is of the form $x\ell_1 y\ell_2 z$ where $x, y, z \in \{a, b\}^*$, $\ell_1, \ell_2 \in \{a, b\}$, and $|x| = |z|$ and $\ell_1 \neq \ell_2$. Thus, letting:

$$C = \{(i, 1, i) \in \mathbb{N}^3 \mid i \in \mathbb{N}\} \ ,$$

we have that $L(A, C) \cap (\{a, b\}^2)^* = \text{NoPAL}$. Note that C is indeed semilinear. Of course, there are PA of lesser dimension for this language, though they may have more states; we chose to present this PA as it exhibits more specific features of the model.

Parikh Automata were introduced by Klaedtke and Rueß [KR03], and subsequently studied in Karianto's diploma thesis [Kar04], in which he gave more details on the proofs of [KR03] and considered some variants. The rest of this section is devoted to the basic results of these works.

▶ **Proposition 1.** *Let Σ be an alphabet and $D \subseteq \mathbb{N}^d$. If $L \subseteq (\Sigma \times D)^*$ is a semilinear language, then $\Psi(L) \subseteq \Sigma^*$ is a semilinear language.*

Of course, the converse is false. However, this implies that $\mathscr{L}_{\text{PA}}$ is a class of semilinear languages; and this is effective if the automaton and the constraint set are given.

A combinatorial argument is used in showing:

▶ **Proposition 2.** *The language of palindromes is not in $\mathscr{L}_{\text{PA}}$.*

The main idea of the proof is that addition over $(\mathbb{N}^d, +)$ being commutative, there is only a polynomial number of different vectors that can be obtained by adding n vectors from a finite set $D \subseteq \mathbb{N}^d$. The language of palindromes would require an exponential number of these to distinguish between all words. See [KR03, Lemma 26] for the details. A similar argument is used in showing Lemma 11 in Paper I.

The following proposition lists the closure properties of $\mathscr{L}_{\text{PA}}$ and $\mathscr{L}_{\text{DetPA}}$ known prior to the work of this thesis.

▶ **Proposition 3.** *The class $\mathscr{L}_{\text{PA}}$ is closed under union, intersection, concatenation, morphisms, and inverse morphisms. It is not closed under complement.*

The class $\mathscr{L}_{\text{DetPA}}$ is closed under union, intersection, complement, and inverse morphisms. It is not closed under morphism.

Similarly with decidability properties:

▶ **Proposition 4.** *Emptiness is decidable for $\mathscr{L}_{\text{PA}}$. Universality is decidable for $\mathscr{L}_{\text{DetPA}}$ but not for $\mathscr{L}_{\text{PA}}$.*

3.3 Constrained languages, constrained automata

The models of this section are one of the original contributions of this thesis. We present them here so that redundancy is reduced among the different articles.

Given a language $L \subseteq \Sigma^*$ and a set $C \subseteq \mathbb{N}^{|\Sigma|}$, the *language L constrained to C*, written $L{\restriction_C}$, is defined as:

$$L{\restriction_C} = L \cap \mathsf{Pkh}^{-1}(C) \enspace ,$$

or, in other words, $L{\restriction_C}$ is the set of words in L with a Parikh image in C.

Example 11 — Constrained regular languages

We show that the set S of constrained *regular* languages are incomparable with the set CFL of context-free languages. Firstly,

$$a^*b^*c^*{\restriction_{\{(i,i,i) \mid i \in \mathbb{N}\}}} = \{a^ib^ic^i \mid i \in \mathbb{N}\} \enspace ,$$

a noncontext-free language, so that $S \not\subseteq$ CFL. Secondly, suppose to the contrary that there is a regular language L and a set C such that $L{\restriction_C} =$ PAL, the language of palindromes defined as $\{w\#w^{\mathrm{R}} \mid w \in \{a,b\}^*\}$. We may suppose that all the words of L have an even number of a's and b's, as this does not change $L{\restriction_C}$ or the fact that L is regular. As $L \supsetneq$ PAL, there is a word $w \in L$ which is not a palindrome. However, $\mathsf{Pkh}(w) \in C$ as $\mathsf{Pkh}(w) = \mathsf{Pkh}(a^{\frac{|w|_a}{2}} b^{|w|_b} a^{\frac{|w|_a}{2}})$ and the latter word is a palindrome. Thus $w \in L{\restriction_C} =$ PAL, a contradiction. Hence PAL $\in$ CFL $\setminus S$

Example 12 — Parikh automata and constrained languages

Let $\Sigma = \{a_1, a_2, \dots, a_n\}$, $L \subseteq \Sigma^*$ be a regular language, and $C \subseteq \mathbb{N}^n$ be a semilinear set. Consider an automaton A for L with each label a_i changed to $(a_i, \mathbf{e}_i)$. Then (A, C) can be seen as a PA, the language of which is $L{\restriction_C}$.

A Constrained Automaton (CA) is a pair (A, C) where A is an ε-automaton with d transitions and $C \subseteq \mathbb{N}^d$ is a semilinear set. Its language is

$$L(A, C) = \mu_A(\mathsf{Run}(A){\restriction_C}) \enspace .$$

The CA is said to be *deterministic* (DetCA) if A is deterministic and *unambiguous* (UnCA) if A is unambiguous. The class of languages recognized by CA (resp. DetCA, UnCA) is written $\mathscr{L}_{\text{CA}}$ (resp. $\mathscr{L}_{\text{DetCA}}$, $\mathscr{L}_{\text{UnCA}}$). We will see that $\mathscr{L}_{\text{CA}}$ is closed under morphisms, this implies that any morphic image of a semilinearly constrained regular language is in $\mathscr{L}_{\text{CA}}$, and conversely.

It is easy to see, and a full proof is found in Paper I (Theorem 6), that PA (resp. DetPA) and CA (resp. DetCA) recognize the same languages; we usually prefer working with CA as the simplicity of its definition helps in devising shorter proofs.

Example 13 — Closure of $\mathscr{L}_{\text{CA}}$ under intersection

We give a proof that $\mathscr{L}_{\text{CA}}$ is closed under intersection based on the characterization by morphisms. Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Let $L, L' \subseteq \Sigma^*$ be languages of $\mathscr{L}_{\text{CA}}$. There exist length-preserving morphisms $h, h' : \Sigma^* \to \Sigma^*$, regular languages $R, R' \subseteq \Sigma^*$, and semilinear sets $C, C' \subseteq \mathbb{N}^n$ such that $L = h(R\restriction_C)$ and $L' = h'(R'\restriction'_C)$. Let $\ddot{\Sigma} = \{\ddot{a}_1, \ddot{a}_2, \ldots, \ddot{a}_n\}$ be a new set of symbols, and let $\ddot{h}'$, $\ddot{R}'$ be the versions of h' and R' for $\ddot{\Sigma}$ instead of Σ. Extend C (resp. C') to vectors of dimension $|\Sigma \cup \ddot{\Sigma}|$ by disregarding the last (resp. first) n components — $\Sigma \cup \ddot{\Sigma}$ is ordered with Σ first then $\ddot{\Sigma}$. Finally, extend h to work on $\Sigma \cup \ddot{\Sigma}$ by mapping the letters of $\ddot{\Sigma}$ to the empty word. Now the language:

$$\begin{aligned} E =& \{u_1\ddot{v}_1u_2\ddot{v}_2 \cdots u_k\ddot{v}_k \mid \\ & \quad u = u_1u_2\cdots u_k \in R \wedge \ddot{v} = \ddot{v}_1\ddot{v}_2\cdots\ddot{v}_k \in \ddot{R}' \wedge h(u) = \ddot{h}'(\ddot{v})\} \end{aligned}$$

is regular, and $L \cap L' = h(E\restriction_{C\cap C'})$, thus $L \cap L' \in \mathscr{L}_{\text{CA}}$.

A $\mathbb{K}$-*affine Parikh automaton* ($\mathbb{K}$-APA) of dimension d is a triple (A, U, C) where $A = (Q, \Sigma, \delta, q_0, F)$, U is a morphism from δ^* to $\mathscr{F}_d^{\mathbb{K}}$, and $C \subseteq \mathbb{K}^d$ is a $\mathbb{K}$-definable set. Recall that U need only be defined on δ, and that, in particular, $U(\varepsilon)$ is the identity function, recall also that we write U_π for $U(\pi)$. The language of the APA is:

$$L(A, U, C) = \mu_A(\{\pi \in \mathsf{Run}(A) \mid U_\pi(\mathbf{0}) \in C\}) \ .$$

The $\mathbb{K}$-APA is *deterministic* ($\mathbb{K}$-DetAPA) or *unambiguous* ($\mathbb{K}$-UnAPA) if A is. We write $\mathscr{L}_{\mathbb{K}\text{-APA}}$ (resp. $\mathscr{L}_{\mathbb{K}\text{-DetAPA}}$, $\mathscr{L}_{\mathbb{K}\text{-UnAPA}}$) for the class of languages recognized by $\mathbb{K}$-APA (resp. $\mathbb{K}$-DetAPA, $\mathbb{K}$-UnAPA).

3.4 Reversal-bounded counter machines

A *one-way, k-counter machine M* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, $\delta \subseteq Q \times (\Sigma \cup \{\lhd\}) \times \{0,1\}^k \times Q \times \{\mathrm{S}, \mathrm{R}\} \times \{-1, 0, +1\}^k$ is the transition function, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. Moreover, we assume that $\lhd \notin \Sigma$. The machine is *deterministic* if for any $(p, \ell, \mathbf{x})$, there exists at most one $(q, h, \mathbf{v})$ such that $(p, \ell, \mathbf{x}, q, h, \mathbf{v}) \in \delta$.

On input w, the machine starts with a read-only tape containing $w\lhd$, and its head on the first character of w. Let c_i denote the value of the i-th counter, then a transition $(p, \ell, \mathbf{x}, q, h, \mathbf{v}) \in \delta$ is taken if the machine is in state p, reading character ℓ, and $c_i = 0$ if $x_i = 0$ and $c_i > 0$ if $x_i = 1$, for all i. The machine then enters state q, its head is moved to the right iff $h = \mathrm{R}$, and v_i is added to c_i, for all i. If the head falls off the tape, or if a counter turns negative, the machine rejects. A word is accepted if an execution leads to a final state.

The machine is *reversal-bounded* (*RBCM*) if there exists an integer r such that any accepting run changes between increments and decrements of the counters a (bounded) number of times less than r. We write DetRBCM for deterministic RBCM. We write $\mathscr{L}_{\text{RBCM}}$ (resp. $\mathscr{L}_{\text{DetRBCM}}$) for the class of languages recognized by RBCM (resp. DetRBCM).

Example 14

Let $L = \{a^n b^n \mid n \in \mathbb{N}\}$. For any k, L^k is the language of an RBCM, but L^* is not.

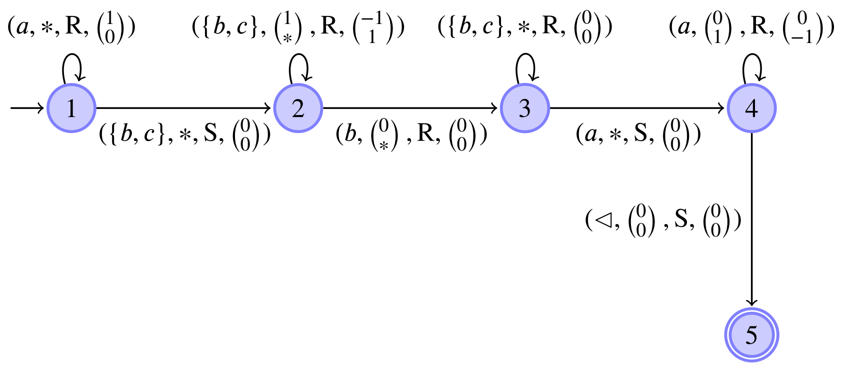
Example 15 — A DetRBCM for a language not in $\mathscr{L}_{\text{DetPA}}$

We give a deterministic RBCM for the language

$$L = \{a^n w a^n \mid w \in \{b, c\}^* \wedge w_{n+1} = b\} \ .$$

The RBCM will count the number of a's in a counter c_1, then, while reading w, will decrease c_1 while incrementing a second counter c_2. When c_1 reaches 0, the letter under the input head should be a b. Lastly, the machine checks that the number of a's at the end of the input is equal to the number in c_2.

We label a transition between states p and q with $(\ell, \mathbf{s}, D, \mathbf{m})$ to mean that a transition $(p, \ell, \mathbf{s}, q, D, \mathbf{m})$ exists in the machine. We use $*$ for a wild-card and use sets instead of ℓ with the obvious meaning. The RBCM is then:


$(a, *, \mathrm{R}, \binom{1}{0})$
$(\{b, c\}, \binom{1}{*}, \mathrm{R}, \binom{-1}{1})$
$(\{b, c\}, *, \mathrm{R}, \binom{0}{0})$
$(a, \binom{0}{1}, \mathrm{R}, \binom{0}{-1})$
1
2
3
4
$(\{b, c\}, *, \mathrm{S}, \binom{0}{0})$
$(b, \binom{0}{*}, \mathrm{R}, \binom{0}{0})$
$(a, *, \mathrm{S}, \binom{0}{0})$
$(\lhd, \binom{0}{0}, \mathrm{S}, \binom{0}{0})$
5

A CA can be built for L, but we can prove, using the tools of Paper I, that no DetCA exists for it. Intuitively, this is because CA use a two-way process: they first follow a path in the automaton, then check that its Parikh image is in the constraint set, thus the automaton of DetCA would have to deterministically find the position in w referred to by the number of a's, and this is an arithmetic task which is not available to finite automata.

RBCM are relevant in the study of our models as:

▶ **Proposition 5** ([KR03]). $\mathscr{L}_{\text{RBCM}} = \mathscr{L}_{\text{PA}}$.

Bibliographical notes and comments

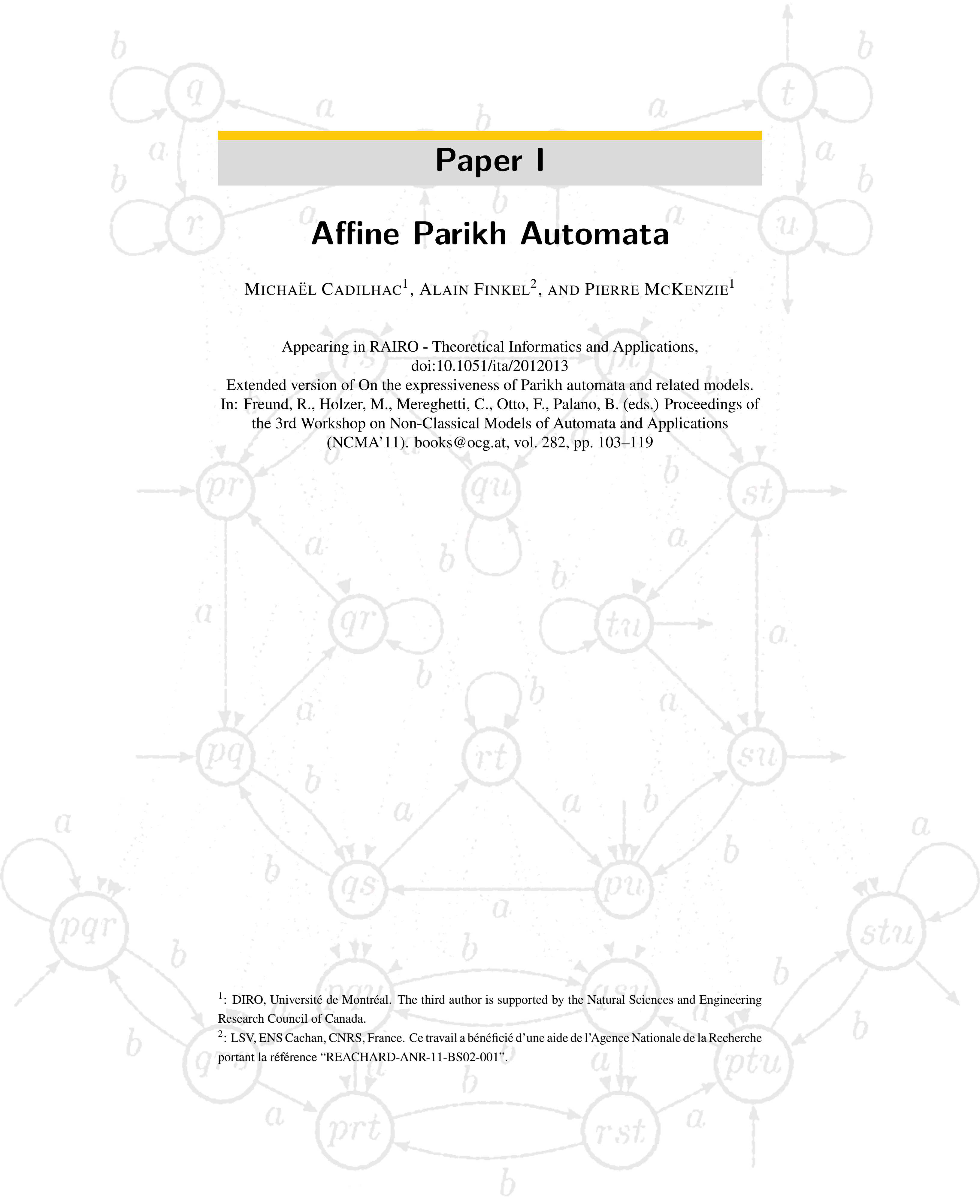
Finite automata are one of the building blocks of language theory and theoretical computer science in general. Any introductory book on theoretical computer science should contain the general properties of automata — including closures, equivalences with other models, and more.

Algebra plays an important role in modern automata theory. Firstly, the *weighted automata* compute a weight associated with the input word; the usual finite automaton fits in this model, as the weight is a Boolean value indicating whether the word is in the language. They are strongly linked to rational power series. A comprehensive book on the subject is [Sak03]. Secondly, for a language $L \subseteq \Sigma^*$, the equivalence classes of the coarsest congruence on Σ^* saturating L form a monoid structure called the *syntactic monoid* of L. It is easily shown that this monoid is finite iff L is regular. A study of the classes of languages resulting from constraining the monoids with a given property has been initiated in the 1960s (see, e.g., [Gin68, Eil76]). This helped in unveiling the fascinating triple play between circuit complexity, logic, and algebra, which is presented in, e.g., Straubing [Str94].

Parikh automata were introduced by Klaedtke and Rueß [KR03] with both model-checking and logic decidability in mind. (It should be noted that the term "constrained language" is used by Klaedtke and Rueß with a different meaning than the one in the present work.) They showed that some natural logic on finite words but infinite universe is equivalent to PA, thus proving some decidability results on the checkability of the formulas in this logic. In his diploma thesis, Karianto [Kar04] expanded on the proofs of Klaedtke and Rueß and considered Parikh automata with pushdown stack [Kar05]. He also worked on extending PA to more general constraint sets [KKT06].

Constrained languages and constrained automata, in the form presented here, were introduced in Paper I. Prior uses of a similar model can be found in [BH99] under the name *constraint-queue decision diagram*. We investigate the exact relationship between these models in Paper II.

Reversal-bounded counter machines emerged from the 1960s systematic study of automata augmented with counters. These machines express at least the regular languages and Minsky [Min61] showed that two counters were enough to simulate Turing machines. Bounding the number of reversals thus came as a good compromise, as Ibarra [Iba78] showed. There is still active research concerning these machines and their variants, e.g., [ISD$^+$02, IS11].



Paper I

Affine Parikh Automata

Michaël Cadilhac[1], Alain Finkel[2], and Pierre McKenzie[1]

Appearing in RAIRO - Theoretical Informatics and Applications,
doi:10.1051/ita/2012013
Extended version of On the expressiveness of Parikh automata and related models.
In: Freund, R., Holzer, M., Mereghetti, C., Otto, F., Palano, B. (eds.) Proceedings of the 3rd Workshop on Non-Classical Models of Automata and Applications (NCMA'11). books@ocg.at, vol. 282, pp. 103–119

[1]: DIRO, Université de Montréal. The third author is supported by the Natural Sciences and Engineering Research Council of Canada.

[2]: LSV, ENS Cachan, CNRS, France. Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence "REACHARD-ANR-11-BS02-001".

Presentation

The starting point of this paper is a deepened study of Parikh Automata (PA), mainly through the use of Constrained Automata (CA). CA were suggested by Finkel as a model worth studying, as a follow-up to his work on related automaton models (see, e.g., [FIS03]). The idea of an automaton constraining the Parikh image of its paths was found in the work of Bouajjani and Habermehl [BH99], where they focused on *flat* automata (see Paper II for a study of those). After some initial results — closure properties, decidability properties, etc. — we realized that most of this preliminary work had been done within an equivalent model, that of Parikh Automata, introduced by Klaedtke and Rueß [KR03]. The results that did not appear previously form essentially Section 1 of this paper.

Further, following ideas of Finkel and Leroux [FL02], we investigated a generalization of those automata where a transition applies an affine transformation on a tuple of registers. This led to the definition of *Affine Parikh Automata* (APA). Our interest stemmed from a restriction of the APA — to be investigated in Papers II and III: finite-monoid APA. The study of the general APA model constitutes Section 2 of this paper.

Finally, we studied not only a generalization, but also a restriction of the models. Section 3 of this paper deals with *Letter Parikh Automata*; that is, PA for which if $(a, \mathbf{x})$ and $(a, \mathbf{y})$ are transition labels, then $\mathbf{x} = \mathbf{y}$.

Personal contribution. Apart from the initial idea of the study, I proposed the ideas and results of this paper in their entirety. A great effort has been made by Finkel and McKenzie to enforce that I write proofs with precision and completeness; in particular, McKenzie offered to write the lengthy proofs of Lemma 28 and Lemma 29 to indicate the level of clarity that is required in a scientific publication. The need for exhaustive proofs was also prompted by the fact that some earlier lightly-proved results turned out to be false. The structure of the paper and its final shape came as a concerted work of the three authors, with the specific emphases dictated by Finkel and McKenzie's experience.

Affine Parikh Automata

Abstract

The Parikh finite word automaton (PA) was introduced and studied in 2003 by Klaedtke and Rueß. Natural variants of the PA arise from viewing a PA equivalently as an automaton that keeps a count of its transitions and semilinearly constrains their numbers. Here we adopt this view and define the *affine PA*, that extends the PA by having each transition induce an affine transformation on the PA registers, and the *PA on letters*, that restricts the PA by forcing any two transitions on the same letter to affect the registers equally. Then we report on the expressiveness, closure, and decidability properties of such PA variants. We note that deterministic PA are strictly weaker than deterministic reversal-bounded counter machines.

Introduction

Klaedtke and Rueß [KR03] introduced the *Parikh automaton* as a pair (A, C) where C is a semilinear subset of $\mathbb{N}^d$ and A is a finite automaton over $(\Sigma \times D)$ for Σ a finite alphabet and D a finite subset of $\mathbb{N}^d$. The word $w_1 \cdots w_n \in \Sigma^*$ is accepted by (A, C) if A accepts some word $(w_1, \mathbf{v}_1) \cdots (w_n, \mathbf{v}_n)$ such that $\sum \mathbf{v}_i \in C$. Motivated by verification issues, Klaedtke and Rueß developed the PA as a tool to probe (weak) monadic second-order logic with successor in which the cardinality $|X|$ of each second-order variable X is available. They proved their logic undecidable but showed decidability of an existential fragment that was successfully applied to verify the specification of actual hardware circuits.

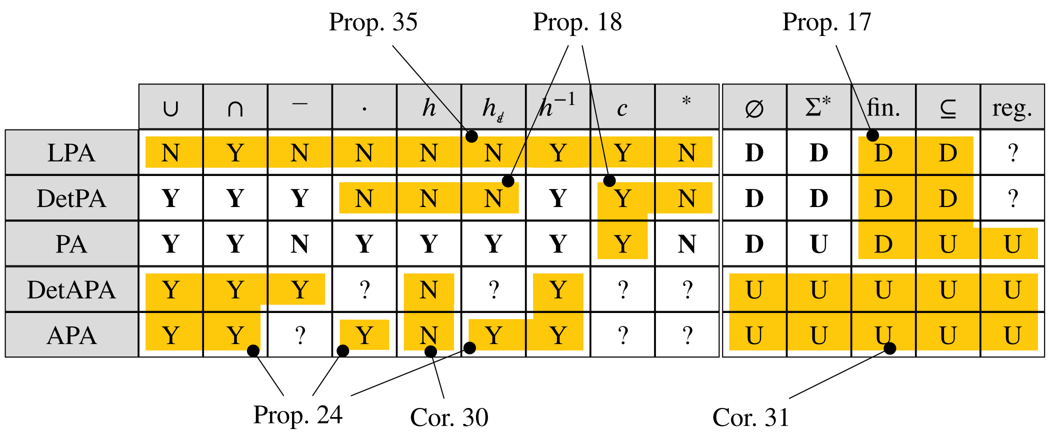
Klaedtke and Rueß also studied decidability properties of the PA and properties of the language classes defined by PA [KR03, KR02]. Karianto [Kar04] took up this study further, elaborating on Klaedtke and Rueß's proofs and considering pushdown automata and constraint sets beyond semilinear. As for tree languages, Klaedtke and Rueß [KR02] introduced Parikh Tree Automata as top-down tree automata with one global semilinear constraint; at the same time, the related notion of Presburger Tree Automata, which combines bottom-up tree automata and semilinear preconditions about the number of children in a given state, was independently introduced by Dal Zilio and Lugiez [DZL03] and Seidl, Schwentick, and Muscholl [SSM03].

Our interest in the PA comes both from its role in the area of verification and from the intricate three-way connection known to exist between automata, descriptive complexity and Boolean circuit complexity (see [Str94, TT07]). Indeed several circuit-based complexity classes within the class LogCFL (of languages reducible to a context-free language) can be described in a natural way using first-order logic. In such a logic description, the (generalized) quantifiers reflect the properties of the automaton-based model defining the language while the (numerical) predicates reflect the level of uniformity allowed to the circuit families accepting the language. Since semilinearity arises in the study of LogCFL (see [MTV10]) and since the circuit depth complexity of regular languages is a major open question in complexity theory, the PA is a very appealing computation model with which to experiment in view of possible future applications to complexity theory.

In this paper we introduce three models closely related to the PA and we carry the study of PA themselves somewhat further. This is our first contribution. Informally, each model involves a finite automaton A and a constraint set $C \subseteq \mathbb{K}^d$ where $\mathbb{K}$ is either $\mathbb{N}$ or $\mathbb{Q}$:

- *Constrained automata* (CA) with d transitions are defined to accept a word $w \in \Sigma^*$ iff C contains the d-tuple that records, for some accepting run of A on w and for each transition t, the number of occurrences of t along that accepting run; we will see that the CA merely provides an alternate view of the PA in that the two models define the same language classes.
- *Affine Parikh automata* (APA) generalize PA by allowing each transition to perform an affine transformation on the d-tuple of PA registers; an APA accepts a word w iff some accepting run of A on w maps the all-zero d-tuple to a d-tuple in C.
- *Parikh automata on letters* (LPA) restrict PA by imposing the condition that any transition on $(a, \mathbf{u}) \in (\Sigma \times D)$ and any transition on $(b, \mathbf{v}) \in (\Sigma \times D)$ must satisfy $\mathbf{u} = \mathbf{v}$ when $a = b$.

Our second contribution is the analysis of the closure and decidability properties of these models and their deterministic variants DetPA and DetAPA. We depict the known properties of PA [KR02, KR03, Kar04] together with our new results in Figure 1, where


Prop. 35
Prop. 18
Prop. 17
Prop. 24
Cor. 30
Cor. 31

	$\cup$	$\cap$	$-$	$\cdot$	h	$h_{\not\varepsilon}$	h^{-1}	c	$*$	$\varnothing$	Σ^*	fin.	$\subseteq$	reg.
LPA	N	Y	N	N	N	N	Y	Y	N	**D**	**D**	D	D	?
DetPA	**Y**	**Y**	**Y**	N	N	N	**Y**	Y	N	**D**	**D**	D	D	?
PA	**Y**	**Y**	**N**	**Y**	**Y**	**Y**	**Y**	Y	**N**	**D**	**U**	D	U	U
DetAPA	Y	Y	Y	?	N	?	Y	?	?	U	U	U	U	U
APA	Y	Y	?	Y	N	Y	Y	?	?	U	U	U	U	U

Figure 1: Closure in the effective sense (Y) or nonclosure (N) of language classes defined by PA variants, under set operations, concatenation, morphisms, nonerasing morphisms, inverse morphisms, commutation, and iteration; decidability (D) or undecidability (U) of emptiness, universality, finiteness, inclusion, and regularity; boldface denotes results known *prior* to this paper.

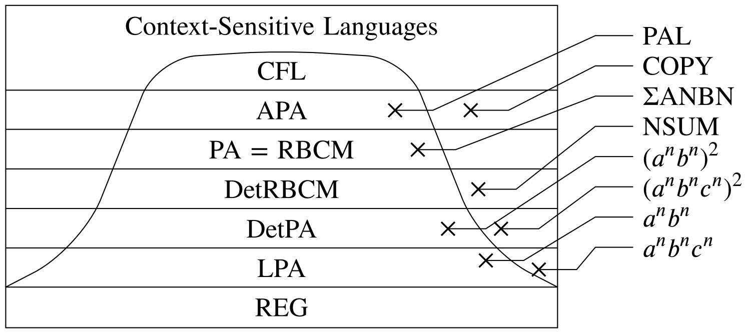


Figure 2: Relationships between language classes, sorted vertically by inclusion except for the class CFL of context-free languages delimited by the bell curve; RBCM stands for reversal-bounded counter machine; PAL is the language of pointed palindromes, COPY that of words $w\#w$, ΣANBN that of words wa^nb^n, and NSUM is discussed in Proposition 15.

DetLPA is not mentioned because DetLPA and LPA define the same languages.

Our third contribution is the comparison of the language classes that arise. We show that the language $\{a,b\}^* \cdot \{a^n\#a^n \mid n \in \mathbb{N}\}$ belongs to $\mathscr{L}_{\text{PA}} \setminus \mathscr{L}_{\text{DetPA}}$ where these two classes were only proved different in [KR03]. We show that APA and DetAPA over $\mathbb{Q}$ can be simulated by APA and DetAPA over $\mathbb{N}$ and vice versa. Refining [KR03] slightly, we compare our models with the reversal-bounded counter machines (RBCM) defined by Ibarra [Iba78]. Figure 2 summarizes these and further results.

This paper is organized as follows. Section 1 defines the PA, introduces the equivalent CA, justifies the PA line and DetPA line entries from Figure 1 and compares the PA with Ibarra's RBCM. Sections 2 and 3 investigate the APA and the LPA respectively, completing the proofs of all remaining entries in Figures 1 and 2. Section 4 concludes with a short discussion.

1 Parikh automata

We propose an alternative view of the PA which will prove very useful. We note that a PA can be viewed equivalently as an automaton that applies a semilinear constraint on the counts of the individual transitions occurring along its accepting runs.

PA, CA: see pp. 11, 14

▶ **Theorem 6.** *CA and PA define the same classes of languages. The same holds in the deterministic case.*

Proof. Let (A, C) be a PA (resp. DetPA) of dimension d, and let $\delta = \{t_1, \ldots, t_n\}$ be the transitions of A. We suppose moreover that A is deterministic — this does not imply the determinism of the PA. Consider the automaton A' which is a copy of A except that the vector part of the transitions is dropped, and note that if (A, C) is a DetPA then A' is deterministic. Suppose that the mapping induced between the transitions of A and A', i.e., $p \bullet\!\!-(a, \mathbf{v})\!\!\to q$ to $p \bullet\!\!-a\!\!\to q$, is a bijection. The contribution of a transition $t_i = p \bullet\!\!-(a, \mathbf{v}_i)\!\!\to q$ to the extended Parikh image of the label of a run in which it appears is $\mathbf{v}_i$; thus, knowing how many times t_i is taken in a path is enough to retrieve the value of the extended Parikh image of the label of a path. More precisely, for a path π in A and the equivalent path π' in A', if we let $\mathsf{Pkh}(\pi') = (x_1, \ldots, x_n)$ then the extended Parikh image of $\mu(\pi)$, $\Phi(\mu(\pi))$, is $\sum_{i=1}^{n} x_i \times \Phi(\mu(t_i))$. Thus, we define $C' \subseteq \mathbb{N}^n$ by $C' = \{(x_1, \ldots, x_n) \mid \sum_{i=1}^{n} x_i \times \Phi(\mu(t_i)) \in C\}$, and the PA (A, C) has the same language as the CA (A', C'), and determinism is preserved.

Now note that the aforementioned bijection exists if no two distinct transitions t_i, t_j are such that $t_i = p \bullet\!\!-(a, \mathbf{v}_i)\!\!\to q$ and $t_j = p \bullet\!\!-(a, \mathbf{v}_j)\!\!\to q$. So suppose that such t_i and t_j exist, we show how to remove them; iterating this process will lead to a PA with no such pair of transitions. First, we increment the dimension of the PA by adding a 0 component to all the vectors appearing as labels, i.e., each label $(\ell, \mathbf{v})$ is replaced by $(\ell, (\mathbf{v}, 0))$. Next, we remove t_i and t_j and add the transition $t = p \bullet\!\!-(a, \mathbf{e}_{d+1})\!\!\to q$ where $\mathbf{e}_{d+1} \in \{0, 1\}^{d+1}$ has a one only in position $d + 1$. Now note that when t is taken in the new automaton, either t_i or t_j could have been taken in the old one. Thus define the semilinear set D to *split* the number of times t is taken

— which is stored in the $d+1$-th component — between t_i and t_j; for $\mathbf{x} \in \mathbb{N}^d$, $c \in \mathbb{N}$:

$$(\mathbf{x}, c) \in D \Leftrightarrow (\exists c_i, c_j \in \mathbb{N}) \left[c = c_i + c_j \wedge \left(\mathbf{x} + c_i.\mathbf{v}_i + c_j.\mathbf{v}_j\right) \in C\right] \ .$$

This preserves the language of the PA and does not affect determinism.

For the reverse direction, let (A, C) be a CA (resp. DetCA). Define A' as the automaton A in which each transition $t = p \bullet\!\!- a \!\rightarrow q$ is replaced by the transition $p \bullet\!\!- (a, \mathsf{Pkh}(t)) \!\rightarrow q$. Now let π be a path in A and π' be the corresponding path in A', the construction is such that $\Phi(\mu(\pi')) = \mathsf{Pkh}(\pi)$, thus (A', C) is a PA with the same language as (A, C), and the determinism of A is preserved.

1.1 On the expressiveness of Parikh automata

The constrained automaton characterization of PA helps deriving pumping-style necessary conditions for membership in $\mathscr{L}_{\mathrm{PA}}$ and in $\mathscr{L}_{\mathrm{DetPA}}$:

▶ **Lemma 7.** *Let $L \in \mathscr{L}_{\mathrm{PA}}$. There exist $p, \ell \in \mathbb{N}^+$ such that any $w \in L$ with $|w| \geq \ell$ can be written as $w = uvxvz$ where:*
(1) $0 < |v| \leq p$, $|x| > p$, and $|uvxv| \leq \ell$;
(2) $uv^2xz \in L$ and $uxv^2z \in L$.

Proof. Let (A, C) be a CA of language L. Let p be the number of states in A and m be the number of elementary cycles (i.e., cycles in which no state except the start state occurs twice) in the underlying multigraph of A. Finally, let $\ell = p \times (2m + 1)$. Now, let $w \in L$ such that $|w| \geq \ell$ and $\pi \in \mathsf{Run}(A)\restriction_C$ such that $\mu(\pi) = w$. Write π as $\pi_1 \cdots \pi_{2m+1}\pi'$ where $|\pi_i| = p$. By the pigeonhole principle, each π_i contains an elementary cycle, and thus, there exist $1 \leq i, j \leq 2m + 1$ with $i + 1 < j$ such that π_i and π_j share the same elementary cycle η labeled with a word v. Thus π can be written as $\rho_1\eta\rho_2\eta\rho_3$, such that, with $u = \mu(\rho_1)$, $x = \mu(\rho_2)$, and $z = \mu(\rho_3)$, we have condition *(1)*. Moreover, $\rho_1\eta^2\rho_2\rho_3$ and $\rho_1\rho_2\eta^2\rho_3$ are two accepting paths the Parikh images of which are in C, thus their labels, uv^2xz and uxv^2z respectively, are in L, showing condition *(2)*.

A similar argument leads to a stronger property for the languages of $\mathscr{L}_{\mathrm{DetPA}}$:

▶ **Lemma 8.** *Let $L \in \mathscr{L}_{\mathrm{DetPA}}$. There exist $p, \ell \in \mathbb{N}^+$ such that any w over the alphabet of L with $|w| \geq \ell$ can be written as $w = uvxvz$ where:*
(1) $0 < |v| \leq p$, $|x| > p$, and $|uvxv| \leq \ell$;

(2) uv^2x, $uvxv$, and uxv^2 are equivalent under the Nerode relation of L.

Proof. Let (A, C) be a DetCA of language $L \subseteq \Sigma^*$. We may suppose that A is complete, as $\mathsf{Run}(A)$ is essentially unchanged when adding a sink state to A. Let p be the number of states in A and m be the number of elementary cycles (i.e., cycles in which no state except the start state occurs twice) in the underlying multigraph of A. Finally, let $\ell = p \times (2m + 1)$. Now, let $w \in \Sigma^{\geq \ell}$ and let π be the path traced by w in A, which exists as A is complete. Write π as $\pi_1 \cdots \pi_{2m+1}\pi'$ where $|\pi_i| = p$. By the pigeonhole principle, each π_i contains an elementary cycle, and thus, there exist $1 \leq i, j \leq 2m + 1$ with $i + 1 < j$ such that π_i and π_j share the same elementary cycle η labeled with a word v. Thus π can be written as $\rho_1\eta\rho_2\eta\rho_3$, such that, with $u = \mu(\rho_1)$, $x = \mu(\rho_2)$, and $z = \mu(\rho_3)$, we have condition *(1)*. Moreover, uv^2x, $uvxv$, and uxv^2 trace the paths $\rho_1\eta^2\rho_2$, $\rho_1\eta\rho_2\eta$, and $\rho_1\rho_2\eta^2$, respectively, in A. Those paths all go from the initial state to the same state q and have the same Parikh image. Thus let π'' be a path in A from q with some label y, then $uv^2xy \in L(A, C)$ iff π'' ends in a final state and $\mathsf{Pkh}(\rho_1\eta^2\rho_2\pi'') \in C$. But since $\mathsf{Pkh}(\rho_1\eta^2\rho_2\pi'') = \mathsf{Pkh}(\rho_1\eta\rho_2\eta\pi'') = \mathsf{Pkh}(\rho_1\rho_2\eta^2\pi'')$, this is the case iff $uvxvy \in L$ and iff $uxv^2y \in L$, showing condition *(2)*.

We apply Lemma 7 to the language COPY, defined as $\{w\#w \mid w \in \{a, b\}^*\}$, as follows:

▶ **Proposition 9.** COPY $\notin \mathscr{L}_{\text{PA}}$.

Proof. Suppose COPY $\in \mathscr{L}_{\text{PA}}$. Let ℓ, p be given by Lemma 7, and consider $w = (a^p b)^\ell \#(a^p b)^\ell \in$ COPY. Lemma 7 states that $w = uvxvz$ where $uvxv$ lays in the first half of w, and $s = uv^2xz \in$ COPY. Note that x contains at least one b. Suppose $v = a^i$ for $1 \leq i \leq p$, then there is a sequence of a's in the first half of s unmatched in the second half. Likewise, if v contains a b, then s has a sequence of a's between two b's unmatched in the second half. Thus $s \notin$ COPY, a contradiction. Hence COPY $\notin \mathscr{L}_{\text{PA}}$.

As Klaedtke and Rueß show using closure properties, DetPA are strictly weaker than PA. The thinner grain of Lemma 8 suggests explicit languages that witness the separation of $\mathscr{L}_{\text{DetPA}}$ from $\mathscr{L}_{\text{PA}}$. Indeed, let EQUAL $\subseteq \{a, b, \#\}^*$ be the language $\{a, b\}^* \cdot \{a^n\#a^n \mid n \in \mathbb{N}\}$, we have:

▶ **Proposition 10.** EQUAL $\in \mathscr{L}_{\text{PA}} \setminus \mathscr{L}_{\text{DetPA}}$.

Proof. We omit the proof that EQUAL $\in \mathscr{L}_{\text{PA}}$. Now, suppose EQUAL $\in \mathscr{L}_{\text{DetPA}}$, and let ℓ, p be given by Lemma 8. Consider $w = (a^p b)^\ell$. Lemma 8 then asserts that a prefix of w can be written as $w_1 = uvxv$, and that $w_2 = uv^2x$ verifies $w_1 \equiv^{\text{N}} w_2$. As $|x| > p$, x contains a b. Let k be the number of a's at the end of w_1. Suppose $v = a^i$ for $1 \le i \le p$, then w_2 ends with $k - i < k$ letters a. Thus $w_1 \# a^k \in$ EQUAL and $w_2 \# a^k \notin$ EQUAL, a contradiction. Suppose then that $v = a^i b a^k$, with $0 \le i + k < p$. Then w_2 ends with $p - i > k$ letters a, and similarly, $w_1 \not\equiv^{\text{N}} w_2$, a contradiction. Thus EQUAL $\notin \mathscr{L}_{\text{DetPA}}$.

For comparison, we mention another line of attack for the study of $\mathscr{L}_{\text{DetPA}}$, derived from an argument used by Klaedtke and Rueß to show that PAL $= \{w\#w^{\text{R}} \mid w \in \{a, b\}^+\}$, where w^{R} is the reversal of w, is not in $\mathscr{L}_{\text{PA}}$.

See Chap. 0, Prop. 2, p. 13

▶ **Lemma 11.** *Let $L \in \mathscr{L}_{\text{DetPA}}$. There exists $c > 0$ such that $|\{[w]^N_L \mid w \in \Sigma^n\}| \in O(n^c)$.*

$[w]^N$: equiv. class under Nerode relation $\equiv^N$, see p. 7

Proof. Let (A, C) be a DetCA of language $L \subseteq \Sigma^*$ where we suppose A complete, as this leaves $\mathsf{Run}(A)$ essentially unchanged. For $w \in \Sigma^*$, write $\pi(w)$ for the unique path in A labeled w and starting with the initial state. Let $\sim$ be the equivalence relation on Σ^* defined by $u \sim v$ iff $\mathsf{Pkh}(\pi(u)) = \mathsf{Pkh}(\pi(v)) \wedge \mathsf{To}(\pi(u)) = \mathsf{To}(\pi(v))$. Then this relation refines $\equiv^{\text{N}}_L$: let $u, v \in \Sigma^*$ such that $u \sim v$, and let $w \in \Sigma^*$ such that $uw \in L$, then $\pi(uw) \in \mathsf{Run}(A)\restriction_C$, thus the same holds for $\pi(vw)$, implying that $vw \in L$. Moreover, the number of equivalence classes of $\sim$ for a given word length is polynomial in the word length (e.g., [Maz10, p. 41]).

▶ **Proposition 12.** *Let $L = \{w \in \{a, b\}^* \mid w_{|w|_a} = b\}$, where w_i is the i-th letter of w. Then $L \in \mathscr{L}_{\text{PA}} \setminus \mathscr{L}_{\text{DetPA}}$.*

Proof. We omit the proof that $L \in \mathscr{L}_{\text{PA}}$; the main point is simply to guess the position of the b referenced by $|w|_a$. On the other hand, let $n > 0$ and $u, v \in \{a, b\}^n$ such that $|u|_a = |v|_a = \frac{n}{2}$ and there exists $p \in \{\frac{n}{2}, \dots, n\}$ with $u_p \neq v_p$. Let $w = a^{p - \frac{n}{2}}$, then $(uw)_{|uw|_a} = (uw)_{|u|_a + |w|_a} = (uw)_p = u_p$, and similarly, $(vw)_{|vw|_a} = v_p$. This implies $uw \notin L \leftrightarrow vw \in L$, thus $u \not\equiv^{\text{N}} v$. Then for $0 \le i \le \frac{n}{2}$, define $E_i = \{a^{\frac{n}{2} - i} b^i z \mid z \in \{a, b\}^{\frac{n}{2}} \wedge |z|_a = i\}$. For any $u, v \in \bigcup E_i$ with $u \neq v$, the previous discussion shows that $u \not\equiv^{\text{N}} v$. Thus $|\{[w]^{\text{N}}_L \mid w \in \{a, b\}^n\}| \ge |\bigcup_{i=0}^{\frac{n}{2}} E_i| = \sum_{i=0}^{\frac{n}{2}} |E_i| = \sum_{i=0}^{\frac{n}{2}} \binom{\frac{n}{2}}{i} = 2^{\frac{n}{2}} \notin O(n^{O(1)})$. Lemma 11 then implies that $L \notin \mathscr{L}_{\text{DetPA}}$.

Finally, let us recall that a language $L \subseteq \Sigma^*$ is said to be *bounded* if there exist $n > 0$ and $w_1, \dots, w_n \in \Sigma^+$ such that $L \subseteq w_1^* \cdots w_n^*$. For a given class of languages, we say that it is *Parikh-bounded* if for any L in the class there exists a bounded language L' in the class with $L' \subseteq L$ and $\mathsf{Pkh}(L) = \mathsf{Pkh}(L')$. This property is known to hold for regular [Lat78] and context-free languages [BL81] (the latter recently reworked in [GMM10]).

See also Ex. 8, p. 8

▶ **Proposition 13.** *$\mathscr{L}_{\text{PA}}$ is Parikh-bounded.*

Proof. Let (A, C) be a constrained automaton, where δ is the transition set of A. Note that $\mathsf{Run}(A)$ is regular, thus, as mentioned, we can find a bounded regular language $R \subseteq \mathsf{Run}(A)$ such that $\mathsf{Pkh}(R) = \mathsf{Pkh}(\mathsf{Run}(A))$. In particular, $\mathsf{Pkh}(R{\restriction_C}) = \mathsf{Pkh}(\mathsf{Run}(A){\restriction_C})$. Closure under morphism of $\mathscr{L}_{\text{PA}}$ and of bounded languages implies that $L' = \mu(R{\restriction_C})$ is a bounded language of $\mathscr{L}_{\text{PA}}$ included in $L(A, C)$. Moreover, $\mathsf{Pkh}(L(A, C)) = \mathsf{Pkh}(\mu(R{\restriction_C}))$, and thus, equals $\mathsf{Pkh}(L')$.

1.2 Parikh automata and reversal-bounded counter machines

RBCM: see Chap. 0, Sec. 3.4, p. 16

Klaedtke and Rueß noticed in [KR02] that Parikh automata recognize the same languages as reversal-bounded counter machines, a model introduced and studied in the seventies by Ibarra [Iba78].

In [KR02, Section A.3], it is shown that PA have the same expressive power as (nondeterministic) RBCM. Although Fact 30 of [KR02], on which the authors rely to prove that $\mathscr{L}_{\text{RBCM}} \subseteq \mathscr{L}_{\text{PA}}$, is technically false as stated,[2] the small gap there can be fixed so that:

▶ **Proposition 14** ([KR02]). $\mathscr{L}_{\text{PA}} = \mathscr{L}_{\text{RBCM}}$.

Proof. We sketch $\mathscr{L}_{\text{RBCM}} \subseteq \mathscr{L}_{\text{PA}}$ for completeness and reprove $\mathscr{L}_{\text{PA}} \subseteq \mathscr{L}_{\text{RBCM}}$ to extract a more precise structure on the constructed RBCM — this will prove useful when comparing the notion of determinism in both models.

($\mathscr{L}_{\text{RBCM}} \subseteq \mathscr{L}_{\text{PA}}$.) First, it is known [Iba78] that any RBCM language can be expressed as an RBCM which makes *at most* one change between increment and

[2]Fact 30 of [KR02] states the following. Consider a RBCM M which, for any counter, changes between increment and decrement only once. Let M' be M in which negative counter values are allowed and the zero-tests are ignored. Then a word is claimed to be accepted by M iff the run of M' on the same word reaches a final state with all its counters nonnegative. A counter-example is the following. Take A to be the minimal automaton for a^*b, and add a counter for the number of a's that blocks the transition labeled b unless the counter is nonzero. This machine recognizes a^+b. Then by removing this test, the machine now accepts b.

decrement on each of its counters. Then a counter can be seen as being in one of three different states: (1) never incremented, (2) incremented but never decremented, (3) decremented. When a counter is in state (1), we may simulate the behavior of the RBCM with the counter set to zero. Similarly, when a counter is in state (2), we may simulate the behavior of the RBCM with this counter set to a nonzero value. Lastly, when in state (3), we may *guess* at some point that the counter reached zero, and act for the rest of the execution as if the counter is actually zero (thus not making any modification to this counter). Now, when in states (1) and (2), the behavior of the RBCM w.r.t. the counter can be simulated using a finite automaton; in state (3), the guess can be taken with a finite automaton, but we must check that the guess was taken at the right moment. Thus we use a PA to count the number of increments and decrements, and we check at the end of the computation that the latter is no greater than the former, and that these are equal iff the counter has been guessed to be zero at some point. Finally, the transitions between the different states can be made knowing only the transitions of the RBCM. This gives the required PA for the RBCM.

($\mathscr{L}_{\mathrm{PA}} \subseteq \mathscr{L}_{\mathrm{RBCM}}$.) Let (A, C) be a CA, where $A = (Q, \Sigma, \delta, q_0, F)$ and let $\delta = \{t_1, \ldots, t_k\}$. We define a RBCM of the same language in two steps. (1). First, let M be the k-counter machine $(Q \cup \{q_f\}, \Sigma, \zeta, q_0, q_f)$, where $q_f \notin Q$ and ζ is defined by:

$$\zeta = \bigcup_{\substack{\mathbf{x} \in \{0,1\}^k \\ 1 \le i \le k}} \{(\mathsf{From}(t_i), \mu(t_i), \mathbf{x}, \mathsf{To}(t_i), \mathrm{R}, \mathbf{e}_i)\} \cup \bigcup_{\substack{\mathbf{x} \in \{0,1\}^k \\ q \in F}} \{(q, \lhd, \mathbf{x}, q_f, \mathrm{S}, \mathbf{0})\} \enspace .$$

This machine does not make any test, and accepts (in q_f) precisely the words accepted by A. Moreover, the state of the counters in q_f is the Parikh image of the path taken (in A) to recognize the input word. (2). We then refine M to check that the counter values belong to C. We note that we can do that as a direct consequence of the proof of [IS99, Theorem 3.5], but this proof relied on nontrivial algebraic properties of systems $M\mathbf{y} = \mathbf{b}$, where M is a matrix, $\mathbf{y}$ are unknowns, and $\mathbf{b}$ is a vector; we present here a proof based on a logical characterization of semilinear sets. Recall that C can be expressed as a quantifier-free first-order formula which uses the function symbol $+$, the congruence relations $\equiv_i$, for $i \geq 2$, and the order relation $<$ (see, e.g., [End72]). So let C be given as such formula ϕ_C with k free variables. Let ϕ_C be put in disjunctive normal form. The machine M then tries each and every clause of ϕ_C for acceptance. First, note that a term can be computed deterministically with a number of counters and reversals which depends only on its size. For instance, computing $c_i + c_j$ requires two new counters x, y: c_i is decremented until it reaches 0, while x and y are incremented, so that their value is c_i, now y is decremented until it reaches 0 while c_i is incremented back to its original value, finally the same process is applied with c_j, and as a result x is now $c_i + c_j$. Second, note that any atomic formula ($t_1 < t_2$ or $t_1 \equiv_i t_2$) can be checked by a DetRBCM: for $t_1 < t_2$, compute $x_1 = t_1$ and $x_2 = t_2$, then decrement x_1 and x_2 until one of them reaches 0, if the first one is x_1, then the atomic formula is true, and false otherwise; for $t_1 \equiv_i t_2$, a simple automaton-based construction depending on i can decide if the atomic formula is true. Thus, a DetRBCM can decide, for each clause, if all of its atomic formulas (or negation) are true, and in this case, accept the word. This process does not use the read-only head, and uses a number of counters and a number of reversals that depend only on the length of ϕ_C.

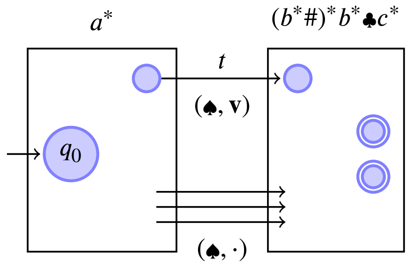
Further, we study how the notion of determinism compares in the two models. Let NSUM $= \{a^n \spadesuit b^{m_1} \# b^{m_2} \# \cdots \# b^{m_k} \clubsuit c^{m_1+\cdots+m_n} \mid k \geq n \geq 0 \wedge m_i \in \mathbb{N}\}$: the number of a's is the number of m_i's to sum to get the number of c's. Note that NSUM is not context-free. Then:

Note
In Paper III, Prop. 67, a similar result is proved with a simpler language.

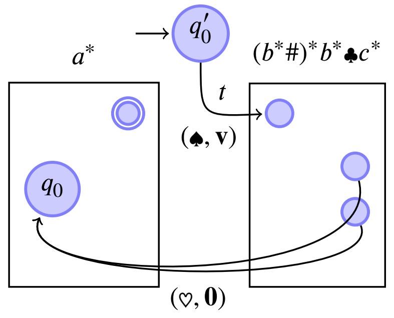
▶ **Proposition 15.** $\mathscr{L}_{\text{DetPA}} \subsetneq \mathscr{L}_{\text{DetRBCM}}$ *and* NSUM $\in \mathscr{L}_{\text{DetRBCM}} \setminus \mathscr{L}_{\text{DetPA}}$.

Proof. The inclusion $\mathscr{L}_{\text{DetPA}} \subseteq \mathscr{L}_{\text{DetRBCM}}$ follows from the proof of Proposition 14, as the resulting RBCM is deterministic if the given CA is.

We now show that NSUM $\in \mathscr{L}_{\text{DetRBCM}} \setminus \mathscr{L}_{\text{DetPA}}$. We omit the fact that NSUM $\in \mathscr{L}_{\text{DetRBCM}}$. Now suppose (A, C) is a DetPA such that $L(A, C) =$ NSUM, with $A = (Q, \Sigma \times D, \delta, q_0, F)$. As (A, C) is a DetPA, A is deterministic — it is indeed already deterministic with respect to the first component of the labels. We may suppose that the projection on Σ of $L(A)$ is a subset of $a^* \spadesuit (b^* \#)^* b^* \clubsuit c^*$, so that there exist $k \geq 0$, $q_1, \ldots, q_k \in Q$, and $j \in \{0, \ldots, k\}$ such that $q_i \bullet\!\!-(a, \mathbf{v}_i)\!\!\to q_{i+1} \in \delta$, for $0 \leq i < k$ and some $\mathbf{v}_i$'s, and $q_k \bullet\!\!-(a, \mathbf{v}_k)\!\!\to q_j \in \delta$. Moreover, we may suppose that no other transition points to one of the q_i's, and that all transitions $t = q_i \bullet\!\!-(\ell, \mathbf{v})\!\!\to q \in \delta$ such that $q \notin \{q_0, \ldots, q_k\}$ are with $\ell = \spadesuit$; let T be the set of all such transitions t. Graphically, A looks like:



We define $|T|$ DetPA such that the union of their languages forms the language SUMN $= \{\spadesuit w \heartsuit a^n \mid a^n \spadesuit w \in \text{NSUM}\}$, the strings of NSUM with a^n pushed at the end. For $t \in T$, define A_t as the automaton similar to A but which starts with the transition t and delays the first part of the computation until the end; graphically:


q_0'
a^*
$(b^*\#)^*b^*\clubsuit c^*$
t
$(\spadesuit, \mathbf{v})$
q_0
$(\heartsuit, \mathbf{0})$

Formally, $A_t = (Q \cup \{q_0'\}, \Sigma \times D, \delta_t, q_0', \{\mathsf{From}(t)\})$ where q_0' is a fresh (i.e., new) state and:

$$\delta_t = (\delta \setminus T) \cup \{q_0' \bullet\!\!- \mu(t) \!\to \mathsf{To}(t)\} \cup \{q_f \bullet\!\!- (\heartsuit, \mathbf{0}) \!\to q_0 \mid q_f \in F\} \ .$$

Now for $\omega \in L(A)$, let t be the transition labeled $\spadesuit$ taken when A reads ω, and let $\omega = \omega_1 \mu(t) \omega_2$. Then $\mu(t)\omega_2(\heartsuit, \mathbf{0})\omega_1 \in L(A_t)$, and this word has the same extended Parikh image as ω. Thus we have that $\bigcup_{t \in T} L(A_t, C) =$ SUMN, and if NSUM $\in \mathscr{L}_{\text{DetPA}}$, then SUMN $\in \mathscr{L}_{\text{DetPA}}$, as $\mathscr{L}_{\text{DetPA}}$ is closed under union (see Figure 1). We now show that SUMN $\notin \mathscr{L}_{\text{DetPA}}$, thus leading to a contradiction showing the result. Suppose SUMN $\in \mathscr{L}_{\text{DetPA}}$ and let ℓ, p be given by Lemma 8 for SUMN. Consider $w = \spadesuit(b^p\#)^\ell$. Lemma 8 then asserts that a prefix of w can be written as $w_1 = uvxv$, and that $w_2 = uv^2x$ verifies $w_1 \equiv^{\mathrm{N}} w_2$. As $|x| > p$ and v is nonempty, x contains a #; moreover, v does not contain $\spadesuit$. Let $s = |w_1|_\# = |w_2|_\#$ and let n_i be the number of b's before the position of the s-th # in w_i, $i = 1, 2$. Suppose $v \in b^+$, then $n_1 < n_2$, thus $w_1 \clubsuit c^{n_1} \heartsuit a^s \in$ SUMN and $w_2 \clubsuit c^{n_1} \heartsuit a^s \notin$ SUMN, a contradiction. Suppose then that $v = b^i\#b^j$, with $0 \le i + j < p$. Similarly, as $i + j < p$, $n_2 < n_1$, and again, $w_1 \not\equiv^{\mathrm{N}} w_2$, a contradiction. Thus SUMN $\notin \mathscr{L}_{\mathrm{DetPA}}$.

The parallel drawn between (Det)PA and (Det)RBCM allows transferring some RBCM and DetRBCM results to PA and DetPA. An example is a consequence of the following lemma proved in 2011 by Chiniforooshan *et al.* [CDI⁺11] for the purpose of showing incomparability results between different models of reversal-bounded counter machines:

▶ **Lemma 16** ([CDI⁺11]). *Let a DetRBCM express $L \subseteq \Sigma^*$. Then there exists $w \in \Sigma^*$ such that $L \cap w\Sigma^*$ is a nontrivial regular language.*

Using this lemma, variants of the language EQUAL from Proposition 10 can be shown outside $\mathscr{L}_{\mathrm{DetPA}}$. For instance, for $\Sigma = \{a, b\}$, $\Sigma\mathrm{ANBN} = \Sigma^* \cdot \{a^n b^n \mid n \in \mathbb{N}^+\}$ is such that any $w \in \Sigma^*$ makes $\Sigma\mathrm{ANBN} \cap w\Sigma^*$ nonregular. Although Lemma 16 thus gives languages in $\mathscr{L}_{\mathrm{PA}} \setminus \mathscr{L}_{\mathrm{DetPA}}$, Lemma 16 seemingly does not apply to EQUAL itself since $\mathrm{EQUAL} \cap \#\{a, b, \#\}^* = \{\#\}$ is regular.

1.3 On decidability and closure properties of Parikh automata

In this section we justify the PA and DetPA line entries on Figure 1. The known decidability results depicted there (in boldface) are from [KR03] and [Iba78], and Karianto [Kar04] provided detailed proofs.

▶ **Proposition 17.** *(1) Finiteness is decidable for PA; (2) Inclusion is decidable for DetPA and undecidable for PA; (3) Regularity is undecidable for PA.*

Proof. (1-2). These decidability properties follow directly from the same properties for RBCM and DetRBCM [ISD⁺02], the effective equivalence between PA and RBCM (Proposition 14), and the effective inclusion of $\mathscr{L}_{\mathrm{DetPA}}$ in $\mathscr{L}_{\mathrm{DetRBCM}}$ (Proposition 15).

(3). This follows from a theorem of [Gre68], which states the following. Let $\mathscr{C}$ be a class of languages effectively closed under union and under concatenation with regular languages and for which $L = \Sigma^*$ is undecidable. Let P be a predicate on languages true of every regular language, false of some languages, preserved by inverse rational transduction, union with $\{\varepsilon\}$ and intersection with regular languages. Then P is undecidable in $\mathscr{C}$. Obviously, $\mathscr{L}_{\mathrm{PA}}$ satisfies the hypothesis for $\mathscr{C}$. More-

over, “being regular in $\mathscr{L}_{\text{PA}}$” is a predicate satisfying the hypothesis for P. Thus, regularity is undecidable for PA.

We now turn to closure properties:

▶ **Proposition 18.** *(1) $\mathscr{L}_{\text{DetPA}}$ is not closed under concatenation; (2) $\mathscr{L}_{\text{DetPA}}$ is not closed under nonerasing morphisms; (3) Both $\mathscr{L}_{\text{PA}}$ and $\mathscr{L}_{\text{DetPA}}$ are closed under commutative closure; (4) Neither $\mathscr{L}_{\text{PA}}$ nor $\mathscr{L}_{\text{DetPA}}$ is closed under starring.*

Proof. (1). The language EQUAL separating $\mathscr{L}_{\text{DetPA}}$ from $\mathscr{L}_{\text{PA}}$ is the concatenation of a regular language and a language of $\mathscr{L}_{\text{DetPA}}$, implying the nonclosure under concatenation.

(2). We note that any language of $\mathscr{L}_{\text{PA}}$ is the image by a nonerasing morphism of a language in $\mathscr{L}_{\text{DetPA}}$. Indeed, say (A, C) is a CA and let B be the deterministic automaton of language $\mathsf{Run}(A)$ defined as a copy of A in which the transition t is relabeled t (i.e., $p \bullet\!\!- a \!\rightarrow q$ becomes $p \bullet\!\!- (p, a, q) \!\rightarrow q$). Then (B, C) is a DetCA such that $L(A, C) = \mu_A(L(B, C))$. This implies the nonclosure of $\mathscr{L}_{\text{DetPA}}$ under nonerasing morphisms.

(3). Let $\Sigma = \{a_1, \ldots, a_n\}$, $L \subseteq \Sigma^*$ a semilinear language, and $C = \mathsf{Pkh}(L)$. Define A to be an automaton with one state, initial and final, with n loops, the i-th labeled $(a_i, \mathbf{e}_i) \in \Sigma \times \{\mathbf{e}_i\}_{1 \leq i \leq n}$. Then $c(L) = L(A, C)$. This implies that both $\mathscr{L}_{\text{PA}}$ and $\mathscr{L}_{\text{DetPA}}$ are closed under commutative closure, as both are classes of semilinear languages [KR03].

(4). We show that the starring of $L = \{a^n b^n \mid n \in \mathbb{N}\}$ is not in $\mathscr{L}_{\text{PA}}$. Suppose $L^* \in \mathscr{L}_{\text{PA}}$, and let $w = (a^p b^p)^\ell$, where ℓ, p are given by Lemma 7. The same lemma asserts that $w = uvxvz$, such that, in particular, uv^2xz and uxv^2z are in L^*. Now suppose $v = a^i$ for some $i \leq p$. Then uv^2x contains $a^{p+i}b^p$ with no more b’s on the right. Thus $uv^2xz \notin L^*$. The case for $v = b^i$ is similar. Now suppose $v = a^i b^j$ with $i, j > 0$. Then uv^2x contains $\cdots a^p b^j a^i b^p \cdots$, but $i < p$, thus $uv^2xz \notin L^*$. The case $v = b^i a^j$ is similar. Thus $L^* \notin \mathscr{L}_{\text{PA}}$.

Remark. Baker and Book [BB74] already note, in different terms, that if $\mathscr{L}_{\text{PA}}$ were closed under starring, it would be an intersection closed full AFL containing the language $\{a^n b^n \mid n \geq 0\}$, and so would be equal to the class of Turing-recognizable languages. Thus $\mathscr{L}_{\text{PA}}$ is not closed under starring.

2 Affine Parikh automata

A PA of dimension d can be viewed as an automaton in which each transition updates a vector $\mathbf{x}$ of $\mathbb{N}^d$ using a function $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}$ where $\mathbf{v}$ depends only on the transition. At the end of an accepting computation, the word is accepted if $\mathbf{x}$ belongs to some semilinear set. We propose to generalize the updating function to an affine function. We show that defining the model over $\mathbb{N}$ is as general as defining it over $\mathbb{Q}$. We study the expressiveness of this model and show it is strictly more powerful than PA. We then study its (non)closure properties and related decidability problems, leading to the observation that the model lacks some desirable properties — e.g., properties usually needed for any real-world application.

$\mathbb{K}$-APA: see p. 15

Remark. It is easily seen that $\mathbb{N}$-APA (resp. $\mathbb{N}$-DetAPA) are a generalization of CA (resp. DetCA). Indeed, let (A, C) be a CA and define, for $t \in \delta$, $U_t = (Id, \mathsf{Pkh}(t))$ where Id is the identity matrix of dimension $|\delta| \times |\delta|$. Then $L(A, C) = L(A, U, C)$. We will later see that this containment is strict.

We present a normal form for APA that is similar to a normal form given for PA by Karianto [Kar04]:

▶ **Lemma 19.** *Every $\mathbb{K}$-APA (A, U, C) of dimension d has the same language as a $\mathbb{K}$-APA (A', U', C') of dimension $d + 1$ with the three following properties:*

(i) The initial state of A' has no incoming transition;

(ii) The automaton A' is complete;

(iii) Every state of A' is final.

The same holds for $\mathbb{K}$-DetAPA.

Proof. Let (A, U, C) be a $\mathbb{K}$-APA of dimension d, that is, $A = (Q, \Sigma, \delta, q_0, F)$, $U : \delta^* \to \mathscr{F}_d^{\mathbb{K}}$, and $C \subseteq \mathbb{K}^d$. We ensure incrementally the three properties; that is, we assume for each property that the previous ones hold.

Ensuring (i). We define (A', U', C') as follows: $A' = (Q', \Sigma', \delta', q_0', F')$, where $Q' = Q \cup \{q_{\text{fresh}}\}$, with q_{fresh} a fresh state; $\Sigma' = \Sigma$; $\delta' = \delta \cup \delta_{\text{fresh}}$ with $\delta_{\text{fresh}} = \{q_{\text{fresh}} \bullet\!\!-a\!\!\to q \mid q_0 \bullet\!\!-a\!\!\to q \in \delta\}$; $q_0' = q_{\text{fresh}}$; if $q_0 \in F$, then $F' = F \cup \{q_{\text{fresh}}\}$ and otherwise $F' = F$. Note that A' is deterministic if A is, and that $L(A) = L(A')$. We define $U' : \delta' \to \mathscr{F}_d^{\mathbb{K}}$ as follows. For $q_{\text{fresh}} \bullet\!\!-a\!\!\to q \in \delta_{\text{fresh}}$, $U'_{q_{\text{fresh}} \bullet-a\to q} = U_{q_0 \bullet-a\to q}$ and for $t \in \delta$, we have $U'_t = U_t$. Finally, we let $C' = C$. Then $L(A, U, C) = L(A', U', C')$ and (A', U', C') verifies (i).

Ensuring (ii). Now suppose (A, U, C) verifies (i). Let A' be the automaton A in which an additional nonfinal sink state q_{sink} is added — that is, if a state $q \in Q$ has no outgoing transition labeled $a \in \Sigma$, the transition $q \bullet\!\!-a\!\!\rightarrow q_{\text{sink}}$ is added to A', and q_{sink} has $|\Sigma|$ self-loops, labeled by each letter of Σ. The new transitions of A' are associated with some function (for instance, the identity or the zero function); the constraint set C is left unchanged. This leaves both the language and the determinism the APA unchanged, and A' now verifies (i) and (ii).

Ensuring (iii). Now suppose (A, U, C) verifies (i) and (ii). We define A' as $(Q, \Sigma, \delta, q_0, Q)$, i.e., the automaton A with all states final. Let us define the $\mathbb{K}$-APA (A', U', C') of dimension $d+1$ where we fix $U' : \delta \to \mathcal{F}^{\mathbb{K}}_{d+1}$ such that the last component of the affine functions serves as a *flag*: it is set to 1 if the last state reached is in F, and 2 otherwise — this component takes the value 0 only for the empty path. Formally, for $t \in \delta$, $\mathbf{x} \in \mathbb{K}^d$, and $f \in \mathbb{K}$:

$$U'_t(\mathbf{x}, f) = \left(U_t(\mathbf{x}), \begin{cases} 1 & \text{if } \mathsf{To}(t) \in F, \\ 2 & \text{otherwise.} \end{cases} \right) .$$

Let us remark that the ability of APA to use constant functions (and not only translations, as in PA) allows to simplify the construction given by Karianto [Kar04]. Finally, if $\varepsilon \in L(A, U, C)$, we let $C' = C \times \{1\} \cup \{0^{d+1}\}$, and otherwise, we let $C' = C \times \{1\}$. We argue that $L(A, U, C) = L(A', U', C')$. For the empty word, the construction is such that $\varepsilon \in L(A, U, C) \to \varepsilon \in L(A', U', C')$. Now if $\varepsilon \notin L(A, U, C)$ then $U'_\varepsilon(\mathbf{0}) = \mathbf{0} \notin C'$, thus $\varepsilon \notin L(A', U', C')$. Now let w be a nonempty word in $L(A, U, C)$ and let π be an accepting path in A such that $\mu(\pi) = w$ and $U_\pi(\mathbf{0}) \in C$. Then π is also an accepting path in A', and as $\mathsf{To}(\pi) \in F$, we have that $U'_\pi(\mathbf{0}) = (U_\pi(0^d), 1)$, and as $U_\pi(\mathbf{0}) \in C$, we have that $U'_\pi(\mathbf{0}) \in C \times \{1\}$. Hence $\mu(\pi) = w$ is in $L(A', U', C')$. Conversely, suppose w is a nonempty word in $L(A', U', C')$ and let π be an accepting path in A' such that $\mu(\pi) = w$ and $U'_\pi(\mathbf{0}) \in C'$. Then π is a path in A, and as $U'_\pi(\mathbf{0}) = (U_\pi(0^d), 1)$, we have that $\mathsf{To}(\pi) \in F$ and $U_\pi(\mathbf{0}) \in C$, thus $\mu(\pi) = w$ is in $L(A, U, C)$.

We may verify that (A', U', C') satisfies the three properties and that the language $L(A', U', C')$ is equal to the language $L(A, U, C)$. Moreover, A' is deterministic if A is.

2.1 Affine Parikh automata on $\mathbb{Q}$ and $\mathbb{N}$

In this section, we show that the expressive power of affine Parikh automata is independent from the choice of $\mathbb{K}$. We first show that the constraint set can have a similar form in the two cases. We call *basic formula* a quantifier-free formula which uses the function symbols $+$ for addition and λ_n, $n \in \mathbb{N}$, for scalar multiplication, together with the relation symbol $<$ and constants from $\mathbb{N}$ — equality is expressible, as $t_1 = t_2$ is equivalent to $\neg(t_1 < t_2) \wedge \neg(t_2 < t_1)$. Of course, the scalar multiplication $\lambda_n(t)$ can be replaced by $t + \cdots + t$ where t appears n times, but its inclusion simplifies the proofs slightly. We remark, for future reference, the following property of basic formulas. For $\mathbf{v}$ a vector of natural numbers and ϕ a basic formula, the fact that ϕ is true of $\mathbf{v}$ is independent of the underlying model, whether it is $\mathbb{Q}$ or $\mathbb{N}$. In symbols, $\mathbb{Q} \models \phi(\mathbf{v})$ iff $\mathbb{N} \models \phi(\mathbf{v})$.

The following lemma shows in particular that the constraint set of $\mathbb{Q}$-APA can be expressed as a basic formula:

▶ **Lemma 20.** *Every $\mathbb{Q}$-definable set can be expressed as a basic formula.*

Quantif. elim. of $\mathbb{Q}$-definable sets, see p. 4

Proof. Recall that a $\mathbb{Q}$-definable set can be expressed with a quantifier-free formula ϕ. Thus, we need only get rid of the c's not in $\mathbb{N}$ appearing either as λ_c or as a constant in ϕ. First, note that we can suppose that if $\lambda_c(t)$ appears in ϕ, with t a term, then t is some variable: we simply apply the distributivity of λ_c (i.e., replace $\lambda_c(t_1+t_2)$ by $\lambda_c(t_1)+\lambda_c(t_2)$ and $\lambda_c(\lambda_{c'}(t))$ by $\lambda_{c\times c'}(t)$), then replace $\lambda_c(c')$ with c' a constant by the constant $c \times c'$, neither of those operations changing the set defined. Second, we take care of the negative c's. For any atomic formula $t_1 < t_2$ appearing in ϕ, if the constant $c < 0$ appears in t_1, we remove it from t_1 and add $-c$ to t_2; if $c < 0$ appears as $\lambda_c(x)$ in t_1, with x a variable, we remove it from t_1 and add $\lambda_{-c}(x)$ to t_2 (the same goes with t_1 and t_2 switched). Third and last, we take care of the denominators: let N be the product of all the denominators appearing in the reduced fractions of the c's appearing in ϕ. Then any atomic formula $t_1 > t_2$ is replaced with the atomic formula $t'_1 > t'_2$ where any c (appearing either as a constant or as λ_c) is replaced by $N \times c$: the fact that $c \geq 0$ implies that $(N \times c) \in \mathbb{N}$. Moreover, for any assignment, the value of t'_1 (resp. t'_2) is N times the value of t_1 (resp. t_2), hence, the value of t_1 is greater than the value of t_2 iff the same holds for t'_1 and t'_2.

Over $\mathbb{N}$, the automaton is needed to incorporate some of the constraint set:

▶ **Lemma 21.** *Every $\mathbb{N}$-APA (A, U, C) has the same language as a $\mathbb{N}$-APA (A, U', C') where C' can be expressed as a basic formula. The same holds for $\mathbb{N}$-DetAPA.*

Quantif. elim. of $\mathbb{N}$-definable sets, see p. 3

Proof. Recall that a semilinear set can be expressed as a basic formula with the additional relations $\equiv_p$, expressing congruence (e.g., [End72]). Thus we need only get rid of these relations. To do so, we equip the affine functions to compute their own value modulo p.

Let (A, U, C) be an $\mathbb{N}$-APA (resp. $\mathbb{N}$-DetAPA) of dimension d. We suppose the initial state of A has no incoming transition (Lemma 19), and let $\phi(x_1, \dots, x_d)$ be the formula for C of the form previously mentioned (i.e., a basic formula with the additional relations $\equiv_p$). Suppose ϕ is not a basic formula, then there is a p such that $\equiv_p$ appears in ϕ. We define (A, U', C') of the same language as (A, U, C) with C' expressed by ϕ in which the $\equiv_p$ relation, for this specific p, is replaced by some basic formulas. Applying this process repeatedly gives an $\mathbb{N}$-APA (resp. $\mathbb{N}$-DetAPA) of the same language with its constraint set expressible as a basic formula.

Our goal is to modify U so that for each $\mathbf{v} \in \{0, \dots, p-1\}^d$, there is an additional variable $m_\mathbf{v}$ available to ϕ which is set to 1 iff the value of x_i modulo p is v_i, for all $1 \leq i \leq d$ (thus only one of the $m_\mathbf{v}$'s can be set to 1). With this information available, all the atomic formulas of the form $t_1 \equiv_p t_2$, for this specific p, can be rewritten without $\equiv_p$ using a basic formula:

$$t_1 \equiv_p t_2 \quad \rightsquigarrow \quad \bigvee_{\substack{\mathbf{v} \in \{0, \dots, p-1\}^d \\ t_1(\mathbf{v}) \equiv_p t_2(\mathbf{v})}} (m_\mathbf{v} = 1) \ .$$

Let t be a transition of A; we give $U'_t \in \mathscr{F}^{\mathbb{N}}_{d+p^d}$. In order to do this, we define an additional 0-1-matrix M_t of dimension $p^d \times p^d$, which we index by vectors in $\{0, \dots, p-1\}^d$ in some natural way (in particular, 0^d is the index of the first row). We let $M_t[\mathbf{u}, \mathbf{v}] = 1$ iff $\mathbf{u} = U_t(\mathbf{v}) \bmod p$, where the modulo is taken component-wise.

We are now ready to define $U'_t(\mathbf{x}, \mathbf{m})$, for $\mathbf{x} \in \mathbb{N}^d$ and $\mathbf{m} \in \mathbb{N}^{p^d}$. If t is an outgoing transition of the initial state of A, then:

$$U'_t(\mathbf{x}, \mathbf{m}) = (U_t(\mathbf{x}), M_t.(1, \mathbf{0})) \ ,$$

where $(1, \mathbf{0})$ is the column vector $(1, 0, \dots, 0) \in \mathbb{N}^{p^d}$. Otherwise, we let:

$$U'_t(\mathbf{x}, \mathbf{m}) = (U_t(\mathbf{x}), M_t.\mathbf{m}) \ .$$

Note that in both definitions, U'_t is indeed an affine function. Now with $\mathbf{m}_1$ (resp. $\mathbf{m}_2$) the vector in $\{0, 1\}^{p^d}$ having a 1 only in position $\mathbf{x} \bmod p$ (resp. $U_t(\mathbf{x}) \bmod p$), we have $U'_t(\mathbf{x}, \mathbf{m}_1) = (U_t(\mathbf{x}), \mathbf{m}_2)$. Moreover, the initial value of $\mathbf{x}$ being 0^d, those hypotheses are established at the first transition taken. Thus for a nonempty path

π, and with $\mathbf{m}$ the vector in $\{0,1\}^{p^d}$ having a 1 only in position $U_\pi(0^d) \bmod p$, we have: $U'_\pi(\mathbf{0}) = (U_\pi(0^d), \mathbf{m})$.

As previously discussed, ϕ can now be rewritten as ϕ' without the use of $\equiv_p$: ϕ' has access to the usual variables $x_1, \ldots, x_d$ and to variables $m_\mathbf{v}$ for $\mathbf{v} \in \{0, \ldots, p-1\}^d$. We take care of the empty word by letting ϕ' consider $m_\mathbf{0}$ to be 1 if no other $m_\mathbf{v}$ variable is set. Thus, with C' the set defined by ϕ', we have that $L(A, U, C) = L(A, U', C')$ and ϕ' has one less p appearing as $\equiv_p$ than ϕ.

Before proving the main result of this section, we show that *affine* functions, in their full generality, are not needed within $\mathbb{K}$-APA or $\mathbb{K}$-DetAPA:

▶ **Lemma 22.** *The language of any $\mathbb{K}$-APA is also the language of a $\mathbb{K}$-APA in which every affine function is either constant or linear — for both $\mathbb{K} = \mathbb{Q}$ and $\mathbb{K} = \mathbb{N}$. Moreover, the first transition of a run is always associated with a constant function. The same holds for $\mathbb{K}$-DetAPA.*

Proof. Let (A, U, C) be a $\mathbb{K}$-APA (resp. $\mathbb{K}$-DetAPA) of dimension d and suppose, thanks to Lemma 19, that the initial state of A has no incoming transition. We define a $\mathbb{K}$-APA (resp. $\mathbb{K}$-DetAPA) (A', U', C') where the outgoing transitions of the initial state of A initialize the registers with the values of all the constant parts given by U. Specifically, we define the morphism $U' : \delta' \to \mathcal{F}^{\mathbb{K}}_{d+dn}$ as follows. Identify the transition set of A with $\{t_1, \ldots, t_n\}$, write $U_{t_i} = (M_i, \mathbf{v}_i)$, for $i \in \{1, \ldots, n\}$, and define $\hat{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_n) \in \mathbb{K}^{dn}$. Then for t an outgoing transition of the initial state, U'_t is the constant function with value $(U_t(0^d), \hat{v})$; for the other t_i's, we set $U'_{t_i}(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_n) = (M_i.\mathbf{x} + \mathbf{y}_i, \mathbf{y}_1, \ldots, \mathbf{y}_n)$, and in this case, U'_{t_i} is the linear function $(M'_i, \mathbf{0})$ where here and in the following $\mathbf{0}$ is of dimension $d + dn$ and:

$$M'_i = \begin{pmatrix} M_i & \mathbb{0}_d & \cdots & \mathbb{0}_d & \overset{\overset{(i+1)\text{-th block of width } d}{\downarrow}}{Id_d} & \mathbb{0}_d & \cdots & \mathbb{0}_d \\ \mathbb{0}_d & & & & & & & \\ \vdots & & & & & & & \\ \vdots & & & & & & & \\ \vdots & & & & \boxed{Id_{dn}} & & & \\ \vdots & & & & & & & \\ \vdots & & & & & & & \\ \mathbb{0}_d & & & & & & & \end{pmatrix}$$

with $\mathbb{0}_k$ (resp. Id_k) the zero (resp. identity) matrix of dimension $k \times k$. Finally, we let $C' = C \times \mathbb{K}^{dn}$.

We now show that $L(A, U, C) = L(A, U', C')$. First, $\varepsilon \in L(A, U, C)$ iff $\varepsilon \in L(A)$ and $U_\varepsilon(\mathbf{0}) = \mathbf{0} \in C$, the latter being equivalent to $U'_\varepsilon(\mathbf{0}) = \mathbf{0} \in C'$, thus $\varepsilon \in L(A, U, C)$ iff $\varepsilon \in L(A, U', C')$. Now let π be a path in A starting from the initial state. Suppose $|\pi| = 1$ then $U'_\pi(\mathbf{0}) = (U_\pi(0^d), \hat{v})$. For $|\pi| > 1$, let $\pi = \rho t$, then:

$$U'_\pi(\mathbf{0}) = U'_t(U'_\rho(\mathbf{0})) \underset{\text{by induction}}{=} U'_t(U_\rho(0^d), \hat{v}) = (U_\pi(0^d), \hat{v}) \ .$$

Thus let w be a nonempty word in $L(A, U, C)$ and let π be an accepting path in A labeled w and such that $U_\pi(\mathbf{0}) \in C$. Then $U'_\pi(\mathbf{0}) = (U_\pi(0^d), \ldots)$ which is in $C \times \mathbb{K}^{dn} = C'$, thus $w \in L(A, U', C')$. Conversely, let w be a nonempty word in $L(A, U', C')$ and let π be an accepting path in A labeled w such that $U'_\pi(\mathbf{0}) \in C'$. We have that $U'_\pi(\mathbf{0}) = (U_\pi(0^d), \ldots)$ is in $C' = C \times \mathbb{K}^{dn}$, thus $U_\pi(0^d) \in C$ and $w \in L(A, U, C)$.

We are now ready to show that the choice of $\mathbb{K}$ in APA does not influence the class of languages defined:

▶ **Theorem 23.** *$\mathscr{L}_{\mathbb{Q}\text{-DetAPA}} = \mathscr{L}_{\mathbb{N}\text{-DetAPA}}$ and $\mathscr{L}_{\mathbb{Q}\text{-APA}} = \mathscr{L}_{\mathbb{N}\text{-APA}}$. Moreover, these correspondences are effective and do not change the underlying automaton.*

Proof. ($\mathscr{L}_{\mathbb{Q}\text{-APA}} \subseteq \mathscr{L}_{\mathbb{N}\text{-APA}}$ and $\mathscr{L}_{\mathbb{Q}\text{-DetAPA}} \subseteq \mathscr{L}_{\mathbb{N}\text{-DetAPA}}$.) Let (A, U, C) be a $\mathbb{Q}$-APA (resp. $\mathbb{Q}$-DetAPA) of dimension d, with δ the set of transitions of A. The underlying automaton A will remain the same throughout the proof.

We suppose that the empty word is not in $L(A)$ it is a simple task to add it back at the very end of this construction if needed. Thanks to Lemma 22, we assume that all the functions given by U are either linear or constant. Lemma 20 then asserts that C is expressible as a basic formula ϕ. We first ensure that no constant appears in ϕ by replacing each of them by a variable (e.g., if $\lambda_3(x) + 8$ is a term in ϕ, we replace it by $\lambda_3(x) + y$ where y is a new variable). Let ϕ' be this modified formula and, for $c_1 < c_2 < \ldots < c_p \in \mathbb{K}$ he increasing sequence of the p constants that appear in ϕ, let $y_1, y_2, \ldots, y_p$ be the associated sequence of new variables. We now update U so that it gives the value c_i to y_i, for all i. Let $\mathbf{c} = (c_1, c_2, \ldots, c_p)$ and define U' from U as follows. For t such that U_t is constant, set $U'_t(\mathbf{x}_1, \mathbf{x}_2) = (U_t(0^d), \mathbf{c})$, which is still a constant function; and for t such that U_t is linear, set $U'_t(\mathbf{x}_1, \mathbf{x}_2) = (U_t(\mathbf{x}_1), \mathbf{x}_2)$, which is also still linear. We let C' be the set described by ϕ'; it verifies $\{\mathbf{x} \mid (\mathbf{x}, \mathbf{c}) \in C'\} = C$. As the first transition of any run in A is associated, by U, with a constant function, any nonempty run π in A verifies $U'_\pi(\mathbf{0}) = (U_\pi(0^d), \mathbf{c})$. Thus the variables $y_1, \ldots, y_p$ of ϕ' are indeed set to $c_1, \ldots, c_p$, implying that $L(A, U, C) = L(A, U', C')$. From now on we denote $d + p$ by n.

We now change U' so that the constants and matrices appearing in the U'_t's are all integer-valued. Let N be the product of all the denominators appearing in the reduced fractions of the entries of the matrices and vectors given by U'. By defining $U''_t = N \times U'_t$, we thus ensure that all the values appearing in the definition of U''_t are integers, hence U'' is a function from δ^* to $\mathscr{F}_n^{\mathbb{Z}}$. Moreover, as all the functions are either linear or constant, this implies that for any path π, there is a $k \leq |\pi|$ such that $U''_\pi = N^k \times U'_\pi$. But as all the atomic formulas of ϕ' are of the form $t_1 < t_2$ where no constant appears and all the λ_c have $c > 0$, we have that $\phi'(\mathbf{v})$ is true iff $\phi'(K \times \mathbf{v})$ is true. Thus $L(A, U, C) = L(A, U'', C')$.

Finally, we change U'' and C' so that they are $\mathbb{N}$-valued. We define U''' as U'' where the positive and negative computations are made in different components. Consider U''_t as n affine functions from $\mathbb{Q}^n$ to $\mathbb{Q}$: $U''_t(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$. Then let $1 \leq i \leq n$, and write $f_i(\mathbf{x}) = c + \sum_{j=1}^{n} v_j \times x_j$.

Let us write $J^+ = \{1 \leq j \leq n \mid v_j \geq 0\}$ and $J^- = \{1 \leq j \leq n \mid v_j < 0\}$. Now, we define f_i^+ and f_i^- by:

$$f_i^+(\mathbf{x}^+, \mathbf{x}^-) = \max(c, 0) + \sum_{j \in J^+} |v_j| \times x_j^+, \text{and,}$$

$$f_i^-(\mathbf{x}^+, \mathbf{x}^-) = |\min(c, 0)| + \sum_{j \in J^-} |v_j| \times x_j^- \ .$$

Now define $U_t''' : \mathbb{N}^{2n} \to \mathbb{N}^{2n}$ as:

$$U_t'''(\mathbf{x}^+, \mathbf{x}^-) = (f_1^+, f_1^-, \dots, f_n^+, f_n^-)(\mathbf{x}^+, \mathbf{x}^-) \ ,$$

where $\mathbf{x}^+, \mathbf{x}^- \in \mathbb{N}^n$. The main property of this construction is that for a path π, we have:

$$U_\pi'''(\mathbf{0}) = (a_1^+, a_1^-, \dots, a_n^+, a_n^-) \quad \Rightarrow \quad U_\pi''(\mathbf{0}) = (a_1^+ - a_1^-, \dots, a_n^+ - a_n^-) \ .$$

Thus define C'' as:

$$C'' = \{(a_1^+, a_1^-, \dots, a_n^+, a_n^-) \mid (a_1^+ - a_1^-, \dots, a_n^+ - a_n^-) \in C'\} \ .$$

Now C'' is a $\mathbb{Q}$-definable set because C' is $\mathbb{Q}$-definable, thus C'' is expressible as a basic formula. But basic formulas on natural numbers take their truth values regardless of whether $\mathbb{K} = \mathbb{N}$ or $\mathbb{K} = \mathbb{Q}$, thus $C'' \cap \mathbb{N}^{2n}$ is $\mathbb{N}$-definable. Finally, $(A, U''', C'' \cap \mathbb{N}^{2n})$ is an $\mathbb{N}$-APA (resp. $\mathbb{N}$-DetAPA) of the same language as (A, U, C).

($\mathscr{L}_{\mathbb{N}\text{-APA}} \subseteq \mathscr{L}_{\mathbb{Q}\text{-APA}}$ and $\mathscr{L}_{\mathbb{N}\text{-DetAPA}} \subseteq \mathscr{L}_{\mathbb{Q}\text{-DetAPA}}$.) This is a consequence of Lemma 21. Let (A, U, C) be an $\mathbb{N}$-APA (resp. $\mathbb{N}$-DetAPA). Now, by Lemma 21, let (A, U', C') be an $\mathbb{N}$-APA (resp. $\mathbb{N}$-DetAPA) with the same language and with C' expressible as a basic formula. The fact that basic formulas take their truth value on natural numbers regardless of the underlying model implies that there exists a $\mathbb{Q}$-definable set C'' such that $C'' \cap \mathbb{N}^d = C'$ — this is the set described by the basic formula for C' interpreted in $\mathbb{Q}$ — and thus (A, U', C'') is a $\mathbb{Q}$-APA (resp. $\mathbb{Q}$-DetAPA) of the same language as (A, U, C).

The previous result allows us to write $\mathscr{L}_{\text{DetAPA}}$ for $\mathscr{L}_{\mathbb{Q}\text{-DetAPA}} = \mathscr{L}_{\mathbb{N}\text{-DetAPA}}$ and $\mathscr{L}_{\text{APA}}$ for $\mathscr{L}_{\mathbb{Q}\text{-APA}} = \mathscr{L}_{\mathbb{N}\text{-APA}}$.

2.2 Closure properties of $\mathscr{L}_{\text{APA}}$ and $\mathscr{L}_{\text{DetAPA}}$

The *pointed* concatenation of L and L' is any language of the form $L \cdot \{\#\} \cdot L'$ where # does not appear in a word of L. The arguments used by Klaedtke and Rueß [KR02] apply equally well to $\mathbb{K}$-APA and $\mathbb{K}$-DetAPA, showing:

▶ **Proposition 24.** *(1) $\mathscr{L}_{\text{APA}}$ is closed under union, intersection, concatenation, non-erasing morphisms, and inverse morphisms; (2) $\mathscr{L}_{\text{DetAPA}}$ is closed under union, intersection, inverse morphisms, complement, and pointed concatenation.*

Proof. (Union and intersection.) Let (A', U', C') and (A'', U'', C'') be two $\mathbb{K}$-APA (resp. $\mathbb{K}$-DetAPA) of dimension d' and d'', respectively, and suppose that A' and A'' are complete and with every state final (Lemma 19). We suppose moreover, w.l.o.g., that the alphabets of the automata are the same. Let $L' = L(A', U', C')$ and $L'' = L(A'', U'', C'')$. We construct two $\mathbb{K}$-APA (resp. $\mathbb{K}$-DetAPA) $(A, U, C^{\cup})$ and $(A, U, C^{\cap})$ such that their languages are the union and intersection, respectively, of L' and L''. Let $A' = (Q', \Sigma', \delta', q'_0, Q')$ and $A'' = (Q'', \Sigma', \delta'', q''_0, Q'')$, and define the Cartesian product of A' and A'' by $A = (Q' \times Q'', \Sigma', \delta, (q'_0, q''_0), Q' \times Q'')$ with:

$$\delta = \{(p', p'') \bullet\!\!-a\!\!\to (q', q'') \mid p' \bullet\!\!-a\!\!\to q' \in \delta' \wedge p'' \bullet\!\!-a\!\!\to q'' \in \delta''\} \ .$$

This automaton is deterministic if both A' and A'' are. Define h' (resp. h''), to be the morphism from δ^* to $(\delta')^*$ (resp. to $(\delta'')^*$) such that:

$$h'((p', p'') \bullet\!\!-a\!\!\to (q', q'')) = p' \bullet\!\!-a\!\!\to q'$$
$$(\text{resp. } h''((p', p'') \bullet\!\!-a\!\!\to (q', q'')) = p'' \bullet\!\!-a\!\!\to q'')$$

The fact that A' and A'' are complete implies that for any run π' in A' and π'' in A'' with the same label, there is a run π in A such that $h'(\pi) = \pi'$ and $h''(\pi) = \pi''$. Then we let $U : \delta^* \to \mathcal{F}^{\mathbb{K}}_{d'+d''}$ compute the values of U' in the first d' components and the values of U'' in the last d'' components, that is, for $\mathbf{x}' \in \mathbb{K}^{d'}$, $\mathbf{x}'' \in \mathbb{K}^{d''}$, and $t \in \delta$:

$$U_t(\mathbf{x}', \mathbf{x}'') = (U'_{h'(t)}(\mathbf{x}'), U''_{h''(t)}(\mathbf{x}'')) \ .$$

Finally, we let $C^{\cup} = C' \times \mathbb{K}^{d''} \cup \mathbb{K}^{d'} \times C''$ and $C^{\cap} = C' \times C''$. We argue that $L(A, U, C^{\cup}) = L' \cup L''$ and $L(A, U, C^{\cap}) = L' \cap L''$.

Let π be a run in A. Then $h'(\pi)$ is a run in A', $h''(\pi)$ is a run in A'', and both have the same label as π. Moreover, $U_\pi(\mathbf{0}) = (U'_{h'(\pi)}(0^{d'}), U''_{h''(\pi)}(0^{d''}))$. Thus if $U_\pi(\mathbf{0}) \in C^{\cup}$ then $\mu_A(\pi) \in L(A, U, C^{\cup})$, and $U'_{h'(\pi)}(0^{d'}) \in C'$ or $U''_{h''(\pi)}(0^{d''}) \in C''$, thus $\mu_A(\pi) \in L' \cup L''$. Likewise, if $U_\pi(\mathbf{0}) \in C^{\cap}$ then $\mu_A(\pi) \in L(A, U, C^{\cap})$, and both $U'_{h'(\pi)}(0^{d'}) \in C'$ and $U''_{h''(\pi)}(0^{d''}) \in C'$ thus $\mu_A(\pi) \in L' \cap L''$.

For the converse, let $w \in L'$ and let π' be a run in A' such that $\mu_{A'}(\pi') = w$ and $U'_{\pi'}(\mathbf{0}) \in C'$. Then there is a run π in A such that $h'(\pi) = \pi'$. Moreover, $U_\pi(\mathbf{0}) = (U'_{h'(\pi)}(\mathbf{0}), U''_{h''(\pi)}(\mathbf{0}))$ which is in $C' \times \mathbb{K}^{d''}$, thus in $C^{\cup}$, and thus $w \in L(A, U, C^{\cup})$. Likewise, if $w \in L''$, then $w \in L(A, U, C^{\cup})$. Now let $w \in L' \cap L''$, and let π' (resp. π'') be a run in A' (resp. A'') such that $\mu_{A'}(\pi') = w$ and $U'_{\pi'}(0^{d'}) \in C'$ (resp. $\mu_{A''}(\pi'') = w$ and $U''_{\pi''}(0^{d''}) \in C''$). There exists a path π in A such that $h'(\pi) = \pi'$ and $h''(\pi) = \pi''$, and it is such that $U_\pi(\mathbf{0}) = (U'_{h'(\pi')}(\mathbf{0}), U''_{h''(\pi')}(\mathbf{0}))$ which is in $C' \times C''$, that is, $C^{\cap}$, thus $w \in L(A, U, C^{\cap})$.

(Inverse morphisms.) We first tackle the $\mathbb{K}$-DetAPA case, which is based on the classical construction on finite automata and followed by the addition of the affine functions. Let (A, U, C) be a $\mathbb{K}$-DetAPA over the alphabet Σ, and let $h : \Sigma'^* \to \Sigma^*$ be a morphism; we will give a $\mathbb{K}$-DetAPA (A', U', C) for the language $h^{-1}(L(A, U, C))$. We first construct A' such that its language is $h^{-1}(L(A))$. Let $A = (Q, \Sigma, \delta, q_0, F)$, and write $\mathsf{Path}(q, u, q')$ for the only path in A from q to q' labeled u if it exists, $\perp$ otherwise. Further, we let $\mathsf{Path}(q, \varepsilon, q) = \varepsilon$, i.e., we consider that the empty path is going and ending in any given state. Then $A' = (Q, \Sigma', \delta', q_0, F)$ where:

$$\delta' = \{q \bullet\!\!-a\!\!\to q' \in Q \times \Sigma' \times Q \mid \mathsf{Path}(q, h(a), q') \neq \perp\} \enspace .$$

The automaton A' is such that $L(A') = h^{-1}(L(A))$ and is deterministic. In particular, if $h(a) = \varepsilon$, then a loop labeled a appears on each state.

When a word w is read in A' from some state q to a state q', the equivalent action in A is to take the path $\mathsf{Path}(q, h(w), q')$; thus we let $U'_{q \bullet\!-a\!\to q'} = U_{\mathsf{Path}(q,h(a),q')}$, and in particular, if a transition is labeled with a letter a such that $h(a) = \varepsilon$, then the associated function is the identity. Then for π' a path in A' and π its counterpart in A (i.e., $\pi = \mathsf{Path}(\mathsf{From}(\pi'), h(\mu(\pi')), \mathsf{To}(\pi'))$), we have that $U'_{\pi'} = U_\pi$. Now let $w \in L(A')$, π' be the accepting path with label w in A', and π be the accepting path with label $h(w)$ in A. Then $U'_{\pi'}(\mathbf{0}) = U_\pi(\mathbf{0})$. Thus we have that $h(w) \in L(A, U, C)$ iff $h(w) \in L(A)$ and the path π for $h(w)$ in A is such that $U_\pi(\mathbf{0}) \in C$, which is the case iff $w \in L(A')$ and the path π' for w in A' is such that $U'_{\pi'}(\mathbf{0}) \in C$, that is iff $w \in L(A', U', C)$, concluding this case.

We now focus on the nondeterministic case. Let (A, U, C) be a $\mathbb{K}$-APA over the alphabet Σ and let $h : \Sigma'^* \to \Sigma^*$ be a morphism. Here, for some states q, q' of A, we may have several paths from q to q' with the same label — say we have k paths. To circumvent this problem, we use at least k copies of the A' of the deterministic case: we go from the i-th copy of q to the j-th of q' applying the affine functions corresponding to the j-th of the k paths.

Formally, let $A = (Q, \Sigma, \delta, q_0, F)$. Define $\mathsf{Paths}(q, u, q')$ as the *set* of paths in A from q to q' labeled u, and impose an order on this set (say, lexicographical order). Again, we consider the empty path as going and ending in any given state, thus we let $\mathsf{Paths}(q, \varepsilon, q) = \{\varepsilon\}$. Let M be the maximum number of elements in $\mathsf{Paths}(q, h(a), q')$ for $q, q' \in Q$, and $a \in \Sigma'$. We define an A' similar to the deterministic case, but duplicated M times to obtain the $\mathbb{K}$-APA (A', U', C) for $h^{-1}(L(A, U, C))$. For $1 \leq i \leq M$ and for a state q in A', we write q_i for a fresh copy of q indexed by i (when $q = q_0$ we write $(q_0)_i$ as $q_{0,i}$); we use this notation to define an automaton A' that includes M copies of the deterministic case one. Let $A' = (Q', \Sigma', \delta', q_{0,1}, F')$ where:

- $Q' = \{q_i \mid q \in Q \wedge 1 \leq i \leq M\}$,
- $\delta' = \{q_i {\bullet\!-} a {\rightarrow} q'_j \in Q' \times \Sigma' \times Q' \mid 1 \leq i, j \leq |\mathsf{Paths}(q, h(a), q')|\}$,
- $F' = \{q_i \mid q \in F \wedge 1 \leq i \leq M\}$.

Again we have that $L(A') = h^{-1}(L(A))$. Finally, define U' by:

$$U'_{q_i {\bullet\!-} a {\rightarrow} q'_j} = U_\pi \text{ where } \pi \text{ is the } j\text{-th path in } \mathsf{Paths}(q, h(a), q') \ .$$

The deterministic case corresponds to $M = 1$, and in this case, the constructed $\mathbb{K}$-APA is the same as in the previous construction. Now suppose $\mathsf{Paths}(q, h(a), q')$ has more than two elements for some $q, q' \in Q$ and $a \in \Sigma'$. In particular, the two transitions $q_1 {\bullet\!-} a {\rightarrow} q'_1$ and $q_1 {\bullet\!-} a {\rightarrow} q'_2$ are in A'; the affine functions associated are such that taking the first (resp. second) transition applies the same function as going through the first (resp. second) path of $\mathsf{Paths}(q, h(a), q')$ in A'. Thus, once again, the possible values computed by the affine functions while reading some $h(w)$ in A are the same as those computed while reading w in A'. By the same token as in the deterministic case, $L(A', U', C) = h^{-1}(L(A, U, C))$.

(Concatenation.) Let (A', U', C') and (A'', U'', C'') be two $\mathbb{K}$-APA of dimension d' and d'', respectively, and let $L' = L(A', U', C')$ and $L'' = L(A'', U'', C'')$. We construct a $\mathbb{K}$-DetAPA (A, U, C) of dimension $d' + d''$ for $L = L' \cdot L''$. Here, A is the merging of A' and A'', where for all transitions in A'' from the initial state to some state q, a transition from each final state of A' to q with the same label is added. We then compute U' and U'' in parallel.

Formally, let $A' = (Q', \Sigma', \delta', q'_0, F')$ and $A'' = (Q'', \Sigma'', \delta'', q''_0, F'')$, and suppose $Q' \cap Q'' = \emptyset$. We assume that $\varepsilon \notin L(A'')$; otherwise, if $\varepsilon \in L''$, then $L = L' \cdot (L'' \setminus \{\varepsilon\}) \cup L'$, and the closure under union allows us to conclude. Define A as the deterministic automaton $(Q, \Sigma, \delta, q_0, F)$ where:

- $Q = Q' \cup Q''$, $\Sigma = \Sigma' \cup \Sigma''$,
- $\delta = \delta' \cup \delta'' \cup \{p {\bullet\!-} a {\rightarrow} q \mid p \in F' \wedge q''_0 {\bullet\!-} a {\rightarrow} q \in \delta''\}$,
- $q_0 = q'_0$ and $F = F''$.

The language of A is thus $L(A') \cdot L(A'')$. We define $U : \delta^* \to \mathbb{K}^{d'+d''}$ so that the d' first components are used for the computations of A', and the d'' last for the computations of A'', i.e., for $\mathbf{x} \in \mathbb{K}^{d'}$ and $\mathbf{y} \in \mathbb{K}^{d''}$, we let $U_t(\mathbf{x}, \mathbf{y})$ be $(U'_t(\mathbf{x}), \mathbf{y})$ if

$t \in \delta'$, $(\mathbf{x}, U''_t(\mathbf{y}))$ if $t \in \delta''$, and $(U_{q''_0 \bullet\!-a\!\to q}(\mathbf{x}), \mathbf{y})$ if $t = p\bullet\!-a\!\to q \notin \delta' \cup \delta''$. Finally, we let C to be the $\mathbb{K}$-definable set $C' \times C''$.

Let $\pi \in \mathsf{Run}(A)$, then π can be written as $\pi'(p\bullet\!-a\!\to q)\pi''$ where $\pi' \in \mathsf{Run}(A')$ and $(q''_0\bullet\!-a\!\to q)\pi'' \in \mathsf{Run}(A'')$. Conversely, for two paths $\pi' \in \mathsf{Run}(A')$, $(q''_0\bullet\!-a\!\to q)\pi'' \in \mathsf{Run}(A'')$, the path $\pi'(\mathsf{To}(\pi')\bullet\!-a\!\to q)\pi''$ is a run in A. Moreover, in both cases, it holds that:

$$U_\pi(\mathbf{0}) = (U'_{\pi'}(0^{d'}), U''_{\pi''}(0^{d''})) \ .$$

Thus $L = L(A, U, C)$.

(Nonerasing morphisms.) Let (A, U, C) be a $\mathbb{K}$-APA over the alphabet Σ and $h\colon \Sigma^* \to \Sigma'^*$ be a nonerasing morphism, that is, for all $a \in \Sigma, h(a) \neq \varepsilon$. We construct a $\mathbb{K}$-APA for $h(L(A, U, C))$ where the main task is the following. For a letter $a \in \Sigma$ and $w = w_1 \cdots w_n = h(a)$, a transition $t = q\bullet\!-a\!\to q'$ of A is replaced by n transitions $q\bullet\!-w_1\!\to q_{t,1}$, …, $q_{t,n-1}\bullet\!-w_n\!\to q'$ where the $q_{t,i}$'s are fresh states named after the transition t. To make the proof concise, we rely on the closure under inverse morphism of $\mathbb{K}$-APA, previously shown. We give a $\mathbb{K}$-APA (A', U', C) for the image of $h(L(A, U, C))$ under the morphism g which maps $a \in \Sigma'$ to $a\#$, for $\# \notin \Sigma'$; we then have that $h(L(A, U, C)) = g^{-1}(L(A', U', C'))$, concluding the proof.

Formally, let $A = (Q, \Sigma, \delta, q_0, F)$ and for $t \in \delta$, write $q^\perp_{t,i}$ and $q^\top_{t,i}$ to denote some fresh states. Let $\#$ be a symbol not in Σ'. Then $A' = (Q', \Sigma' \cup \{\#\}, \delta', q_0, F)$ where:

- $Q' = Q \cup \{q^\perp_{t,i}, q^\top_{t,i} \mid t \in \delta \wedge 1 \leq i \leq |h(\mu(t))|\}$,
- $\delta' = \{q\bullet\!-w_1\!\to q^\perp_{t,1},\quad q^\perp_{t,i}\bullet\!-\#\!\to q^\top_{t,i},\quad q^\top_{t,i}\bullet\!-w_{i+1}\!\to q^\perp_{t,i+1},\quad q^\perp_{t,n}\bullet\!-\#\!\to q' \mid$
  $q\bullet\!-a\!\to q' \in \delta \wedge w_1 \cdots w_n = h(a) \wedge 1 \leq i < n\}$.

We now adjust U' so that the computations of the two $\mathbb{K}$-APA are the same. We let U'_t be the identity function for any t with $\mathsf{From}(t) \notin Q$, and for $t' = q\bullet\!-a\!\to q^\perp_{t,1}$, we let $U'_{t'} = U_t$. We argue that this $\mathbb{K}$-APA recognizes $h(L(A, U, C))$ with a $\#$ in every even position. First, $L(A')$ is $h(L(A))$ in which a $\#$ is inserted in every even position. Next, let $w_1\# \cdots \#w_n\# \in L(A')$ with $w_i \in \Sigma'$, and let π' be a run with this label in A' such that $U'_{\pi'}(\mathbf{0}) \in C$. Let π be the corresponding path in A defined by replacing each transition of the form $q\bullet\!-a\!\to q^\perp_{t,1}$ by t and removing the other transitions. Then π is an accepting path, its label is in $h^{-1}(w_1 \cdots w_n)$, and $U_\pi(\mathbf{0}) = U'_{\pi'}(\mathbf{0})$, thus $w_1 \cdots w_n \in h(L(A, U, C))$. Conversely, if $w \in L(A, U, C)$, then let π be a run with label w in A such that $U_\pi(\mathbf{0}) \in C$. Then the path π' in A' whose only states of the form $q^\top_{t,1}$ are $q^\top_{\pi_1,1}, \ldots, q^\top_{\pi_{|\pi|},1}$ in that order is accepting and such that $U'_{\pi'}(\mathbf{0}) = U_\pi(\mathbf{0})$ which is in C. Thus its label, which is $h(w)$ with $\#$ inserted in every even position, is in $L(A', U', C)$.

(Complement.) Let (A, U, C) be a $\mathbb{K}$-DetAPA. Clearly, a word does *not* belong to $L(A, U, C)$ iff it is not in $L(A)$ or, while being in $L(A)$, the path π corresponding to the word is such that $U_\pi(\mathbf{0}) \notin C$. Thus the complement of $L(A, U, C)$ is $\overline{L(A)} \cup L(A, U, \overline{C})$, which is in $\mathscr{L}_{\mathbb{K}\text{-DetAPA}}$, semilinear sets being closed under complement.

(Pointed concatenation.) This is similar to the closure under concatenation for the nondeterministic case. Let (A', U', C') and (A'', U'', C'') be two $\mathbb{K}$-DetAPA and # a symbol not in the alphabet of A'. The main difference with the closure under concatenation of $\mathbb{K}$-APA is that the automaton A is constructed by adding #-labeled transitions from the final states of A' to the initial state of A''. As # is a symbol which is not in the alphabet of A', this preserves the determinism.

Remark. These closures are *effective* in the sense that for every operation (e.g., intersection of $\mathbb{K}$-APA), there is an algorithm which computes it (e.g., given two $\mathbb{K}$-APA computes a $\mathbb{K}$-APA whose language is the intersection of the languages of the two). Also, we give the closure of $\mathscr{L}_{\text{DetAPA}}$ under *pointed* concatenation because we were not able to give a construction for the usual concatenation — we even conjecture that $\mathscr{L}_{\text{DetAPA}}$ is not closed under the usual concatenation.

We now give a large class of languages belonging to $\mathscr{L}_{\text{APA}}$ in two steps. First, we show that the language PAL of pointed palindromes, i.e., $\text{PAL} = \{w\#w^{\text{R}} \mid w \in \{a, b\}^*\}$, is recognized by a deterministic APA:

▶ **Proposition 25.** $\text{PAL} \in \mathscr{L}_{\text{DetAPA}}$.

Proof. We sketch an $\mathbb{N}$-DetAPA (A, U, C) for PAL over $\{0, 1\}^*$ rather than $\{a, b\}$. The automaton A accepts words of the form $u\#v$, with $u, v \in \{0, 1\}^*$. The affine functions compute the value of u (resp. v) seen as a binary number with the most (resp. least) significant bit first. Checking that those values are equal and that $|u| = |v|$ is then the same as checking that $u = v^{\text{R}}$.

Formally, $A = (\{q_0, q_1\}, \{0, 1, \#\}, \delta, q_0, \{q_1\})$ where δ is defined, together with the affine functions of U, by:

$$\begin{array}{lllllll}
t_1 = q_0 \bullet\!\!-0\!\!\to q_0 & \text{performs } (x, p, y, \ell) \mapsto (& 2x, & 0, & 0, & \ell + 1), \\
t_2 = q_0 \bullet\!\!-1\!\!\to q_0 & \text{performs } (x, p, y, \ell) \mapsto (& 2x + 1, & 0, & 0, & \ell + 1), \\
t_3 = q_0 \bullet\!\!-\#\!\!\to q_1 & \text{performs } (x, p, y, \ell) \mapsto (& x, & 1, & 0, & \ell), \\
t_4 = q_1 \bullet\!\!-0\!\!\to q_1 & \text{performs } (x, p, y, \ell) \mapsto (& x, & 2p, & y, & \ell - 1), \\
t_5 = q_1 \bullet\!\!-1\!\!\to q_1 & \text{performs } (x, p, y, \ell) \mapsto (& x, & 2p, & y + p, & \ell - 1).
\end{array}$$

Now when reading a word $u \in \{0,1\}^*$ from q_0, with x, p, y, and ℓ starting at 0, the final value is $(x, 0, 0, |u|)$ where x is the value of u seen as a binary number with the most significant bit first. Reading u from q_1 with starting value $(x, 1, 0, \ell)$ leads to the value $(x, 2^{|u|}, y, \ell - |u|)$ with y the value of u seen as a binary number with the least significant bit first. Thus, letting C to be the semilinear set $\{(n, n', n, 0) \mid n, n' \in \mathbb{N}\}$ means that we check, on reading $u\#v$, that $|u| = |v|$ and, in this case, that $u = v^{\mathrm{R}}$, hence $L(A, U, C) = \mathrm{PAL}$.

Now recall that a semi-AFL is a family of languages closed under nonerasing morphisms, inverse morphisms, intersection with a regular language, and union. Define $\mathscr{M}_\cap(L)$ as the smallest semi-AFL containing L and closed under intersection. The closure properties of $\mathscr{M}_\cap(\mathrm{PAL})$ are implied by those of $\mathscr{L}_{\mathrm{APA}}$ (Proposition 24), hence:

▶ **Proposition 26.** $\mathscr{M}_\cap(\mathrm{PAL}) \subseteq \mathscr{L}_{\mathrm{APA}}$.

We do not know whether $\mathscr{M}_\cap(\mathrm{PAL}) \subseteq \mathscr{L}_{\mathrm{DetAPA}}$ essentially since we do not know whether $\mathscr{L}_{\mathrm{DetAPA}}$ is closed under nonerasing morphisms, though we conjecture it is not.

The class $\mathscr{M}_\cap(\mathrm{PAL})$ contains a wide range of languages. First, the closure of PAL under nonerasing morphisms, inverse morphisms, and intersection with regular sets is the class of *linear languages* (e.g., [Bra81][3]). In turn, adding closure under intersection permits to express the languages of nondeterministic multipushdown automata where in every computation, each pushdown store makes a bounded number of reversals (that is, going from pushing to popping) [BNP74]; in particular, if there is only one such pushdown store, this corresponds to the *ultralinear languages* [GS66b]. Further, as $\mathscr{M}_\cap(\mathrm{COPY}) \subseteq \mathscr{M}_\cap(\mathrm{PAL})$ (e.g., [Bra81]) this implies that $\mathrm{COPY} \in \mathscr{L}_{\mathrm{APA}}$.

Next, we note that APA express only context-sensitive languages (CSL):

▶ **Proposition 27.** $\mathscr{L}_{\mathrm{APA}} \subseteq \mathrm{CSL}$.

Proof. Let (A, U, C) be an $\mathbb{N}$-APA of dimension d, we show that $L(A, U, C) \in \mathrm{NSPACE}[n]$ (which is equal to CSL [Kur64]). Let $A = (Q, \Sigma, \delta, q_0, F)$, and $w = w_1 \cdots w_n \in \Sigma^*$. First, initialize $\mathbf{v} \leftarrow \mathbf{0}$ and $q \leftarrow q_0$. Iterate through the letters w_i of w: on the i-th letter, choose nondeterministically a transition $t = q \bullet\!\!-w_i\!\!\rightarrow q' \in \delta$. Update $\mathbf{v}$ by setting $\mathbf{v} \leftarrow U_t(\mathbf{v})$ and q with $q \leftarrow q'$. Upon reaching the last letter of w, accept w iff $q \in F$ and $\mathbf{v} \in C$.

[3]Brandenburg [Bra81] defines PAL as $\{w\overline{w}^{\mathrm{R}} \mid w \in \{a,b\}^*\}$, where $\overline{w}$ is the image of w by the morphism $a \mapsto \overline{a}$ and $b \mapsto \overline{b}$, for $\overline{a}, \overline{b}$ two fresh symbols. We note that the results of [Bra81] carry over with our definition of PAL.

We now bound the value of $\mathbf{v}$. Let c be the greatest value appearing in any of the matrices or vectors in U_t, for any t. For a given $\mathbf{v}$, let $\max \mathbf{v}$ be $\max\{v_1, \ldots, v_d\}$. Then for any t, $(U_t(\mathbf{v}))_i \leq d \times (c \times \max \mathbf{v}) + c$. Let π be a path, we then have that $(U_\pi(\mathbf{0}))_i \leq (c(d+1))^{n-1}c$, thus the size of $\mathbf{v}$ at the end of the algorithm is in $O(n)$. Now note that, as C is semilinear, the language of the binary encoding of its elements is regular [WB95], and thus, checking $\mathbf{v} \in C$ can be done in, say, logarithmic space. Hence the given algorithm is indeed in NSPACE[n].

We now show that $\mathscr{L}_{\text{DetAPA}}$ is not closed under morphisms and we deduce new undecidability results. We rely on the following technical lemma that illustrates the subtle way in which a DetAPA can "perform the conjunction of an unbounded number of conditions by maintaining a nonzero flag." Let SPACING be the language $\{(a^m\#a^m\#)^n \mid m, n \geq 0\}$; note, for instance, that $a\#a\#a\#a\#$ is in SPACING while $a\#a\#aa\#aa\#$ is not.

▶ **Lemma 28.** SPACING $\in \mathscr{L}_{\text{DetAPA}}$.

Proof. Let $\Sigma = \{a, \#\}$, $L_0 = \{a^m\#a^m\# \mid m \geq 0\}^*$ and $L_1 = L_0 \cdot a^*\#a^*$. Then:

$$\begin{aligned}\text{SPACING} &= \{\varepsilon\} \cup [L_0 \cap a^*\# \cdot L_0 \cdot a^*\#] \\ &= \{\varepsilon\} \cup [L_0 \cap a^*\# \cdot (L_1 \cap \Sigma^*\#)].\end{aligned}$$

We will show that $L_0, L_1 \in \mathscr{L}_{\text{DetAPA}}$. This implies the result as follows. Since $\mathscr{L}_{\text{DetAPA}}$ is closed under intersection (Figure 1), $L_1 \cap \Sigma^*\# \in \mathscr{L}_{\text{DetAPA}}$. By closure of $\mathscr{L}_{\text{DetAPA}}$ under pointed concatenation (Proposition 24), $a^*\# \cdot (L_1 \cap \Sigma^*\#) \in \mathscr{L}_{\text{DetAPA}}$. Applying closure properties again yields SPACING $\in \mathscr{L}_{\text{DetAPA}}$. (Note that L_1 is needed to express SPACING because $L_0 \in \mathscr{L}_{\text{DetAPA}}$ is not known to imply $L_0 \cdot a^*\# \in \mathscr{L}_{\text{DetAPA}}$.)

We first construct a $\mathbb{Q}$-DetAPA D_0 on two registers x and y for L_0. As its underlying automaton, D_0 will have a two-state automaton A with initial and final state q_0. The 4 transitions of A, and 4 affine functions $\mathbb{Q}^2 \to \mathbb{Q}^2$ assigned to these transitions, are:

$$\begin{aligned}
t_1 &= q_0 \bullet\!\!-a\!\!\to q_0 \text{ performs } \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x+1 \\ 2y \end{pmatrix}, \\
t_2 &= q_0 \bullet\!\!-\#\!\!\to q_1 \text{ performs } \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \end{pmatrix}, \\
t_3 &= q_1 \bullet\!\!-a\!\!\to q_1 \text{ performs } \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x-1 \\ 2y \end{pmatrix}, \\
t_4 &= q_1 \bullet\!\!-\#\!\!\to q_0 \text{ performs } \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 0 \\ x+y \end{pmatrix}.
\end{aligned}$$

As usual, $\begin{pmatrix} x \\ y \end{pmatrix}$ is $\binom{0}{0}$ initially. The constraint set C_0 for D_0 will be $\{\binom{0}{0}\}$ which is $\mathbb{Q}$-definable. (Only integers will ever appear in the counters; we use $\mathbb{Q}$ rather than $\mathbb{N}$ only to have access to negative integers.) Surprisingly, this works.

We must argue that $L(D_0) = L_0$. We will write $(q, \begin{pmatrix} i \\ j \end{pmatrix})$ for the configuration of D_0 in which the state of A is $q \in \{q_0, q_1\}$ and i and j are the contents of registers x and y. For $w \in \Sigma^*$, we will write $(q, \begin{pmatrix} i \\ j \end{pmatrix})_w$ for the configuration reached when A starts in configuration $(q, \begin{pmatrix} i \\ j \end{pmatrix})$ and reads w. We need to prove two facts:

(i) $\forall w \in L_0,\ (q_0, \binom{0}{0})_w = (q_0, \binom{0}{0})$;

(ii) $\forall w \in (a^*\#a^*\#)^*a^*$, if $(q_0, \binom{0}{0})_w = (q_0, \binom{0}{0})$ then $w \in L_0$.

Fact (i) proves $L_0 \subseteq L(D_0)$ because q_0 is final in A and $\binom{0}{0} \in C_0$. Fact (ii) proves $L(D_0) \subseteq L_0$ because $L(A)$ is seen to be $(a^*\#a^*\#)^*a^*$; hence fact (ii) states that any word that is in $L(A)$ and that further sets $\begin{pmatrix} x \\ y \end{pmatrix}$ to $\binom{0}{0}$ belongs to L_0.

To prove fact (i), let $w = a^{m_1}\#a^{m_1}\#a^{m_2}\#a^{m_2}\#\cdots a^{m_k}\#a^{m_k}\#$ for $k \geq 0$. Any $w \in L_0$ has this form, and an induction on k shows that $(q_0, \binom{0}{0})_w = (q_0, \binom{0}{0})$.

To prove fact (ii), we make the following claim, crucial to the operation of D_0:
Claim. For any $u, v \in \Sigma^*$, if $(q_0, \binom{0}{0})_u$ sets $y \neq 0$ then $(q_0, \binom{0}{0})_{uv}$ also sets $y \neq 0$.
We first show that this implies fact (ii). Let $w \in (a^*\#a^*\#)^*a^i$ satisfy $(q_0, \binom{0}{0})_w = (q_0, \binom{0}{0})$. We show $w \in L_0$. Identify w with the word $a^{m_1}\#a^{m_2}\#\cdots a^{m_{2k-1}}\#a^{m_{2k}}\#a^i$ for some $k \geq 0$. By the Claim, every prefix of w sets $y = 0$. Because reading every second # returns A to q_0 and resets x to 0, we necessarily have

$$\begin{aligned}(q_0, \tbinom{0}{0}) &= (q_0, \tbinom{0}{0})_{a^{m_1}\#a^{m_2}\#}\\ &= (q_0, \tbinom{0}{0})_{a^{m_1}\#a^{m_2}\#a^{m_3}\#a^{m_4}\#}\\ &= (q_0, \tbinom{0}{0})_{a^{m_3}\#a^{m_4}\#}\\ &\ \vdots\\ &= (q_0, \tbinom{0}{0})_{a^{m_{2k-1}}\#a^{m_{2k}}\#},\end{aligned}$$

and

$$(q_0, \tbinom{0}{0})_w = (q_0, \tbinom{0}{0})_{a^i} = (q_0, \tbinom{i}{0}).$$

By inspection of A, $(q_0, \binom{0}{0}) = (q_0, \binom{0}{0})_{a^m\#a^{m'}\#}$ implies $m = m'$. Hence $m_1 = m_2, m_3 = m_4, \ldots, m_{2k-1} = m_{2k}$. Moreover, $(q_0, \binom{0}{0})_w = (q_0, \binom{0}{0})$ by assumption. Hence $i = 0$ and $w = a^{m_1}\#a^{m_1}\#\cdots a^{m_{2k-1}}\#a^{m_{2k-1}}\# \in L_0$, concluding fact (ii).

We now prove the Claim. Let $u \in \Sigma^*$ be such that $(q_0, \binom{0}{0})_u$ sets $y \neq 0$. We need to show that for all $v \in \Sigma^*$, $(q_0, \binom{0}{0})_{uv}$ also sets $y \neq 0$. Let $u = u_1u_2$ where u_1 is the shortest prefix of u that sets $y \neq 0$. By inspection of A, $u_1 = u'a^i\#a^j\#$ for some $u' \in (a^*\#a^*\#)^*$ such that $(q_0, \binom{0}{0})_{u_1} = (q_0, \binom{0}{0})_{a^i\#a^j\#} = (q_0, \binom{0}{i-j})$ and $i \neq j$. We will prove by induction on the length of w that for any $w \in \Sigma^*$, $(q_0, \binom{0}{i-j})_w = (q, \binom{x_w}{y_w})$ for some $q \in \{q_0, q_1\}$ and some $x_w, y_w \in \mathbb{Q}$ such that $|y_w| \geq \max\{1, 2|x_w|\}$. This will complete the proof of the Claim since we can pick $w = u_2v$, and conclude that $(q_0, \binom{0}{0})_{uv} = (q_0, \binom{0}{i-j})_{u_2w} = (q, \binom{x_{u_2v}}{y_{u_2v}})$ with $|y_{u_2v}| \geq \max\{1, 2|x_{u_2v}|\} > 0$.

For the basis of the induction, let $w = \varepsilon$. Then $(q_0, \binom{0}{i-j})_w = (q_0, \binom{0}{i-j})$. Now $|i-j| \geq 1 = \max\{1, 2 \times |0|\}$. For the inductive step, let $w \in \Sigma^n$ for some $n > 0$. Then $w = va$ or $w = v\#$ and by induction, $(q_0, \binom{0}{i-j})_v = (q, \binom{x_v}{y_v})$ with $|y_v| \geq \max\{1, 2|x_v|\}$.

Case 1: $w = va$. Then $(q, \binom{0}{i-j})_{va} = (q, \binom{x_v \pm 1}{2y_v})$. If $x_v = 0$, then $|y_{va}| = |2y_v| \geq 2\max\{1, 2|x_v|\} = 2 = \max\{1, 2 \times |\pm 1|\} = \max\{1, 2|x_{va}|\}$. If $x_v \neq 0$, then

$|y_{va}| = |2y_v| \geq 2\max\{1, 2|x_v|\} = 2(|x_v| + |x_v|) \geq 2(|x_v| + 1) \geq \max\{1, 2|x_v \pm 1|\} = \max\{1, 2|x_{va}|\}$.

Case 2: $w = v\#$. If t_2 is the transition that consumed the last #, then $x_v = x_{v\#}$ and $y_v = y_{v\#}$ so the induction hypothesis immediately yields $|y_w| \geq \max\{1, 2|x_w|\}$. So let t_4 be the transition that consumed the last #. Then $(q, \binom{0}{i-j})_{v\#} = (q, \binom{0}{x_v+y_v})$. Now $|y_{va}| = |x_v + y_v| \geq |y_v| - |x_v| \geq \max\{1, 2|x_v|\} - |x_v| \geq \max\{1, |x_v|\} \geq 1 = \max\{1, 2\times|0|\} = \max\{1, 2\times|x_{v\#}|\}$. This concludes the proof of the Claim and the proof that $L(D_0) = L_0$.

We have yet to construct a $\mathbb{Q}$-DetAPA D_1 for L_1. The automaton underlying D_1 will be A, as above, except that the final state will be q_1 rather than q_0. The affine functions associated with the transitions remain the same. The constraint set C_1 will be $\{\binom{r}{0} : r \in \mathbb{Q}\}$ and it is $\mathbb{Q}$-definable. We need to prove the following facts:
(iii) $\forall w \in L_1$, $(q_0, \binom{0}{0})_w = (q_1, \binom{i}{0})$ for some $i \in \mathbb{Q}$;
(iv) $\forall w \in (a^*\#a^*\#)^*a^*\#a^*$, if $(q_0, \binom{0}{0})_w = (q_1, \binom{i}{0})$ then $w \in L_1$.

Fact (iii) follows from fact (i) since any $w \in L_1$ is of the form $w = ua^i\#a^j$ with $u \in L_0$, so that $(q_0, \binom{0}{0})_w = (q_0, \binom{0}{0})_{a^i\#a^j} = (q_1, \binom{i-j}{0})$. To prove fact (iv), let $w \in (a^*\#a^*\#)^*a^*\#a^*$ satisfy $(q_0, \binom{0}{0})_w = (q_1, \binom{x_w}{0})$. By the Claim above, every prefix of w sets $y = 0$. By inspection of A, some suffix $a^i\#a^j$ of w must have sent A to state q_1, that is, $w = ua^i\#a^j$ with $(q_0, \binom{0}{0})_u = (q_0, \binom{0}{0})$ and $x_w = i - j$. But then, $u \in L_0$ by fact (ii) and thus $w \in L_1$. This concludes the proof that $L(D_1) = L_1$ and proves the lemma.

▶ **Lemma 29.** *Given a Turing machine M, we can construct a morphism h and a DetAPA D such that $L(M) = h(L(D))$.*

Proof. We adapt [BB74, Theorem 1]. With no loss of generality, we assume that M is a one-tape Turing machine that accepts by halting and makes an odd number of moves on any accepting computation. Let L_1 (resp. L_2) be the set of strings

$$ID_0\#ID_2\#\cdots\#ID_{2k}\$(ID_{2k+1})^R\#\cdots\#(ID_3)^R\#(ID_1)^R\# \tag{2}$$

such that ID_i, $0 \leq i \leq 2k+1$, are instantaneous descriptions of M padded with the blank symbol b to a common length ℓ, $ID_0 = [^{w_1}_{q_0}]w_2\cdots w_n b^{\ell-n}$ codes the initial configuration of M ($[^{w_1}_{q_0}]$ is considered as a single letter, and, $w_1 = b$ when the word $w = w_1w_2\cdots w_n \in \Sigma^*$ input to M is ε), ID_{2k+1} codes an accepting configuration and for $0 \leq i \leq k$, ID_{2i+1} (resp. for $0 < i \leq k$, ID_{2i}) codes the configuration which would be reached in one step from configuration ID_{2i} (resp. ID_{2i-1}). Each ID_i other than ID_0 is coded using an alphabet Γ disjoint from $\Sigma \cup \{[{}^{\sigma}_{q_0}] \mid \sigma \in \Sigma\} \cup \{[{}^{b}_{q_0}]\}$. It should be clear that $w \in L(M)$ iff $w \in h(L_1 \cap L_2)$ where for every $\sigma \in \Sigma$ and every $\gamma \in \Gamma$,

$$h([{}^{\sigma}_{q_0}]) = h(\sigma) = \sigma \text{ and } h([{}^{b}_{q_0}]) = h(b) = h(\#) = h(\$) = h(\gamma) = \varepsilon \enspace .$$

To complete the proof, we claim that $L_1 \cap L_2 \in \mathscr{L}_{\text{DetAPA}}$ in the effective sense. Since $\mathscr{L}_{\text{DetAPA}}$ is closed under intersection in that sense (Figure 1), it suffices to show that $L_1 \in \mathscr{L}_{\text{DetAPA}}$ and $L_2 \in \mathscr{L}_{\text{DetAPA}}$. We first show how to construct a $DetAPA$ recognizing L_1. We will construct an $\mathbb{N}$-DetAPA D_1 able to handle only the words of the form (2) in which the distance between any two consecutive symbols # or \$ is $|ID_0|$. Handling only those words will be sufficient because the language L_1 can then be expressed as $L_1 = g^{-1}(\text{SPACING}) \cap L(D_1)$ where g is the morphism mapping both # and \$ to # and mapping every other symbol to the letter a. Since Lemma 28 shows SPACING $\in \mathscr{L}_{\text{DetAPA}}$ and since $\mathscr{L}_{\text{DetAPA}}$ is closed under intersection and inverse morphisms in the effective sense (Figure 1), a DetAPA for L_1 can be constructed from D_1.

So we now describe D_1. Let m be the size of the alphabet Γ. We argue as if ID_0 in (2) were coded over the same alphabet Γ used to code ID_i for $0 < i \leq 2k+1$, since a finite automaton can easily adjust for this. Our strategy will extend the strategy used to construct an $\mathbb{N}$-DetAPA for pointed palindromes (Proposition 25) as follows. As D_1 reads the prefix $ID_0\#ID_2\# \cdots \#ID_{2k}$ of (2), D_1 will internally translate that prefix into $ID_1 ID_3 \cdots ID_{2k+1}$ and will treat the latter as the prefix u of a pointed palindrome $u\$u^{\mathrm{R}}$. As D_1 processes u, D_1 builds in a register the natural number having u as its m-ary representation with the most significant bit first (as in Proposition 25, where m was 2). Then D_1 encounters \$ and begins to do the matchup with the suffix $(ID_{2k+1})^{\mathrm{R}}\# \cdots \#(ID_3)^{\mathrm{R}}\#(ID_1)^{\mathrm{R}}\#$ of (2). D_1 does this matchup by internally translating this suffix into:

$$(ID_{2k+1})^{\mathrm{R}} \cdots (ID_3)^{\mathrm{R}}(ID_1)^{\mathrm{R}} = (ID_1 ID_3 \cdots ID_{2k+1})^{\mathrm{R}} = u^{\mathrm{R}} \enspace .$$

As D_1 processes this suffix, D_1 computes in a register the natural number having the suffix as its m-ary representation, with the least significant bit first this time (again as in Proposition 25, now with m rather than 2). We set D_1 to accept iff reading (2) indeed leads to $u\$u^{\mathrm{R}}$ with ID_{2k+1} final. Two subtleties are worth mentioning concerning processing the prefix. First, when processing ID_{2i}, D_1 always reads one symbol ahead of position p to determine the proper symbol at position p

in ID_{2i+1}, to account for the input head of M possibly moving left from position $p+1$ to position p. Second, D_1 rejects immediately if ID_0 is not a legal coding of an initial configuration of M or if another ID_i in the prefix contains two input head symbols. This completes the operational description of D_1. The formal definition (A_1, U_1, C_1) of D_1 thus needs to implement these operations. The $\mathbb{N}$-definable set C_1 is the set $C \subseteq \mathbb{N}^4$ given in the proof of Proposition 25 but adapted to handle m-ary representation rather than binary. The affine functions assigned to the transitions of A_1 are the identity function together with the five (adapted) functions assigned to the transitions t_1, t_2, t_3, t_4, t_5 from the proof of Proposition 25: transitions performing the bookkeeping operations of A_1 (such as when A_1 processes the symbol #) will be assigned the identity function, and transitions that discover the next PAL symbol are assigned the affine transformation prescribed by Proposition 25 on reading that symbol (with the understanding that reading \$ here corresponds to reading # there).

The strategy to construct an $\mathbb{N}$-DetAPA D_2 recognizing L_2 is of course similar. But now, since ID_2 (for example) does not uniquely determine ID_1, the prefix of (2) is handled by D_2 as the suffix of (2) was handled by D_1. Specifically, D_2 internally translates the prefix $ID_0\#ID_2\#\cdots\#ID_{2k}$ into $u = ID_2ID_4\cdots ID_{2k}$ and stores the m-ary number u in a register, with the most significant bit first. Then D_2 encounters \$, discards $(ID_{2k+1})^{\mathrm{R}}$ and translates the remainder $(ID_{2k-1})^{\mathrm{R}}\#\cdots\#(ID_3)^{\mathrm{R}}\#(ID_1)^{\mathrm{R}}\#$ of the suffix into $(ID_{2k})^{\mathrm{R}}\cdots(ID_4)^{\mathrm{R}}(ID_2)^{\mathrm{R}} = (ID_2ID_4\cdots ID_{2k})^{\mathrm{R}} = u^{\mathrm{R}}$. As did D_1 when reading the prefix, D_2 needs to look ahead by one symbol while processing the suffix. The matchup with u^{R} is otherwise done by D_2 just as the matchup was done by D_1. This completes the description of D_2 and proves the lemma.

▶ **Corollary 30.** *Neither $\mathscr{L}_{\mathrm{APA}}$ nor $\mathscr{L}_{\mathrm{DetAPA}}$ is closed under morphisms.*

Proof. Given a Turing machine M, we can construct a morphism h and a DetAPA (and a fortiori an APA) D such that $L(M) = h(L(D))$. If either $\mathscr{L}_{\mathrm{APA}}$ or $\mathscr{L}_{\mathrm{DetAPA}}$ were closed under morphisms, then the language $h(L(D))$ would be the language of an APA. But the language of any APA is context-sensitive (Proposition 27), thus decidable, so we could decide $L(M)$.

▶ **Corollary 31.** *The problems of emptiness, universality, inclusion, finiteness, and regularity are undecidable for* DetAPA*.*

Proof. (Emptiness, universality, and inclusion.) Given a Turing machine M with $L(M) \subseteq \Sigma^*$, let h be the morphism and D the DetAPA provided by Lemma 29. For any $x \in \Sigma^*$, $x \in L(M)$ iff $x = h(y)$ for some $y \in L(D)$ iff $L(D) \cap h^{-1}(x)$ is nonempty. Now $\{x\} \in \mathscr{L}_{\text{DetAPA}}$ and $\mathscr{L}_{\text{DetAPA}}$ is closed (in the effective sense) under inverse morphisms and intersection (Figure 1). Hence we can construct a $DetAPA$ for $L(D) \cap h^{-1}(x)$ and deciding its emptiness would decide $x \in L(M)$. Moreover, $\mathbb{K}$-DetAPA being closed under complement (in the effective sense), the emptiness problem reduces to the universality problem. Finally, the undecidability of emptiness implies that we cannot decide if the language of a $\mathbb{K}$-DetAPA is included in the empty set.

(Finiteness and regularity [pointed out by Andreas Krebs]) Let $L \subseteq \Sigma^*$ be a language of $\mathscr{L}_{\text{DetAPA}}$, and let $\# \notin \Sigma$. Then $L \cdot \{\#a^n b^n \mid n \in \mathbb{N}\}$ is in $\mathscr{L}_{\text{DetAPA}}$ and effectively constructible from the given $DetAPA$, as a pointed concatenation (Proposition 24). Its language is finite iff it is regular iff L is empty.

Remark. None of these results allow us to conclude that $\mathscr{L}_{\text{DetAPA}}$ and $\mathscr{L}_{\text{APA}}$ are different, though we conjecture they are. One argument supporting this conjecture is the fact that DetAPA do not need their automaton: we can encode the transition function of an automaton *within* the affine functions, showing that any language of $\mathscr{L}_{\text{DetAPA}}$ can be expressed using a two-state DetAPA.

Note

See Prop. 38, p. 62 in Discussion for a proof; see also Paper IV, Cor. 103 for the separation of the classes under the hypothesis that P $\neq$ NP

3 Parikh automata on letters

The PA *on letters* requires that the “weight” of a transition depends only on the input letter from Σ triggering the transition. In a way similar to the CA characterization of PA, we characterize PA on letters solely in terms of automata over Σ and semilinear sets. We further give expressiveness and closure properties of the classes of languages that arise.

▶ **Definition 1** (Parikh automaton on letters). A *Parikh automaton on letters* (LPA) is a PA (A, C) where whenever $(a, \mathbf{v}_1)$ and $(a, \mathbf{v}_2)$ are labels of some transitions in A, then $\mathbf{v}_1 = \mathbf{v}_2$. We write $\mathscr{L}_{\text{LPA}}$ (resp. $\mathscr{L}_{\text{DetLPA}}$) for the class of languages recognized by LPA (resp. LPA which are DetPA).

First, we prove that $\mathscr{L}_{\text{LPA}}$ and $\mathscr{L}_{\text{DetLPA}}$ coincide:

▶ **Theorem 32.** $\mathscr{L}_{\text{LPA}} = \mathscr{L}_{\text{DetLPA}}$.

Proof. Let (A, C) be a LPA. Without loss of generality, we can consider $A = (Q, \Sigma \times D, \delta, q_0, F)$ to be deterministic (this does not imply that the PA is deterministic). Now let $t, t' \in \delta$ with $t = p \bullet\!\!-(a, \mathbf{v}_1)\!\!\rightarrow q$ and $t' = p \bullet\!\!-(a, \mathbf{v}_2)\!\!\rightarrow q'$. The fact that (A, C) is a LPA implies that $\mathbf{v}_1 = \mathbf{v}_2$, and A being deterministic, this implies that $q = q'$, and in turn that $t = t'$. Thus (A, C) is a DetPA.

For $L \subseteq \Sigma^*$ and $C \subseteq \mathbb{N}^{|\Sigma|}$, recall that $L{\upharpoonright_C} = \{w \in L \mid \mathsf{Pkh}(w) \in C\}$. Then:

▶ **Proposition 33.** *Let $L \subseteq \Sigma^*$ be a language. The following are equivalent:*

(i) $L \in \mathscr{L}_{\text{LPA}}$;

(ii) There exist a regular language $R \subseteq \Sigma^*$ and a semilinear set $C \subseteq \mathbb{N}^{|\Sigma|}$ such that $R{\upharpoonright_C} = L$.

Proof. *(i) → (ii)* Let (A, C) be a LPA which is a DetPA over $\{a_1, \ldots, a_n\}$. For $1 \leq i \leq n$, let $\mathbf{v}_i$ be the only vector appearing as the label $(a_i, \mathbf{v}_i)$ of a transition in A. Define $C' \subseteq \mathbb{N}^n$ by $(x_1, \ldots, x_n) \in C' \Leftrightarrow \sum_i x_i \times \mathbf{v}_i \in C$. Then let $w \in \Sigma^*$ and ω be the word which can be read from the initial state of A with $\Psi(\omega) = w$. We have that $\sum_i \mathsf{Pkh}(w)_i \times \mathbf{v}_i = \Phi(\omega)$, and thus $w \in L(A, C)$ iff $w \in \Psi(L(A)){\upharpoonright_{C'}}$.

(ii) → (i) Let $R \subseteq \{a_1, \ldots, a_n\}^*$ be a regular language and $C \subseteq \mathbb{N}^n$ be a semilinear set. Let A be an automaton for R, and change each transition label a_i in A by $(a_i, \mathbf{e}_i)$. Now for $\omega \in L(A)$, $\Phi(\omega) = \mathsf{Pkh}(\Psi(\omega))$ and thus (A, C) is a LPA with language $R{\upharpoonright_C}$.

The following property will be our central tool for showing nonclosure results:

▶ **Lemma 34.** *Let $L \in \mathscr{L}_{\text{LPA}}$. For any regular language E:*

$$L \cap E \text{ is not regular} \quad \Rightarrow \quad (\exists w \in E)[c(w) \cap L = \emptyset] \ .$$

Proof. Let $R \subseteq \Sigma^*$ be a regular language and $C \subseteq \mathbb{N}^{|\Sigma|}$ be a semilinear set. Define $L = R{\upharpoonright_C}$. Let E be a regular language such that $L \cap E$ is not regular. As $L \subseteq R$, we have $(L \cap E) \subseteq (R \cap E)$. The left hand side being non regular, those two sets differ. Thus, let $w \in (R \cap E)$ such that $w \notin L \cap E$, we have $w \notin L$. Hence, $w \in (R \setminus L)$, which implies that $\mathsf{Pkh}(w) \notin C$, and in turn, $c(w) \cap L = \emptyset$.

▶ **Proposition 35.** *(1) $\mathscr{L}_{\text{LPA}}$ is not closed under union, complement, concatenation, nonerasing morphisms, and starring; (2) $\mathscr{L}_{\text{LPA}}$ is closed under intersection, commutative closure, and inverse morphisms.*

Proof. (1) Let $L = \{a^m b^n \mid m, n \in \mathbb{N} \wedge m \neq n\}$ be a language of LPA. (Union.) Suppose $L' = L \cup \overline{a^* b^*} \in \mathscr{L}_{\text{LPA}}$. Let E be the regular language $(a^+ b^+)$. By the pumping lemma, $L' \cap E$ is not regular, thus Lemma 34 states there exists $w \in E$ such that $c(w) \cap L' = \emptyset$. But $u = b^{|w|_b} a^{|w|_a} \in c(w)$ and $u \in L'$, a contradiction, thus $L' \notin \mathscr{L}_{\text{LPA}}$. (Complement.) We note that L' is the complement in $\{a, b\}^*$ of $\{a^n b^n \mid n \in \mathbb{N}\}$, which is the language of an LPA. (Concatenation.) Suppose $L^2 \in \mathscr{L}_{\text{LPA}}$. Again, as $L^2 \cap E^2$ is not regular, Lemma 34 asserts that there exists $w \in E^2$ such that $c(w) \cap L^2 = \emptyset$. But $a^{|w|_a} b^0 a^0 b^{|w|_b} \in c(w) \cap L^2$, a contradiction, thus $L^2 \notin \mathscr{L}_{\text{LPA}}$. (Nonerasing morphism.) We note that L^2 is the image of the LPA language $\{a_1^m b_1^n a_2^r b_2^s \mid m \neq n \wedge r \neq s\}$ by the nonerasing morphism $h(a_i) = a, h(b_i) = b, i \in \{1, 2\}$. Also, by the very definition of constrained automata (Definition 12), each language of $\mathscr{L}_{\text{PA}}$ is the image by a nonerasing morphism of a language of $\mathscr{L}_{\text{LPA}}$, but the two classes are different. (Starring.) The proof of the nonclosure under starring of $\mathscr{L}_{\text{PA}}$ (Proposition 18) shows that the starring of $\{a^n b^n \mid n \in \mathbb{N}\}$ is not in $\mathscr{L}_{\text{PA}}$, thus not in $\mathscr{L}_{\text{LPA}}$.

(2) Let $R, R' \subseteq \Sigma^*$ be two regular languages and let $C, C' \subseteq \mathbb{N}^{|\Sigma|}$ be two semilinear sets. (Intersection.) Note that $(R \upharpoonright_C) \cap (R' \upharpoonright_{C'}) = (R \cap R') \upharpoonright_{C \cap C'}$, the latter being a language of $\mathscr{L}_{\text{LPA}}$. (Commutative closure.) Likewise, note that $c(R \upharpoonright_C) = \Sigma^* \upharpoonright_{C \cap \mathsf{Pkh}(R)}$, which is in $\mathscr{L}_{\text{LPA}}$ since $\mathsf{Pkh}(R)$ is effectively semilinear by Parikh's theorem. (Inverse morphism.) Let $h : \{a_1, \ldots, a_n\}^* \to \Sigma^*$ be a morphism, and let $C^h = \{\mathbf{x} \in \mathbb{N}^n \mid \sum_i x_i \times \mathsf{Pkh}(h(a_i)) \in C\}$. Then we claim that $h^{-1}(R \upharpoonright_C) = (h^{-1}(R)) \upharpoonright_{C^h}$, which concludes the proof as $h^{-1}(R)$ is regular and C^h is semilinear. Indeed, let $w \in h^{-1}(R \upharpoonright_C)$, then $w \in h^{-1}(R)$ and $\mathsf{Pkh}(h(w)) \in C$, the latter implying that $\sum_i |w|_{a_i} \times \mathsf{Pkh}(h(a_i)) \in C$, and thus, $\mathsf{Pkh}(w) \in C^h$; in particular, if a letter a is such that $h(a) = \varepsilon$, it is discarded when looking at the Parikh image of $h(w)$. Conversely, if $w \in h^{-1}(R) \upharpoonright_{C^h}$ then $h(w) \in R$ and $\mathsf{Pkh}(h(w)) = \sum_i |w|_{a_i} \times \mathsf{Pkh}(h(a_i)) \in C$, thus $h(w) \in R \upharpoonright_C$, implying that $w \in h^{-1}(R \upharpoonright_C)$.

4 Conclusion

Figures 1 and 2 in our introductory section summarize the current state of knowledge concerning the PA and its variants studied here.

An intriguing question is whether there are context-free or context-sensitive languages outside $\mathscr{L}_{\text{APA}}$. How difficult is that question? How about $\mathscr{L}_{\text{DetAPA}}$? We have been unable to locate the latter class meaningfully. In particular, can $\mathscr{L}_{\text{DetAPA}}$ be separated from $\mathscr{L}_{\text{APA}}$?

Several questions thus remain open concerning the poorly understood (and possibly overly powerful) APA model. But surely we expect testing a LPA or a DetPA for regularity to be decidable. How can regularity be tested for these models? One avenue for future research towards this goal might be characterizing $\mathscr{L}_{\text{DetPA}}$ along the lines of algebraic automata theory.

Acknowledgments

The first author thanks L. Beaudou, M. Kaplan, and A. Lemaître for stimulating discussions and comments on early versions of this paper. We further thank the anonymous referees for their critical comments and especially for pointing out a flawed claim made in an earlier version of this work concerning DetAPA.

Discussion

In this paper, we proposed several models of computation and argued for their relevance. Among these, we started by giving a fresh view of the PA model using CA, which provide a clean and simple mathematical framework for the proofs appearing in this paper and the forthcoming ones. We note however that it is useful to have the two formalisms at hand, as in the proof of Proposition 15. We investigated the main open questions about PA using CA, and this is a study we continued throughout the different papers of this thesis.

The main original contribution of this article is the study of two novel models: LPA and APA. While LPA suffer from poor expressiveness and closure properties, APA appear to be an interesting model which, in addition to offering good properties, seem to require new techniques and approaches in order to solve the related open problems. One of the major problems left open in this article is the separation of DetAPA and APA, which we investigate in Paper IV.

We note that the closure under quotient has not been attacked in this paper. It is shown in [KR03] that $\mathscr{L}_{\text{PA}}$ is closed under quotient. The class $\mathscr{L}_{\text{DetPA}}$ is shown to enjoy the same in Paper III, p. 101. The classes of affine Parikh automata are shown to lack closure under quotient in Paper IV, Corollary 100. For completeness, we show that the class $\mathscr{L}_{\text{LPA}}$ is not closed under quotient:

▶ **Proposition 36.** *$\mathscr{L}_{\text{LPA}}$ is not closed under quotient.*

Proof. Recall the language $L \in \mathscr{L}_{\text{LPA}}$ of the proof of Proposition 35, that is: $L = \{a^m b^n \mid m, n \in \mathbb{N} \wedge m \neq n\}$. It is shown therein that $L' = L \cup \overline{a^* b^*} \notin \mathscr{L}_{\text{LPA}}$. However, the languages $L_1 = x \cdot L \cup y \cdot \overline{a^* b^*}$ and $L_2 = L \cdot x \cup \overline{a^* b^*} \cdot y$, where x, y

are two new symbols, are in $\mathscr{L}_{\text{LPA}}$. But $L' = \{x, y\}^{-1}L_1 = L_2\{x, y\}^{-1}$, which is not in $\mathscr{L}_{\text{LPA}}$.

Similarly, closure under reversal was not thoroughly studied. The DetAPA case is covered in Paper IV, Corollary 96, while the APA case is tackled in the Discussion of Paper IV, Proposition 107. We present the closure of the other models at hand:

▶ **Proposition 37.** *$\mathscr{L}_{\text{DetPA}}$ is not closed under reversal. $\mathscr{L}_{\text{LPA}}$ and $\mathscr{L}_{\text{PA}}$ are closed under reversal.*

Proof. (DetPA) We showed in Proposition 10 that EQUAL is not in $\mathscr{L}_{\text{DetPA}}$. However, the language $\{a^n\#a^n \mid n \in \mathbb{N}\} \cdot \{a, b\}^*$ which is the reversal of EQUAL is easily seen to be in $\mathscr{L}_{\text{DetPA}}$.

(LPA) Given a regular language R and a semilinear set C, we have that the language $(R{\restriction_C})^{\text{R}}$ is $R^{\text{R}}{\restriction_C}$, and R^{R} is a regular language, concluding the proof.

(PA) Let (A, C) be a PA. Suppose that A has only one final state q_{f}. Let B be the automaton A where every transition is reversed (i.e., $q \bullet\!\!-(a, \mathbf{v})\!\!\rightarrow q'$ is a transition of A iff $q' \bullet\!\!-(a, \mathbf{v})\!\!\rightarrow q$ is a transition of B), the initial state of B is q_{f} and the only accepting state of B is the initial state of A. Let π be an accepting path in A and π' the accepting path in B obtained from π^{R} by reversing each transition. Then clearly $\Psi(\pi) = (\Psi(\pi'))^{\text{R}}$ and $\Phi(\pi) = \Phi(\pi')$. As $\mathsf{Run}(B)$ is the set of paths obtained from $\mathsf{Run}(A)$ in this fashion, $L(A, C) = L(B, C)^{\text{R}}$. Now if A has $f > 1$ final state, $L(A, C)$ can be seen as $\bigcup_{i=1}^{f} L(A_i, C)$ where A_i in the automaton A where the i-th final state of A is the only one to be set final. We can then rely on the $f = 1$ case.

Also, we provide a proof of the remark made on page 56.

▶ **Proposition 38.** *Let Σ be an alphabet. There exists a two-state automaton A_Σ such that for any $\mathbb{K}$-DetAPA over Σ, there exists a $\mathbb{K}$-DetAPA accepting the same language whose underlying automaton is A_Σ.*

Proof. Let (A, U, C) be a $\mathbb{K}$-DetAPA of dimension d with $A = (Q, \Sigma, \delta, q_0, F)$, $Q = \{1, \ldots, k\}$, and $\Sigma = \{a_1, \ldots, a_m\}$. Let $N = k(d + 1)$, we show that there exist $f_{a_1}, \ldots, f_{a_m} \in \mathscr{F}_N^{\mathbb{K}}$, a $\mathbb{K}$-definable set $G \subseteq \mathbb{K}^N$, and $\mathbf{s} \in \mathbb{K}^N$ such that:

$$w = \ell_1 \cdots \ell_{|w|} \in L(A, U, C) \quad \Leftrightarrow \quad f_{\ell_{|w|}} \circ \cdots \circ f_{\ell_1}(\mathbf{s}) \in G\ . \tag{3}$$

Our goal is to represent the state in which the $\mathbb{K}$-DetAPA is with a vector of size N. This vector is composed of k smaller vectors of size $(d+1)$. On taking a path π in

A, let $q = \mathsf{To}(\pi)$ and $\mathbf{v} = (U(\pi))(0^d)$; then q and $\mathbf{v}$ describe the current configuration of the $\mathbb{K}$-DetAPA. Thus we define, for any $q \in Q$ and $\mathbf{v} \in \mathbb{K}^d$:

$$\mathsf{Vec}(q, \mathbf{v}) = (0^{d+1} \ \cdots \ 0^{d+1} \ \ \underbrace{1 \ \mathbf{v}}_{q\text{-th subvector}} \ \ 0^{d+1} \ \cdots \ 0^{d+1}) \ .$$

Now, for $t \in \delta$, let M_t and $\mathbf{b}_t$ be such that $U(t) = (M_t, \mathbf{b}_t)$. For the purpose of describing the matrix U_a below, when $t \notin \delta$ we let M_t stand for the all-zero matrix of dimension $d \times d$ and $\mathbf{b}_t$ be the all-zero vector of dimension d. Let χ be the characteristic function of δ. For $a \in \Sigma$, define:

$$U_a = \left(\begin{array}{c|c|c|c|c}
\chi((1,a,1)) & 0 \cdots 0 & \cdots & \chi((k,a,1)) & 0 \cdots 0 \\ \hline
\mathbf{b}_{(1,a,1)} & M_{(1,a,1)} & \cdots & \mathbf{b}_{(k,a,1)} & M_{(k,a,1)} \\ \hline
\vdots & \vdots & \ddots & \vdots & \vdots \\ \hline
\chi((1,a,k)) & 0 \cdots 0 & \cdots & \chi((k,a,k)) & 0 \cdots 0 \\ \hline
\mathbf{b}_{(1,a,k)} & M_{(1,a,k)} & \cdots & \mathbf{b}_{(k,a,k)} & M_{(k,a,k)}
\end{array}\right)$$

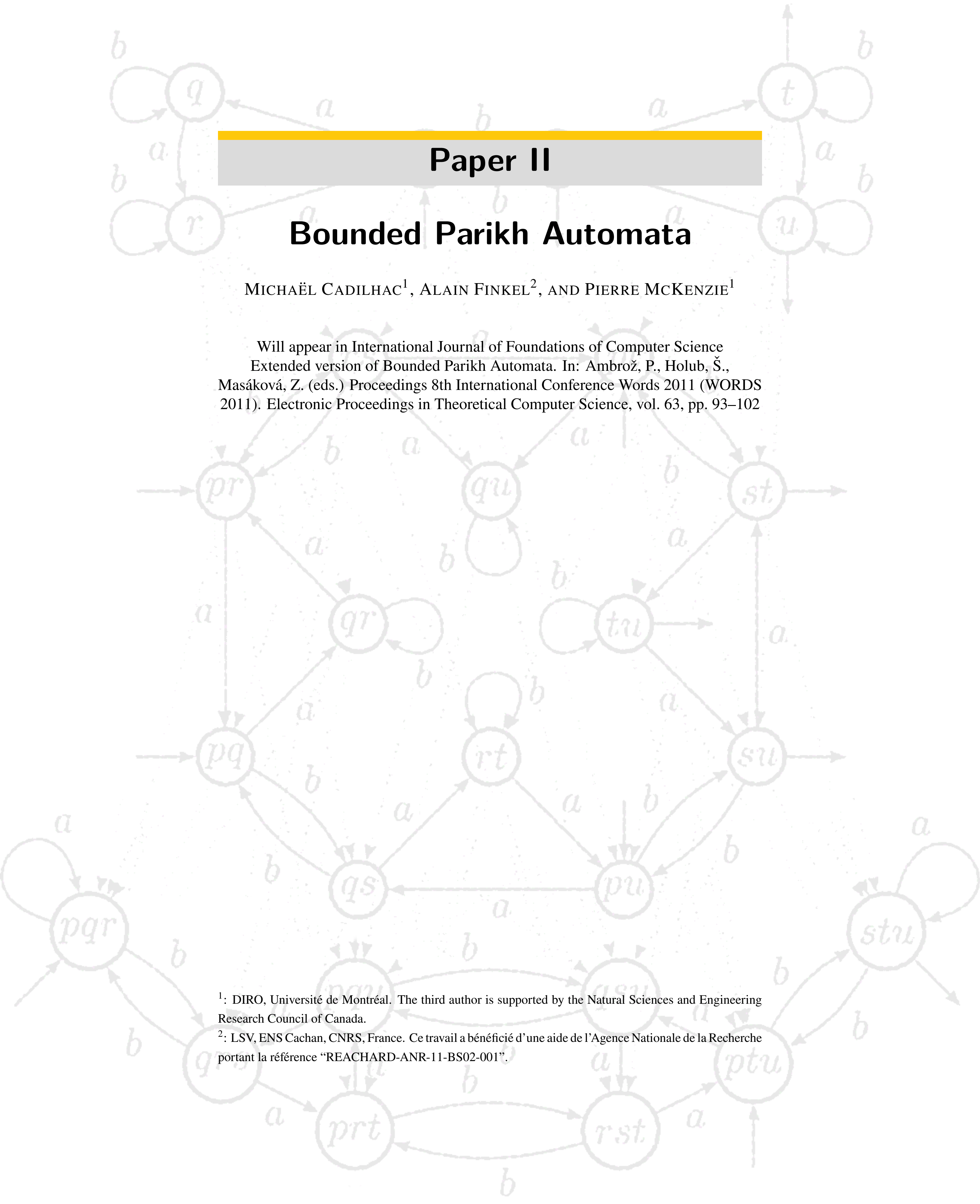
For $p \bullet\!\!- a \!\to q \in \delta$ and $\mathbf{v} \in \mathbb{K}^d$, we have that $U_a.\mathsf{Vec}(p, \mathbf{v}) = \mathsf{Vec}(q, M_{p \bullet\!- a \to q}.\mathbf{v} + \mathbf{b}_{p \bullet\!- a \to q})$. In other words, U_a computes the transition function and, according to the current state, applies the right affine function. More generally, for a path π in A starting at q_0 and labeled by $w = \ell_1 \cdots \ell_{|w|}$, we have $U_{\ell_{|w|}} \cdots U_{\ell_1}.\mathsf{Vec}(q_0, 0^d) = \mathsf{Vec}(\mathsf{To}(\pi), (U(\pi))(0^d))$, where 0^d is the all-zero vector of dimension d. We then let G be the $\mathbb{K}$-definable set which contains $\mathsf{Vec}(q, \mathbf{v})$ iff q is final and $\mathbf{v} \in C$: $G = \bigcup_{i \in F} \bigcup_{\mathbf{v} \in C} \mathsf{Vec}(i, \mathbf{v})$.

Now let $f_{a_i} \in \mathscr{F}_N^{\mathbb{K}}$ be defined as $(U_{a_i}, 0^N)$ and let $\mathbf{s} = \mathsf{Vec}(q_0, 0^d)$. Then we have precisely Equation (3). Now let $A' = (Q', \Sigma, \delta', q_0', F')$ with $Q' = F' = \{r, p\}, \delta' = \{r, p\} \times \Sigma \times \{p\}$, and, $q_0' = r$. Define $U' : \delta'^* \to \mathscr{F}_N^{\mathbb{K}}$ by:

$$U'_{q \bullet\!- a_i \to q'}(\mathbf{x}) = \begin{cases} U_{a_i}(\mathsf{Vec}(q_0, 0^d)) & \text{if } q = r \wedge q' = p, \\ U_{a_i}.\mathbf{x} & \text{otherwise, i.e., if } q = q' = p. \end{cases}$$

Finally, a special case should be added for the empty word: we let $C' = G$ if $\varepsilon \notin L(A, U, C)$ and $C' = G \cup \{0^N\}$ otherwise. We have that (A', U', C') is a

$\mathbb{K}$-DetAPA where A' has only two states, and it is of the same language as (A, U, C). Finally, note that we need two states, and not one, because $\mathbb{K}$-APA use $\mathbf{0}$ as the starting value for their registers but $\mathbf{s}$ is needed here.



Paper II

Bounded Parikh Automata

Michaël Cadilhac[1], Alain Finkel[2], and Pierre McKenzie[1]

Will appear in International Journal of Foundations of Computer Science
Extended version of Bounded Parikh Automata. In: Ambrož, P., Holub, Š., Masáková, Z. (eds.) Proceedings 8th International Conference Words 2011 (WORDS 2011). Electronic Proceedings in Theoretical Computer Science, vol. 63, pp. 93–102

[1]: DIRO, Université de Montréal. The third author is supported by the Natural Sciences and Engineering Research Council of Canada.
[2]: LSV, ENS Cachan, CNRS, France. Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence "REACHARD-ANR-11-BS02-001".

Presentation

Given that a language is BSL, it is easy to construct a CA (or a PA) for it. Moreover, it is not hard to show that any bounded language recognized by a CA is a language in BSL, hence the bounded languages of $\mathscr{L}_{\text{CA}}$ are those in BSL. The question thus arises: what are the bounded languages of $\mathscr{L}_{\text{DetCA}}$? Does nondeterminism help in the context of bounded languages? In this paper, we answer this latter question in the negative.

Bounded lang., BSL: see p. 8

The present results arose partly from the study of the *finite-monoid APA*, that is, APA (A, U, C) where $\mathscr{M}(U)$ is finite. An essential intermediate result of this paper, in Section 4.1, states that a class provably larger than DetCA (see Paper III) which includes BSL can be expressed using finite-monoid DetAPA. In Section 4.2, we then show that the bounded languages of finite-monoid DetAPA can be expressed as DetCA's.

$\mathscr{M}(U)$: see p. 2

Previous results in this vein are related to (one-way) DetRBCM, a model provably stronger than DetCA (see Paper I, Section 3.4). First, Ibarra and Su in 1999 [IS99, Theorem 3.5] showed that the bounded languages of RBCM (i.e., those of CA) are recognizable by DetRBCM when the words of the socle are of length one. Second, the fact that *two-way* DetRBCM are equivalent to RBCM over bounded languages can be found in [DIB$^+$00, Theorem 3] without proof and in [LK05] with an extremely terse proof. In 2011, while the conference version of this paper was in print, Ibarra and Seki [IS11] announced at the *Automata and Formal Languages Conference* a similar result, i.e., that one-way DetRBCM and RBCM recognize the same bounded languages. Their proof relies on techniques of combinatorics on words, while our proof is automata-driven.

Personal contribution. I proposed the results of this paper, together with their proofs. Finkel and McKenzie helped in factorizing and refining the proofs. McKenzie isolated the key concept of constraint-determinism, which became a prominent feature of the structure of the paper. The final shape of the paper came as a joint work of the authors.

Bounded Parikh Automata

Abstract

The Parikh finite word automaton model (PA) was introduced and studied by Klaedtke and Rueß. Here, we present some expressiveness properties of a restriction of the deterministic affine PA recently introduced, and use them as a tool to show that the bounded languages recognized by PA are the same as those recognized by deterministic PA. Moreover, this class of languages is shown equal to the class of bounded languages with a semilinear iteration set.

Introduction

Motivation. Adding features to finite automata in order to capture situations beyond regularity has been fruitful to many areas of research. Such features include making the state sets infinite, adding power to the logical characterizations, having the automata operate on infinite domains rather than finite alphabets, adding stack-like mechanisms, etc. (See, e.g., [Fis65, Boj09, KF94, AM09].) Model-checking and complexity theory below NC^2 are areas that have benefited from an approach of this type (e.g., [KS89, Str94]). In such areas, *determinism* plays a key role and is usually synonymous with a clear understanding of the situation at hand, yet often comes at the expense of other properties, such as expressiveness. Thus, cases where determinism can be achieved without sacrificing other properties are of particular interest.

Context. Klaedtke and Rueß introduced the Parikh automaton (PA) as an extension of the finite automaton [KR03]. A PA is a pair (A, C) where C is a semilinear subset of $\mathbb{N}^d$ and A is a finite automaton over $(\Sigma \times D)$ for Σ a finite alphabet and D a finite subset of $\mathbb{N}^d$. The PA accepts the word $w_1 \cdots w_n \in \Sigma^*$ if A accepts a word $(w_1, \mathbf{v}_1) \cdots (w_n, \mathbf{v}_n)$

such that $\sum \mathbf{v}_i \in C$. Klaedtke and Rueß used the PA to characterize an extension of (existential) monadic second-order logic in which the cardinality of sets expressed by second-order variables is available. To use PA as symbolic representations for model-checking, the closure under the Boolean operations is needed; unfortunately, PA are not closed under complement. Moreover, although they allow for great expressiveness, they are not determinizable.

Bounded languages and semilinearity. Bounded languages were defined by Ginsburg and Spanier in 1964 [GS64] and intensively studied in the sixties. Recently, they played a role in the theory of acceleration in regular model-checking [FIS03, BH99]. A language $L \subseteq \Sigma^*$ is *bounded* if there exist words $w_1, w_2, \ldots, w_n \in \Sigma^*$ such that $L \subseteq w_1^* w_2^* \cdots w_n^*$. Bounded context-free languages received much attention thanks to their better decidability properties than those of context-free languages [Gin66] (e.g., inclusion between two context-free languages is decidable if one of them is bounded, while it is undecidable in the general case). Moreover, given a context-free language it is possible to decide whether it is bounded [GS64]. Connecting semilinearity and boundedness, the class BSL of bounded languages $L \subseteq w_1^* \cdots w_n^*$, for which $\{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\}$ is a semilinear set, was also investigated (e.g., [GS64, Gin66, CMMP10, DV08]). The class BSL was also very recently characterized[4] by means of one-way deterministic reversal-bounded multi-counter machines [IS11].

Our contribution. We study PA whose languages are bounded. Our main result is that bounded PA languages are also accepted by deterministic PA, and that bounded PA languages characterize BSL. We obtain as a consequence that BSL is captured by another model studied in the literature, the 1-CQDD [BH99]. We thus provide characterizations of BSL involving a one-way model (the deterministic PA) that is mildly (but provably) weaker than the one-way deterministic reversal-bounded multi-counter machine used in [IS11]. As a tool of independent interest, our argument uses two related models introduced in Paper I: the constrained automaton (CA), which is equivalent to the PA, and the affine Parikh automaton (APA), which we subject to the restriction that the matrix semigroup generated by the set of all defining APA matrices is finite (see Definition 5).

This paper is organized as follows. Section 2 defines the PA, the equivalent CA, and 1-CQDD. Section 3 shows that the class of bounded languages recognized by PA is

[4]The proceedings of the WORDS 2011 conference, in which we lay claim to possibly providing the first characterization of BSL in terms of one-way deterministic automata, was already in print when this characterization was announced by the authors of [IS11] at the *Automata and Formal Languages Conference* held in Debrecen, Hungary, August 17–22, 2011.

BSL. Section 4 presents the main result of this work, namely that nondeterministic and deterministic PA recognize the same bounded languages. Section 5 concludes with a short discussion.

1 Preliminaries

Regular bounded languages and branches. Regular bounded languages can be characterized by a subclass of regular expressions. Let Σ be an alphabet. A *semilinear*[5] *regular expression* (SLRE) [FIS03] is a finite set of *branches* on alphabet Σ, defined as expressions of the form $y_1 x_1^* y_2 x_2^* \cdots y_n x_n^* y_{n+1}$, where $x_i \in \Sigma^+$ and $y_i \in \Sigma^*$. The language of an SLRE is the union of the languages of each of its branches. A regular language is bounded iff it is expressible as a SLRE [FIS03].

Bounded languages: see p. 6

Flat and restricted flat automata. For an ε-automaton $A = (Q, \Sigma, \delta, q_0, F)$, several notions of *flatness* have been defined in the literature and we wish to specify our usage of the word here. A *cycle* in A is a path $\pi \in \delta^+$ such that $\mathsf{From}(\pi) = \mathsf{To}(\pi)$. An *elementary* cycle in A is a cycle π in which the only repeated state is the initial (and final) state $\mathsf{From}(\pi)$. Our notion of flatness is the following: the automaton A is *flat* if no two elementary cycles in A share a state. This definition is equivalent to those in [BFLS05, DFGvD10].

Note that if A is flat, then A induces a natural directed acyclic graph D_A on the vertex set $\{[q] : q \in Q\}$, where $[q]$ is the set $\{q\}$ together with all the states reachable from q along an elementary cycle: there is an arc between $[q]$ and $[q']$, with $[q] \neq [q']$, iff there are $p \in [q]$ and $p' \in [q']$ such that there is a transition between p and p' in A. We introduce a (proper) subclass of flat automata: an ε-automaton $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ is *rflat* (for *restricted flat*) if A is flat and D_A is a straight line from $[q_0]$ to $[q_f]$.

The following properties are easy to see:

(i) No nested cycles occur in a flat automaton;
(ii) If A is flat then $\mathsf{Run}(A)$ is a regular bounded language on δ (hence SLRE);
(iii) Any regular bounded language on δ (hence SLRE) is $\mathsf{Run}(A)$ for some flat A;
(iv) If A is rflat then $\mathsf{Run}(A)$ is expressible as a branch on δ;
(v) Any language of a branch on δ is $\mathsf{Run}(A)$ for some rflat A;
(vi) A is rflat iff it is *restricted simple* in the sense of [BH99].

[5] The usage of *semilinear* here is not directly related to its usage in *semilinear set*.

Rational transduction. Let Σ and T be two alphabets. Let A be an automaton over the alphabet $(\Sigma \cup \{\varepsilon\}) \times (\mathrm{T} \cup \{\varepsilon\})$, where the concatenation is defined by $(u_1, v_1).(u_2, v_2) = (u_1u_2, v_1v_2)$. Then A defines the *rational transduction* τ_A from languages L on Σ to languages on T given by $\tau_A(L) = \{v \in \mathrm{T}^* \mid (\exists u \in L)[(u, v) \in L(A)]\}$. Closure under rational transduction for a class $\mathscr{C}$ is the property that for any language $L \in \mathscr{C}$ and any automaton A, $\tau_A(L) \in \mathscr{C}$. We say that τ_A is a *deterministic* rational transduction if A is deterministic with respect to the first component of its labels, i.e., if $p \xrightarrow{(a,b)} q$ and $p \xrightarrow{(a,b')} q'$ are transitions of A, then $b = b'$ and $q = q'$.

2 Parikh automata and constrained automata

PA, CA: see pp. 11, 14

▶ **Definition 2** (Flat constrained and Parikh automata)**.** A CA (resp. PA) (A, C) is said to be *rflat* if A is rflat.

▶ **Theorem 39.** *(i) PA and CA define the same class of languages (see Paper I);*
(ii) DetPA and DetCA define the same class of languages (see Paper I);
(iii) ε-CA and CA (and thus PA) define the same class of languages;
(iv) Rflat DetPA and rflat DetCA define the same class of languages.

Proof. Parts (i) and (ii) are proved in Paper I. Part (iii) follows from (i) and the closure of the class of PA languages under erasing morphisms [KR03]. To prove part (iv), we argue that the proof of (i) and (ii) appearing in Paper I applies verbatim.

In one direction, let (A, C) be a DetPA where A is rflat. Define B as the automaton A in which the vector-part of the labels is dropped: a transition $p \xrightarrow{(a,\mathbf{v})} q$ in A appears as $p \xrightarrow{a} q$ in B and write $h\colon \delta_A \to \delta_B$ this correspondence. Note that h is a 1-1 correspondence between the transitions of A and B thanks to the rflatness of A: indeed, for two transitions t_1, t_2 in A to get merged by h, they should be of the form $t_1 = p \xrightarrow{(a,\mathbf{v}_1)} q$ and $t_2 = p \xrightarrow{(a,\mathbf{v}_2)} q$. Suppose that $t_1 \neq t_2$, hence $\mathbf{v}_1 \neq \mathbf{v}_2$, and this would render A non rflat; we deduce that h is injective and it is surjective by construction, hence h is a bijection. This 1-1 correspondence shows in particular that B is rflat, as G_A and G_B are the same. Moreover (A, C) is a DetPA, thus it describes a deterministic automaton with respect to the first component of the labels of A, hence B is deterministic. With $\{t_1, \ldots, t_n\}$ the set of transitions of B, define D as the following semilinear set:

$$(x_1, \ldots, x_n) \in D \quad \Leftrightarrow \quad \sum_{i=1}^{n} x_i \times \Phi(\mu_A(h^{-1}(t_i))) \in C.$$

Then (B, D) is a rflat DetCA with language $L(A, C)$.

In the other direction, let (A, C) be a DetCA where $A = (Q, \Sigma, \delta, q_0, F)$ is rflat. Write $\delta = \{t_1, \dots, t_n\}$. Define B as the automaton with state set Q and with a transition $q \bullet\!\!-(a, \mathsf{Pkh}(t_i))\!\!\to q'$ for each transition $t_i = q \bullet\!\!-a\!\!\to q'$ of A. Then B is rflat, as it has the same graph as A, and (B, C) is a DetPA, as B is deterministic with respect to the first component of its labels. Finally, the language of B is:

$$L(B) = \{\omega \mid (\exists \pi \in \mathsf{Run}(A))[\Psi(\omega) = \mu_A(\pi) \wedge \Phi(\omega) = \mathsf{Pkh}(\pi)]\},$$

hence $L(B, C) = L(A, C)$.

We note that a related model has been defined and used in the context of model-checking:

▶ **Definition 3** ([BH99])**.** A 1-*CQDD* (for *constrained queue-content decision diagram*) is a finite set of rflat DetCA. Its language is the union of the languages of each DetCA. We write $\mathscr{L}_{\text{1-CQDD}}$ for the class of languages recognized by 1-CQDD.

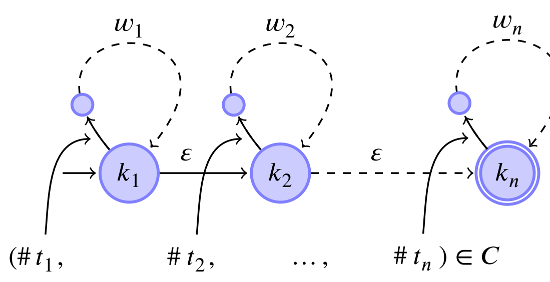
3 Bounded Parikh automata

Let $\mathscr{L}_{\text{BoundedPA}}$ be the set $\mathscr{L}_{\text{PA}} \cap$ BOUNDED of bounded PA languages, and similarly let $\mathscr{L}_{\text{BoundedDetPA}}$ be $\mathscr{L}_{\text{DetPA}} \cap$ BOUNDED.

Theorem 41 below characterizes $\mathscr{L}_{\text{BoundedPA}}$ as the class BSL of bounded semilinear languages.[6] In one direction of the proof, given $L \in$ BSL, an ε-CA for L is constructed. We describe this simple construction here:

Construction 40 (Canonical ε-CA for $w_1, \dots, w_n$ subject to $C \subseteq \mathbb{N}^n$). Let $w_1, \dots, w_n \in \Sigma^+$ be given words and $C \subseteq \mathbb{N}^n$ be a semilinear set. We describe a PA for the language $\{w_1^{i_1} w_2^{i_2} \cdots w_n^{i_n} \mid (i_1, \dots, i_n) \in C\}$. Informally, the automaton A will consist of n elementary cycles labeled $w_1, \dots, w_n$ which do not share any state, and traversed at their origins by a single ε-labeled path from k_1 leading to a unique final state k_n. Then the semilinear constraint E will be defined to monitor $(\#t_1, \dots, \#t_n)$ in accordance with C, where t_i is the first transition of the cycle for w_i and $\#t_i$ is the number of occurrences of t_i in a run of A. Graphically:

[6] This result can be deduced from the recent result of Ibarra and Seki [IS11]. However, we need its forthcoming proof to provide some corollaries.



Formally, let $k_j = \sum_{1 \le i < j} |w_i|$, $1 \le j \le n+1$, with, in particular, $k_1 = 0$, and set $Q = \{0, 1, \dots, k_{n+1} - 1\}$. Then A is the ε-automaton $(Q, \Sigma, \delta, q_0, F)$ where $q_0 = k_1$, $F = \{k_n\}$ and for any $1 \le i < n$, there is a transition $k_i \bullet\!\!-\varepsilon\!\!\to k_{i+1}$ and for any $1 \le i \le n$ an elementary cycle $t_i \rho_i$ labeled w_i through the states $k_i, k_i + 1, \dots, k_{i+1} - 1, k_i$, where t_i is a transition and ρ_i a path. Then $E \subseteq \mathbb{N}^{|\delta|}$ is the semilinear set defined by $(\#t_1, \dots, \#t_2, \dots, \dots, \#t_n, \dots) \in E$ iff $(\#t_1, \#t_2, \dots, \#t_n) \in C$.

▶ **Theorem 41.** $\mathscr{L}_{\text{BoundedPA}} = \text{BSL}$.

Proof. ($\mathscr{L}_{\text{BoundedPA}} \subseteq \text{BSL}$.) Let $L \subseteq \Sigma^*$ be a bounded language of $\mathscr{L}_{\text{PA}}$, and $w_1, \dots, w_n$ be a socle of L. Define $E = \mathsf{Iter}_{(w_1,\dots,w_n)}(L)$. Let $\mathrm{T} = \{a_1, \dots, a_n\}$ be a fresh alphabet ($\mathrm{T} \cap \Sigma = \emptyset$), and let $h : \mathrm{T}^* \to \Sigma^*$ be the morphism defined by $h(a_i) = w_i$. Then the language $L' = h^{-1}(L) \cap (a_1^* \cdots a_n^*)$ is in $\mathscr{L}_{\text{PA}}$ by closure of $\mathscr{L}_{\text{PA}}$ under inverse morphism and intersection [KR03]. But $\mathsf{Pkh}(L') = E$, and as any language of $\mathscr{L}_{\text{PA}}$ has a semilinear Parikh image [KR03], E is semilinear. Thus the iteration set E of the bounded language L with respect to its socle $w_1, \dots, w_n$ is semilinear, and this is the meaning of L belonging to BSL.

(BSL $\subseteq \mathscr{L}_{\text{BoundedPA}}$.) Let $L \in \text{BSL}$. Of course $L \in \text{BOUNDED}$. Let $w_1, \dots, w_n$ be a socle of L such that $C = \mathsf{Iter}_{(w_1,\dots,w_n)}(L)$ is semilinear. We leave out the simple proof that L equals the language of the "canonical ε-CA for $w_1, \dots, w_n$ subject to C" of Construction 40. Since ε-CA and PA capture the same languages by Theorem 39, $L \in \mathscr{L}_{\text{PA}}$.

Theorem 41 and the known closure properties of BOUNDED and $\mathscr{L}_{\text{PA}}$ imply:

▶ **Corollary 42.** BSL *is closed under union, intersection, concatenation, and morphism.*

Proof. Let $L_1, L_2 \in \text{BSL}$. By Theorem 41, both languages are in $\mathscr{L}_{\text{PA}}$. Moreover, both $\mathscr{L}_{\text{PA}}$ and BOUNDED are closed under union, intersection, concatenation, and morphisms (see, respectively, [KR03] and [Gin66]). This implies that $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, and $h(L_1)$ are all bounded languages in $\mathscr{L}_{\text{PA}}$, and by Theorem 41, are all in BSL.

We note, in the same vein, that although BSL is not closed under inverse morphism (e.g., with h the all-erasing morphism on $\{a, b\}^*$, we have $h^{-1}(\{\varepsilon\}) = \{a, b\}^*$, which is not bounded) we have:

▶ **Corollary 43.** BSL *is closed under inverse morphisms followed by the intersection with a language in* BSL*.*

Proof. Let $L_1, L_2 \in \text{BSL}$. By Theorem 41, both languages are in $\mathscr{L}_{\text{PA}}$. Moreover, $\mathscr{L}_{\text{PA}}$ is closed under intersection and inverse morphism [KR03], and the intersection of any language with a bounded language is a bounded language [Gin66]. This implies that $h^{-1}(L_1) \cap L_2$ is a bounded language in $\mathscr{L}_{\text{PA}}$, and by Theorem 41, in BSL.

Moreover, Theorem 41 helps in showing that if the iteration set of a bounded language w.r.t. *one* of its socles is semilinear, then for *every* socle of the language, the iteration set of the language w.r.t. that socle is semilinear:

▶ **Corollary 44.** BSL *is the set of bounded languages which have* all *of their iteration sets semilinear.*

Proof. If a bounded language has *all* of its iteration sets semilinear, then it is in BSL. For the reverse inclusion, note that the first half of the proof of Theorem 41 shows that for any socle of a language in $\mathscr{L}_{\text{BoundedPA}}$, the iteration set of the language w.r.t. this socle is semilinear. As $\mathscr{L}_{\text{BoundedPA}} = \text{BSL}$, all the iteration sets of a language in BSL are semilinear.

Finally, note that the iteration sets of a BSL language are semilinear sets with a special form, which depends on a socle: they contain *all* the possible ways to iterate the words of the socle to obtain a word in the language. Thus defining a bounded language "using" a semilinear set does not directly show that it is in BSL; e.g., $C = \{(x, y) \mid x \text{ is even} \land y \in \mathbb{N}\}$ is a semilinear set defining the language a^* using the socle a, a, and yet $\mathsf{Iter}_{(a,a)}(a^*) \neq C$, so C is not a semilinear iteration set of a^*, and we may not directly conclude that $a^* \in \mathrm{BSL}$. However, Theorem 41 provides an easy proof that if a bounded language is defined "using" a semilinear set, then its iteration sets w.r.t. any other prescribed socle are computable semilinear sets:

▶ **Corollary 45.** *Let $w_1, \ldots, w_n \in \Sigma^*$ and $C \subseteq \mathbb{N}^n$ be a semilinear set. Then $L = \{w_1^{i_1} \cdots w_n^{i_n} \mid (i_1, \ldots, i_n) \in C\}$ is in* BSL. *Also, for any given socle $w'_1, \ldots, w'_m$ of L, the iteration set of L w.r.t. $w'_1, \ldots, w'_m$ is a semilinear set that we can compute.*

Proof. First, Construction 40 for $w_1, \ldots, w_n$ subject to C provides an ε-CA for L. This does not directly show that $L \in \mathrm{BSL}$, since C is not, in the general case, an iteration set (i.e., $\mathsf{Iter}(L)$ for some socle). However, this shows that L is a bounded language of $\mathscr{L}_{\mathrm{PA}}$, and thus, by Theorem 41, it is a language of BSL.

For the second part, we follow the first half of the proof of Theorem 41 and check that all the operations are computable. So we construct the morphism h which maps a_i to w'_i for $1 \leq i \leq m$. The closure properties of PA being effective [KR03], we can construct a PA for $L' = h^{-1}(L) \cap (a_1^* \cdots a_m^*)$. Finally, the Parikh image of L' is a semilinear set that we can compute [KR03], concluding the proof, as this set is the iteration set of L w.r.t. $w'_1, \ldots, w'_m$.

To the best of our knowledge, Corollary 45 provides the first effective method to obtain the iteration set, w.r.t. a prescribed socle, of a bounded semilinear language described using a semilinear set.

4 Bounded Parikh automata are determinizable

Parikh automata cannot be made deterministic in general. Indeed, Klaedtke and Rueß, in [KR03], have shown that $\mathscr{L}_{\mathrm{DetPA}}$ is closed under complement while $\mathscr{L}_{\mathrm{PA}}$ is not, so that $\mathscr{L}_{\mathrm{DetPA}} \subsetneq \mathscr{L}_{\mathrm{PA}}$, and Paper I further exhibits languages witnessing the separation. In this section, we show that PA can be determinized when their language is bounded. The purpose of this section is to show:

▶ **Theorem 46.** *Every $L \in \mathscr{L}_{\mathrm{BoundedPA}}$ is the union of the languages of rflat DetCA.*

This implies:

▶ **Corollary 47.** $\mathrm{BSL} = \mathscr{L}_{\mathrm{BoundedPA}} = \mathscr{L}_{\mathrm{BoundedDetPA}} = \mathscr{L}_{\text{1-CQDD}}$.

Proof. The first equality is Theorem 41. For the second, we have $\mathscr{L}_{\text{BoundedDetPA}} \subseteq \mathscr{L}_{\text{BoundedPA}}$; for the converse, if $L \in \mathscr{L}_{\text{BoundedPA}}$ then it is the union of the languages of rflat DetCA by Theorem 46, and as $\mathscr{L}_{\text{DetPA}}$ is closed under union, $L \in \mathscr{L}_{\text{BoundedDetPA}}$. For the third, we note that rflat DetCA, and thus 1-CQDD, recognize only bounded languages, thus $\mathscr{L}_{\text{1-CQDD}} \subseteq \mathscr{L}_{\text{BoundedPA}}$; for the converse, Theorem 46 already states that $\mathscr{L}_{\text{BoundedPA}} \subseteq \mathscr{L}_{\text{1-CQDD}}$.

We show Theorem 46 in two steps. First, in Section 4.1, we note that the canonical ε-CA of Construction 40 has a crucial property that we call "constraint-determinism," i.e., the fact that the nondeterminism of the automaton is not used in the constraint set (formal definitions follow). We show that CA with this property are naturally expressed with a model of one-way deterministic automaton which allows for some counter manipulation: a restricted version of the *deterministic affine PA* introduced in Paper I. Second, in Section 4.2, we show that any bounded language accepted by such a device is a finite union of languages of rflat DetCA. We then conclude the proof of Theorem 46 in Section 4.3.

4.1 From constraint-deterministic CA to deterministic affine PA

We first formally define the property of constraint-determinism:

▶ **Definition 4** (Constraint-determinism)**.** A CA (A, C) is said to be *constraint-deterministic* if no two paths π_1 and π_2 in $\mathsf{Run}(A)$ for which $\mu_A(\pi_1) = \mu_A(\pi_2)$ can be distinguished by C. Formally:

$$(\forall \pi_1, \pi_2 \in \mathsf{Run}(A)) \quad \mu_A(\pi_1) = \mu_A(\pi_2) \Rightarrow \big(\mathsf{Pkh}(\pi_1) \in C \leftrightarrow \mathsf{Pkh}(\pi_2) \in C\big) .$$

Given a constraint-deterministic CA (A, C), we will consider the deterministic version of A and follow, within it, the paths traced in A. To this purpose, we will need a model which allows for some simple counter manipulations; we propose the *affine Parikh automaton*, that we introduced and studied in Paper I, as the right model for this task, as it allows for the needed expressiveness while providing a nice mathematical framework for the proofs.

APA: see p. 15; all the APA here are $\mathbb{N}$-APA

▶ **Definition 5** (Finite-monoid affine Parikh automaton)**.** An affine Parikh automaton (A, U, C) is said to be *finite-monoid* if $\mathscr{M}(U)$ is finite; this is not the general case.

One may see APA as automata equipped with d counters $c_1, \ldots, c_d$, and each transition computing some action $c_i \leftarrow k_i + \sum_j a_{i,j} \times c_j$ on the d counters. One interesting class of finite-monoid APA, and the one we will focus on, is when *no sum of counter is allowed*, i.e., when all $a_{i,j}$ are either 0 or 1, and if $a_{i,j}$ is 1, then for all $j' \neq j$, $a_{i,j'}$ is 0.

▶ **Lemma 48.** *Any ε-CA (A, C) having the constraint-determinism property has the same language as a finite-monoid DetAPA (A', U, E) such that $L(A) = L(A')$.*

Proof. We outline the idea before giving the details. Let (A, C) be the ε-CA. We first apply the standard subset construction and obtain a deterministic automaton $\underline{A}$ equivalent to A. Consider a state $\underline{q}$ of $\underline{A}$. Suppose that after reading some word w leading $\underline{A}$ into state $\underline{q}$ we had, for each $q \in \underline{q}$, the Parikh image $\mathbf{c}_{w,q}$ (counting transitions in A, i.e., recording the occurrences of each transition in A) of *some* initial w-labeled path leading A into state q. Suppose that $\underline{q} \bullet\!\!-a\!\!\to \underline{r}$ is a transition in $\underline{A}$. How can we compute, for each $r \in \underline{r}$, the Parikh image $\mathbf{c}_{wa,r}$ of *some* initial wa-labeled path leading A into r? It suffices to pick any $q \in \underline{q}$ for which some a-labeled path leads A from q to r (possibly using the ε-transitions in A) and to add to $\mathbf{c}_{w,q}$ the contribution of this a-labeled path. A DetAPA transition on a is well-suited to mimic this computation, since an affine transformation can first "flip" the current Parikh q-count tuple "over" to the Parikh r-count tuple and then add to it the q-to-r contribution. Hence a DetAPA $(\underline{A}, \cdot, \cdot)$ upon reading a word w leading to its state $\underline{q}$ is able to keep track, for each $q \in \underline{q}$, of the Parikh image of *some* initial w-labeled path leading A into q. We need constraint-determinism only to reach the final conclusion: if a word w leads $\underline{A}$ into a final state $\underline{q}$, then some $q \in \underline{q}$ is final in A, and because of constraint-determinism, imposing membership in C for the Parikh image of the particular initial w-labeled path leading A to q kept track of by the DetAPA is as good as imposing membership in C for the Parikh image of any other initial w-labeled path leading A to q.

We now give the details. Say $A = (Q, \Sigma, \delta, q_0, F)$, and let us identify Q with $\{1, \ldots, |Q|\}$. For $p, q \in Q$ and $a \in \Sigma$ define $p \overset{a}{\rightsquigarrow} q$ to be a shortest path from p to q labeled by a — lexicographically smallest among shortest paths, for definiteness, as its length can be greater than one because of ε-transitions, — or $\perp$ if none exists. Let $\underline{A} = (2^Q, \Sigma, \underline{\delta}, \underline{q_0}, \underline{F})$ be the deterministic version of A defined by $\underline{q_0} = \{q_0\}$, $\underline{F} = \{\underline{q} \mid \underline{q} \cap F \neq \varnothing\}$, and:

$$\underline{\delta} = \{\underline{p} \bullet\!\!-a\!\!\to \underline{q} \mid q \in \underline{q} \leftrightarrow (\exists p \in \underline{p})[p \overset{a}{\rightsquigarrow} q \neq \perp]\}.$$

We have that $L(\underline{A}) = L(A)$. Note that, by construction, for any path π in A from q_0 to q, there exists a path $\underline{\pi}$ in $\underline{A}$ from $\underline{q_0}$ to a state $\underline{q}$ such that $q \in \underline{q}$ and $\mu_A(\pi) = \mu_{\underline{A}}(\underline{\pi})$.

We now attach an affine function to each transition of $\underline{A}$, where the functions are of dimension $(|Q|.|\delta|+1)$. We first define $V : \underline{\delta}^* \to \mathscr{F}_{|Q|.|\delta|}$, and will later add the extra component. We write V_π for $V(\pi)$. The intuition is as follows. Consider a path $\underline{\pi}$ on $\underline{A}$ from the initial state to a state $\underline{q}$ — the empty path is considered to be from $\underline{q_0}$ to $\underline{q_0}$. We view $V_{\underline{\pi}}(\mathbf{0})$ as a list of counters $(\mathbf{c}_1, \dots, \mathbf{c}_{|Q|})$ where $\mathbf{c}_q \in \mathbb{N}^{|\delta|}$. We will ensure that for any $q \in \underline{q}$, $\mathbf{c}_q$ is the Parikh image of a path π in A from q_0 to q such that $\mu_A(\pi) = \mu_{\underline{A}}(\underline{\pi})$. If two such paths π_1 and π_2 exist, we may choose one arbitrarily, as they are equivalent in the following sense: if ρ is such that $\pi_1\rho \in \mathsf{Run}(A)$ and $\mathsf{Pkh}(\pi_1\rho) \in C$, then the same holds for π_2.

For $\underline{p} \subseteq Q$, $q \in Q$, and $a \in \Sigma$, let $P(\underline{p}, q, a)$ be the smallest $p \in \underline{p}$ such that $p \overset{a}{\leadsto} q \neq \bot$ (we will consider only cases where at least one such p exists). Let $\underline{t} = \underline{p}\bullet\!\!-a\!\!\to\underline{q}$ be a transition of $\underline{A}$. We define $V_{\underline{t}}$ such that for $q \in \mathbf{q}$ and $p = P(\mathbf{p}, q, a)$, the application of $V_{\underline{t}}$ sets $\mathbf{c}_q$ to $\mathbf{c}_p + \mathsf{Pkh}(p \overset{a}{\leadsto} q)$. Formally:

$$V_{\underline{t}} = \left(\sum_{q \in \underline{q}} M\big(P(\underline{p}, q, a),\ q\big), \qquad \sum_{q \in \underline{q}} N\big(q,\ \mathsf{Pkh}(P(\underline{p}, q, a) \overset{a}{\leadsto} q)\big) \right)$$

where $M(p, q)$ is the matrix which transfers the p-th counter to the q-th, and zeroes the others, and $N(q, \mathbf{d})$ is the shift of $\mathbf{d} \in \mathbb{N}^{|\delta|}$ to the q-th counter. More precisely, $M(p, q)_{i,j} = 1$ iff there exists $1 \leq e \leq |\delta|$ such that $i = (q-1).|\delta| + e$ and $j = (p-1).|\delta| + e$; likewise, $N(q, \mathbf{d}) = (0^{(q-1).|\delta|})\cdot\mathbf{d}\cdot(0^{(|Q|-q).|\delta|})$. The matrices appearing in V are 0-1 matrices with at most one nonzero entry per row; composing such matrices preserves this property, thus $\mathscr{M}(V)$ is finite.

Fact 49. *Let $\underline{\pi}$ be a path on $\underline{A}$ from $\underline{q_0}$ to some state $\underline{q}$. Let $(\mathbf{c}_1, \dots, \mathbf{c}_{|Q|}) = V_{\underline{\pi}}(\mathbf{0})$, where $\mathbf{c}_q \in \mathbb{N}^{|\delta|}$. Then for all $q \in \underline{q}$, $\mathbf{c}_q$ is the Parikh image of a path in A from q_0 to q labeled by $\mu(\underline{\pi})$.*

We show Fact 49 by induction. If $|\underline{\pi}| = 0$, then $\mathsf{To}(\pi) = \{q_0\}$ and $\mathbf{c}_{q_0}$ is by definition all-zero. Thus $\mathbf{c}_{q_0}$ is the Parikh image of the empty path from and to q_0 in A. Let $\underline{\pi}$ be such that $|\underline{\pi}| > 0$, and consider a state $q \in \mathsf{To}(\underline{\pi})$. Write $\underline{\pi} = \underline{\rho} \cdot \underline{t}$, with $\underline{t} \in \underline{\delta}$, and let $p = P(\mathsf{To}(\underline{\rho}), q, \mu(t))$ and $\zeta = p \overset{\mu(t)}{\leadsto} q$. The induction hypothesis asserts that the p-th counter of $V_{\underline{\rho}}(\mathbf{0})$ is the Parikh image of a path ρ on A from q_0 to p labeled by $\mu(\underline{\rho})$. Thus, the q-th counter of $V_{\underline{\pi}}(\mathbf{0})$ is $\mathsf{Pkh}(\rho) + \mathsf{Pkh}(\zeta)$, which is the Parikh image of $\rho\zeta$, a path from q_0 to q labeled by $\mu(\underline{\pi})$. This concludes the proof of Fact 49.

We now define $U : \underline{\delta}^* \to \mathscr{F}_{|Q|.|\delta|+1}$. We add a component to the functions of V, such that for $\underline{\pi} \in \mathsf{Run}(\underline{A})$, the last component of $U_{\underline{\pi}}(\mathbf{0})$ is 0 if $\mathsf{To}(\underline{\pi}) \cap F = \varnothing$ and

$\min(\mathsf{To}(\underline{\pi}) \cap F)$ otherwise. For $\underline{t} = \underline{p} \bullet\!\!-\! a \!\to \underline{q} \in \underline{\delta}$, let:

$$U_{\underline{t}} : (\mathbf{x}, s) \mapsto \left(V_{\underline{t}}(\mathbf{x}), \quad \begin{cases} q & \text{if } q \text{ is the smallest s.t. } q \in \underline{q} \cap F, \\ 0 & \text{if no such } q \text{ exists.} \end{cases} \right)$$

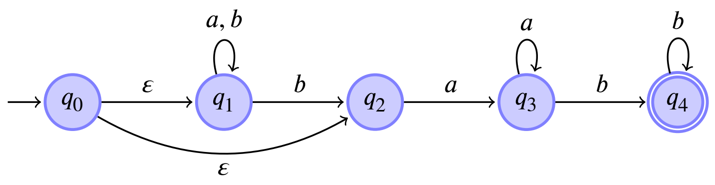
Now define $E \subseteq \mathbb{N}^{|Q|.|\delta|+1}$ to be such that $(\mathbf{v}_1, \ldots, \mathbf{v}_{|Q|}, q) \in E$ iff $\mathbf{v}_q \in C$; E is semilinear. We adjoin $\mathbf{0}$ to E iff $\mathbf{0} \in C$, in order to deal with the empty word. Now, by Fact 49, a word w is accepted by the DetAPA $(\underline{A}, U, E)$ iff there exists a path in A from q_0 to $q \in F$, labeled by w, and whose Parikh image belongs to C, i.e., $w \in L(A, C)$.

Finally, recall that $L(\underline{A}) = L(A)$ and note that $\mathscr{M}(U)$ is finite as $\mathscr{M}(V)$ is: the extra component of U only adds a column and a row of 0's to the matrices.

We note, for completeness, that constraint-deterministic CA, and thus finite-monoid DetAPA, strictly generalize DetPA. Indeed, the language:

$$\{a, b\}^* \cdot \{a^n b^n \mid n \in \mathbb{N}^+\}$$

is not expressible by a DetPA (see Paper I, Lemma 16), but is expressible as the constraint-deterministic CA (A, C) where A is:



and C constrains the two loops on q_3 and q_4 to occur the same number of times. As any word in $\{a, b\}^*$ has at most one accepting path in A, this PA is constraint-deterministic.

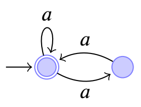
4.2 From finite-monoid DetAPA of bounded language to DetPA

Let us first recall the following classical result:

▶ **Lemma 50** (e.g., [Gin66, Lemmata 5.5.1 and 5.5.4]). *Let $u, v \in \Sigma^*$. Then $(u+v)^*$ is bounded iff there exists $z \in \Sigma^*$ such that $u, v \in z^*$.*

Example 16

Consider the automaton:



Its language is bounded while $\mathsf{Run}(A)$ is not.

We will need the following technical lemma. Bounded languages being closed under morphism, for all automata A if $\mathsf{Run}(A)$ is bounded then so is $L(A)$. The converse is true when A is deterministic (and false otherwise):

▶ **Lemma 51.** *Let A be a deterministic automaton for a bounded language, then* $\mathsf{Run}(A)$ *is bounded. Moreover,* $\mathsf{Run}(A)$ *is expressible as a SLRE whose branches are of the form $\rho_1\pi_1^* \cdots \rho_n\pi_n^*\rho_{n+1}$ where $\rho_i \neq \varepsilon$ for all $1 < i \leq n$ and the first transition of π_i differs from that of ρ_{i+1} for every i (including $i = n$ if $\rho_{n+1} \neq \varepsilon$).*

Proof. Recall that bounded languages are closed under *deterministic* rational transduction (see, e.g., [Gin66, Lemma 5.5.3]). Let a deterministic automaton $A = (Q, \Sigma, \delta, q_0, F)$ accept a bounded language and define the automaton A' as a copy of A over the alphabet $\Sigma \times \delta$ where a transition t is relabeled $(\mu(t), t)$. Then $\tau_{A'}$, the deterministic rational transduction defined by A', is such that $\mathsf{Run}(A) = \tau_{A'}(L(A))$, and thus $\mathsf{Run}(A)$ is bounded.

It will be useful to note the claim that if $X\pi_1^*\pi_2^*Y \subseteq \mathsf{Run}(A)$ for some nonempty paths π_1, π_2 and some bounded languages X and Y, then for some path π, $X\pi_1^*\pi_2^*Y \subseteq X\pi^*Y \subseteq \mathsf{Run}(A)$. To see this, note that if $X\pi_1^*\pi_2^*Y \subseteq \mathsf{Run}(A)$ then $X(\pi_1 + \pi_2)^*Y \subseteq \mathsf{Run}(A)$ because π_1 and π_2 are loops on a same state. Now $X(\pi_1 + \pi_2)^*Y$ is bounded because $\mathsf{Run}(A)$ is bounded, hence $(\pi_1 + \pi_2)^*$ is bounded. So pick π such that $\pi_1, \pi_2 \in \pi^*$ (by Lemma 50). Then $X(\pi_1 + \pi_2)^*Y \subseteq X\pi^*Y$. But π is a loop in A because $\pi_1 = \pi^j$ for some $j > 0$ is a loop so that $\mathsf{From}(\pi) = \mathsf{To}(\pi)$ in A. Hence $X\pi^*Y \subseteq \mathsf{Run}(A)$. Thus $X\pi_1^*\pi_2^*Y \subseteq X(\pi_1 + \pi_2)^*Y \subseteq X\pi^*Y \subseteq \mathsf{Run}(A)$.

Let E be a SLRE for $\mathsf{Run}(A)$, and consider one branch $P = \rho_1\pi_1^* \cdots \rho_n\pi_n^*\rho_{n+1}$ of E. We assume n to be minimal among the set of all n' such that $\rho_1\pi_1^* \cdots \rho_n\pi_n^*\rho_{n+1} \subseteq \rho_1'\pi_1'^* \cdots \rho_{n'}'\pi_{n'}'^*\rho_{n'+1}' \subseteq \mathsf{Run}(A)$ for some paths $\rho_1', \pi_1', \ldots, \rho_{n'}', \pi_{n'}', \rho_{n'+1}'$.

First we do the following for $i = n, n-1, \ldots, 1$ in that order. If $\pi_i = \zeta\pi$ and $\rho_{i+1} = \zeta\rho$ for some maximal nonempty path ζ and for some paths π and ρ, we rewrite $\rho_i\pi_i^*\rho_{i+1}$ as $\rho_i'\pi_i'^*\rho_{i+1}'$ by letting $\rho_i' = (\rho_i\zeta)$, $\pi_i' = (\pi\zeta)$ and $\rho_{i+1}' = \rho$. This leaves the language of P unchanged and ensures at the ith stage that the first transition of π_j' (if any) differs from that of ρ_{j+1}' (if any) for $i \leq j \leq n$. Note that n has not changed.

Let $\rho_1'\pi_1'^* \cdots \rho_n'\pi_n'^*\rho_{n+1}'$ be the expression for P resulting from the above process. By the minimality of n, $\pi_i' \neq \varepsilon$ for $1 \leq i \leq n$. And for the same reason, $\rho_i' \neq \varepsilon$ for $1 < i \leq n$, since $\rho_i' = \varepsilon$ implies $X\pi_{i-1}'^*\pi_i'^*Y \subseteq Xz^*Y \subseteq \mathsf{Run}(A)$ for some z by the claim above, where $X = \rho_1' \cdots \pi_{i-2}'^*\rho_{i-1}'$ and $Y = \rho_{i+1}'\pi_{i+1}'^* \cdots \rho_{n+1}'$ are bounded languages.

We are now ready to show the result of this section:

▶ **Lemma 52.** *Let (A, U, C) be a finite-monoid DetAPA such that $L(A)$ is bounded. Then there exist a finite number of rflat DetCA having $L(A, U, C)$ as the union of their languages.*

Proof. Let $A = (Q_A, \Sigma, \delta, q_0, F_A)$ be a deterministic automaton whose language is bounded, let $U : \delta^* \to \mathscr{F}_d$ for some $d > 0$ be a morphism such that $\mathscr{M}(U)$ is finite, and let $C \subseteq \mathbb{N}^d$ be a semilinear set.

By the finiteness of $\mathscr{M}(U)$, every $M \in \mathscr{M}(U)$ has a minimum $threshold$ and a strictly positive minimum $period$ such that $M^{threshold+period} = M^{threshold}$. Let

$$0 < s = \max\{threshold(M) : M \in \mathscr{M}(U)\} + 1 \text{ and}$$
$$0 < p = \operatorname{lcm}\{period(M) : M \in \mathscr{M}(U)\} \ .$$

Then every $M \in \mathscr{M}(U)$ verifies $M^{j+p} = M^j$ for every $j \geq s$.

Our main task will be to show the following:

Fact 53. *For $n \geq 0$, $\rho_1, \pi_1, \ldots, \rho_n, \pi_n, \rho_{n+1} \in \delta^*$ satisfying the hypothesis of Lemma 51 and $s \leq j_1, \ldots, j_n < s + p$, there is an automaton D such that (A, C) is a rflat DetPA with:*

1. *The initial state of D has no incoming transition and at most one outgoing transition, which is labeled by the first transition of ρ_1 if $\rho_1 \neq \varepsilon$,*
2. *$\Psi(L(D))$ is $\rho_1 \pi_1^{j_1} (\pi_1^p)^* \rho_2 \pi_2^{j_2} (\pi_2^p)^* \cdots \rho_n \pi_n^{j_n} (\pi_n^p)^* \rho_{n+1}$,*
3. *$\forall \omega \in L(D), \Phi(\omega) = U_{\Psi(\omega)}(\mathbf{0})$.*

We first show how Fact 53 implies the result. Consider the set $\mathsf{Run}(A)$ of accepting paths in A; it is, by Lemma 51, a bounded language. Let P be the language defined a branch $\rho_1 \pi_1^* \cdots \rho_n \pi_n^* \rho_{n+1}$ of the SLRE for $\mathsf{Run}(A)$ given by Lemma 51. For $0 \leq j_1, \ldots, j_n < s + p$, define:

$$P_{(j_1,\ldots,j_n)} = \rho_1 \pi_1^{j_1} (\pi_1^p)^* \cdots \rho_n \pi_n^{j_n} (\pi_n^p)^* \rho_{n+1}.$$

Then P can be (redundantly) described as:

$$P = \bigcup_{0 \leq j_1, \ldots, j_n < s+p} P_{(j_1,\ldots,j_n)}.$$

Now, for some $0 \leq j_1, \ldots, j_n < s + p$, we argue that $\{\pi \in P_{(j_1,\ldots,j_n)} \mid U_\pi(\mathbf{0}) \in C\}$ is the union of the languages of some rflat DetPA. If each j_i is greater than s, then this is the statement of Fact 53. Otherwise, if:

$$P_{(j_1,\ldots,j_n)} = \underbrace{\rho_1 \cdots}_{\alpha} \rho_i \pi_i^{j_i} (\pi_i^p)^* \rho_{i+1} \underbrace{\cdots \rho_{n+1}}_{\beta} \ ,$$

with $j_i < s$, it can be expressed as $\alpha\rho_i\pi_i^{j_i+mp}(\pi_i^p)^*\rho_{i+1}\beta$ together with

$$\bigcup_{\ell=0}^{m-1} \alpha\rho_i\pi_i^{j_i+\ell p}\rho_{i+1}\beta \ , \tag{4}$$

where $m = \min\{\ell : j_i + \ell p \geq s\}$. Now $\alpha\rho_i\pi_i^{j_i+\ell p}\rho_{i+1}\beta$ can be rewritten as $\alpha\rho_i'\beta$, with the first transition of π_{i-1} still different from the first transition of $\rho_i' = \rho_i\pi_i^{j_i+\ell p}\rho_{i+1}$. Instances of $j_{i'} < s$ occurring in $\alpha\rho_{i-1}'\beta$ for $i' \neq i$ can be rewritten as well. When all occurrences of $j_i < s$ have been processed in this way, each resulting language is the language of a rflat DetPA by Fact 53.

Now $\{\pi \in P \mid U_\pi(\mathbf{0}) \in C\}$ is thus the union of the languages of rflat DetPA (D, C). Now define D' as the automaton D where a label $(t, \mathbf{u})$ in D appears as $(\mu_A(t), \mathbf{u})$ in D', we argue that (D', C) is still a rflat DetPA: for any two transitions $q \bullet\!\!-(a, \mathbf{u})\!\!\rightarrow q'$ and $q \bullet\!\!-(a, \mathbf{v})\!\!\rightarrow q''$ in D', there are two transitions $q \bullet\!\!-(t, \mathbf{u})\!\!\rightarrow q'$ and $q \bullet\!\!-(t', \mathbf{v})\!\!\rightarrow q''$ with $\mu_A(t) = \mu_A(t')$ in D. Since t and t' may appear after the same prefix in $\Psi(L(D))$ (as any state of a rflat automaton is both accessible and co-accessible) and as $\Psi(L(D))$ is a set of paths, this implies that $\mathsf{From}(t) = \mathsf{From}(t')$. In turn, as A is deterministic, this implies that $t = t'$, and thus, as (D, C) is a DetPA, that $\mathbf{u} = \mathbf{v}$ and $q' = q''$, i.e., the two transitions we considered in D' are the same, hence (D', C) is a DetPA. Moreover, since D is rflat and D' has the same graph as D, D' is rflat. Finally, $L(D', C) = \mu(L(D, C))$, and thus $\{\mu(\pi) \mid \pi \in P \wedge U_\pi(\mathbf{0}) \in C\}$ is the union of rflat DetPA languages. Going through all the branches thus leads to a finite set of rflat DetPA languages, with $L(A, U, C)$ as their union.

We now prove Fact 53 by induction on n, the number of π_i's. For succinctness, we construct automata where the labels are pairs $(\pi, \mathbf{v})$ where $\pi = t_1 t_2 \cdots t_k$ is a nonempty word over δ; this is to be understood as a string of transitions with fresh states in between, with the first transition labeled $(t_1, \mathbf{v})$ and the other ones $(t_i, \mathbf{0})$, $i \geq 2$.

Suppose $n = 0$, then we are only given ρ_1. If $\rho_1 = \varepsilon$, then D is a single initial and final state. Otherwise, D is an automaton with states $\{q_0, q_f\}$, q_0 initial and q_f final, with a single transition, between q_0 and q_f, labeled $(\rho_1, U_{\rho_1}(\mathbf{0}))$. This verifies the conclusions of Fact 53.

Suppose $n > 0$. We introduce a few notations: for a path $\pi = t_1 t_2 \cdots t_k \in \delta^*$, we write M_π for $M_{t_k} \ldots M_{t_2} M_{t_1}$ and Δ_π for $U_\pi(\mathbf{0})$. Note that $U_\pi = (M_\pi, \Delta_\pi)$. Let $\pi \in \rho_1\pi_1^{j_1}(\pi_1^p)^*\rho_2\pi_2^{j_2}(\pi_2^p)^* \cdots \rho_n\pi_n^{j_n}(\pi_n^p)^*\rho_{n+1}$, and write $\pi = \rho_1\pi_1^{j_1}\pi_1^{pk}\gamma$ with k maximal. Define $M = M_{\rho_2\pi_2^{j_2}\cdots\rho_n\pi_n^{j_n}\rho_{n+1}}$, and note that $M_\gamma = M$, by the finite-monoid property. Then:

$$\begin{aligned}
\Delta_\pi &= U_\gamma(U_{\pi_1^{j_1}}(U_{\pi_1^{pk}}(U_{\rho_1}(\mathbf{0})))) \\
&= M(M_{\pi_1^{j_1}}.U_{\pi_1^{pk}}(\Delta_{\rho_1}) + \Delta_{\pi_1^{j_1}}) + \Delta_\gamma && \text{(as } M_\gamma = M\text{)} \\
&= M.\Delta_{\pi_1^{j_1}} + M.M_{\pi_1^{j_1}}(M_{\pi_1^p}.U_{\pi_1^{p(k-1)}}(\Delta_{\rho_1}) + \Delta_{\pi_1^p}) + \Delta_\gamma \\
&= M.\Delta_{\pi_1^{j_1}} + M.M_{\pi_1^{j_1}}(U_{\pi_1^{p(k-1)}}(\Delta_{\rho_1}) + \Delta_{\pi_1^p}) + \Delta_\gamma && \text{(as } M_{\pi_1^{j_1}}.M_{\pi_1^p} = M_{\pi_1^{j_1}}\text{)} \\
&\quad\vdots && \text{(repeating the two previous lines)} \\
&= M.\Delta_{\pi_1^{j_1}} + M.M_{\pi_1^{j_1}}(\Delta_{\rho_1} + k.\Delta_{\pi_1^p}) + \Delta_\gamma \\
&= \underbrace{M.\Delta_{\pi_1^{j_1}} + M.M_{\pi_1^{j_1}}.\Delta_{\rho_1}}_{K} + k.\underbrace{M.M_{\pi_1^{j_1}}.\Delta_{\pi_1^p}}_{K'} + \Delta_\gamma\ .
\end{aligned}$$

Note that the values of K and K' are independent of γ and k. Now construct D' as the automaton with states $\{q_0, q_f\}$, a transition between q_0 and q_f labeled $(\rho_1\pi_1^{j_1}, K)$, and a transition from and to q_f labeled (π_1^p, K'). Let D'' be the automaton given by the induction hypothesis on the parameters $\rho_2, \pi_2, \ldots, \rho_n, \pi_n, \rho_{n+1}, j_2, \ldots, j_n$. We construct D by merging D' and D'': we set q_0 initial and identify q_f with the initial state of D''. Note that (D, C) is indeed a flat DetPA: by induction hypothesis there is no cycle on the initial state of D'' and D'' is rflat, thus D is rflat; moreover, since D'' either has an empty language (if $\rho_2 = \varepsilon$, and thus $n = 1$) or starts with the first transition of ρ_2, which differs from that of π_1, (D, C) is a DetPA.

We argue that D fulfills the conclusions of Fact 53. Point (1) is clear. Point (2) is verified thanks to the induction hypothesis: the projection of the language of D' is indeed $\rho_1\pi_1^{j_1}(\pi_1^p)^*$, and that of D'' is $\rho_2 \cdots \rho_{n+1}$. Finally, for (3): let $\omega \in L(D)$, and write $\omega = \omega_1\omega_2^k\omega_3$ with $\Psi(\omega_1) = \rho_1\pi_1^{j_1}$, $\Psi(\omega_2) = \pi_1^p$, and k maximal. Note that ω_3 is read over D''. Then:

$$\begin{aligned}
\Phi(\omega) = \Phi(\omega_1) + k.\Phi(\omega_2) + \Phi(\omega_3) &= K + k.K' + \Phi(\omega_3) \\
&= K + k.K' + U_{\Psi(\omega_3)}(\mathbf{0}) && \text{(by induction hypothesis)} \\
&= K + k.K' + \Delta_{\Psi(\omega_3)} = \Delta_{\Psi(\omega)} = U_{\Psi(\omega)}(\mathbf{0})\ .
\end{aligned}$$

This concludes the proof of Fact 53.

Note

In Paper IV, Cor. 98, we show a stronger result: there are *unary* languages in $\mathscr{L}_{\text{DetAPA}} \setminus \mathscr{L}_{\text{PA}}$

We note that there exist bounded languages with nonsemilinear Parikh image in $\mathscr{L}_{\text{DetAPA}}$ (e.g., $\{a^n b^{2^n}\}$), thus there exist bounded languages in $\mathscr{L}_{\text{DetAPA}} \setminus \mathscr{L}_{\text{PA}}$.

4.3 Proof of Theorem 46 and effectiveness

Proof of Theorem 46. Let $Y \in \mathscr{L}_{\text{BoundedPA}}$. By Theorem 41, $Y \in$ BSL, so let $w_1, \ldots, w_n$ be a socle of Y with $C = \mathsf{Iter}_{(w_1,\ldots,w_n)}(L)$ semilinear. Let (A, E) be the canonical ε-CA for $w_1, \ldots, w_n$ subject to C obtained by applying Construction 40. By construction, $L(A)$ is bounded. Moreover, (A, E) is constraint-deterministic: if two accepting paths π_1 and π_2 in A have the same label w, then π_1 and π_2 describe two ways to iterate the words in the socle of Y to get w. As the semilinear set C describes *all* ways to iterate these words to get a specific label, $\mathsf{Pkh}(\pi_1) \in E$ iff $\mathsf{Pkh}(\pi_2) \in E$.

Now applying Lemma 48 to the ε-CA (A, E) yields a finite-monoid DetAPA for Y whose underlying automaton has the bounded language $L(A)$. In turn, Lemma 52 yields a finite number of rflat DetCA having Y as the union of their languages.

We note that all the constructions are effective, in the sense that given words $w_1, \ldots, w_n$ and a semilinear set $C \subseteq \mathbb{N}^n$, we can construct a DetCA for the language $\{w_1^{i_1} \cdots w_n^{i_n} \mid (i_1, \ldots, i_n) \in C\}$. We leave open whether there is an effective procedure to determinize a PA when the promise is made that its language is bounded.

5 Discussion and further work

We showed that PA and DetPA recognize the same class of bounded languages, namely BSL. To this end, we used related models (e.g., APA) and provided expressiveness results of independent interest (e.g., related to constraint-determinism and the finite-monoid property). Moreover, we noted that the union of rflat DetCA is a concept that has already been defined, in the context of model-checking, as 1-CQDD [BH99], showing that 1-CQDD capture exactly BSL. The closure properties observed in Corollaries 42 and 43 also apply to 1-CQDD, thus providing alternative proofs to those appearing in [BH99]. In particular, given $\mathscr{L}_{\text{1-CQDD}} =$ BSL, the proof of the closure of 1-CQDD under concatenation is a consequence of the simple proofs of closure of $\mathscr{L}_{\text{PA}}$ and BOUNDED under concatenation, thus avoiding the long and technical proof of [BH99].

A related model, *reversal-bounded multi-counter machines (RBCM)* [Iba78], has been shown to have the same expressive power as PA [KR03]. It is known that *one-way* deterministic RBCM are strictly more powerful than DetPA (see Paper I), thus our result carries over to RBCM, showing that RBCM and one-way deterministic RBCM recognize the same class of bounded languages, namely BSL. This provides an alter-

RBCM: see Chap. 0, Sec. 3.4, p. 16

native proof of the same fact appearing in a recent paper of Ibarra and Seki [IS11], and yields as a by-product a characterization of BSL using a model provably weaker than one-way deterministic RBCM.

Further work includes an in-depth study of the finite-monoid property of APA. In particular, we suspect that finite-monoid APA are no more expressive than PA, and that finite-monoid DetAPA are no more expressive than constraint-deterministic CA. One further avenue of research is to investigate the related decision problems, e.g., is it decidable whether the language of a PA is bounded? or whether it is that of a constraint-deterministic CA?

Acknowledgments

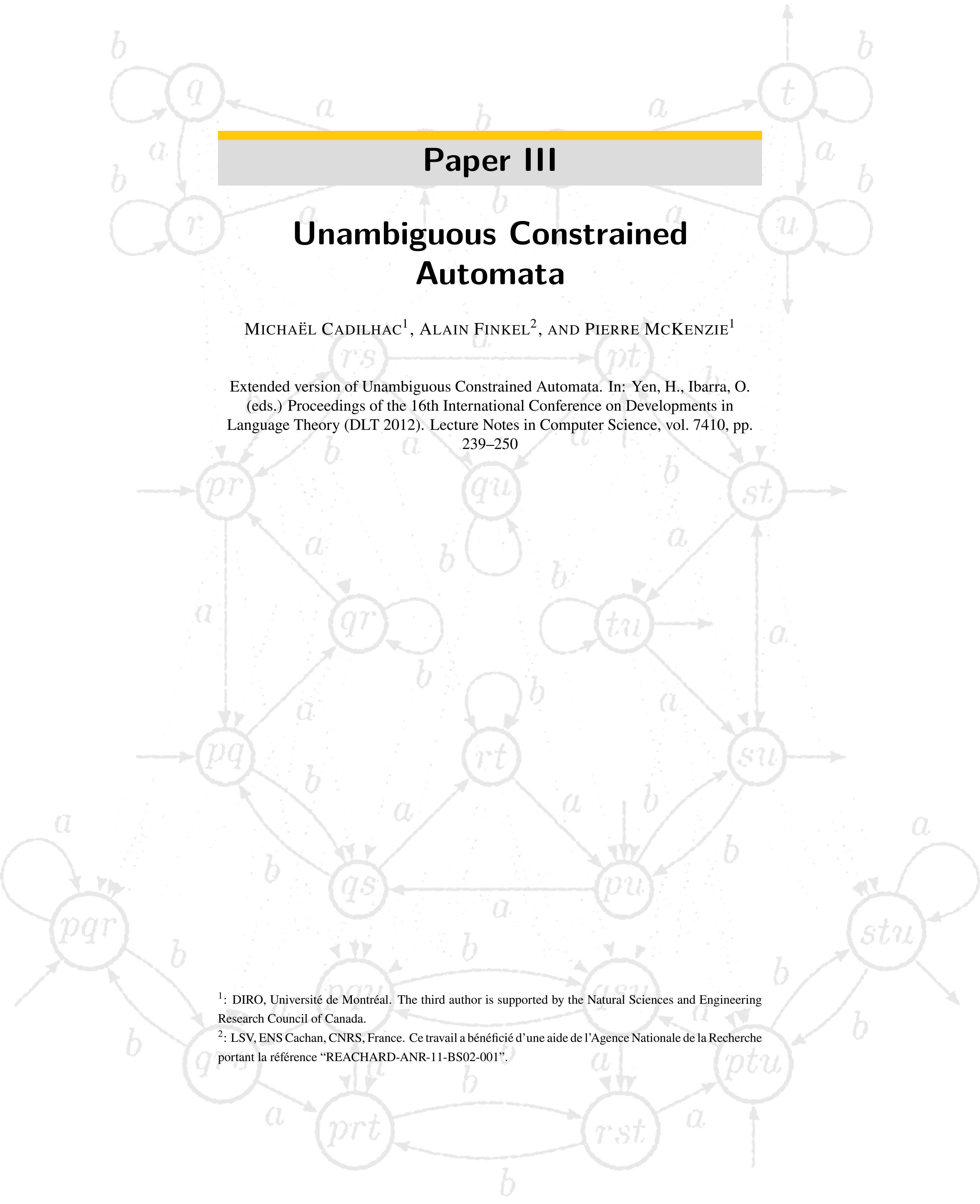
The first author thanks Laurent Beaudou, Marc Kaplan, and Adrien Lemaître for stimulating discussions and comments on early versions of this paper. We further thank the anonymous referees for their critical comments and especially for pointing out a flawed claim made in an earlier version of this work, namely that finite-monoid DetAPA are equivalent to DetPA.

Discussion

The first goal of this research was to study in which cases the statement "finite-monoid DetAPA are equivalent to DetCA" is true. The finite nature of the affine transformations indeed indicates that they can be simulated using the operations available to the DetCA; in particular, a DetAPA for which all the matrices used to define the affine functions are the identity is a DetCA. The limit of this parallel is drawn with the example on page 80: there is a finite-monoid DetAPA expressing a language not in $\mathscr{L}_{\text{DetPA}}$. However, we identified a possible candidate for an equivalent CA formalism of finite-monoid DetAPA: constraint-deterministic CA. In Paper III, we show that this equivalence holds.

Our interest in bounded languages is also related to their use in model-checking and we hope further development will be done in this field. This was the motivation behind the introduction of CQDD by Bouajjani and Habermehl [BH99], and more investigation has to be made to evaluate the impact of the main result of our paper in this context. In particular, it is interesting to consider the following open question: given a CA, can we decide whether its language is bounded, and if so, can we find a socle for it? If we can do so, then we can determinize the CA, and thus use, for instance, the closure properties of DetCA to allow for more property-testing.

Finally, we had the intuition that bounded languages would help in solving the regularity problem for DetCA, and the study of $\mathscr{L}_{\text{BoundedPA}}$ thus was the natural thing to carry. As it turns out, bounded languages were indeed of a great help for showing the decidability of regularity (see Paper III, Section 4) and constraint-deterministic CA were of interest in this work, but the main result of the present paper, namely, $\text{BSL} = \mathscr{L}_{\text{BoundedPA}} = \mathscr{L}_{\text{BoundedDetPA}}$, was not needed.



Paper III

Unambiguous Constrained Automata

MICHAËL CADILHAC[1], ALAIN FINKEL[2], AND PIERRE MCKENZIE[1]

Extended version of Unambiguous Constrained Automata. In: Yen, H., Ibarra, O. (eds.) Proceedings of the 16th International Conference on Developments in Language Theory (DLT 2012). Lecture Notes in Computer Science, vol. 7410, pp. 239–250

[1]: DIRO, Université de Montréal. The third author is supported by the Natural Sciences and Engineering Research Council of Canada.
[2]: LSV, ENS Cachan, CNRS, France. Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence "REACHARD-ANR-11-BS02-001".

Presentation

When introducing a model of computation, we are interested in getting a balance between expressiveness and good closure and decidability properties. DetCA, while offering a good expressiveness, cannot express languages specified by a semilinear constraint on the *end* of the word; for instance, the language $\{a, b\}^*\{a^n b^n \mid n \geq 1\}$ is not in $\mathscr{L}_{\text{DetCA}}$ (see Paper I). A natural compromise between determinism and nondeterminism, on which Colcombet [Col12] has recently made a survey, is *unambiguity*. An automaton is said to be unambiguous if there is at most one accepting path per word; in turn, a CA is said to be unambiguous if the underlying automaton is. In his survey, Colcombet argues for the usefulness of unambiguity in multiple contexts (performance, expressiveness, decidability, …), and proposes several open questions. The goal of the study of the present paper is to see how unambiguity affects CA. In particular, Paper II introduced the concept of *constraint-deterministic* CA, and it is shown here that it is equivalent to the concept of unambiguous CA, thus continuing the investigation of finite-monoid APA.

One of the main open problems we left at the end of Paper I is whether regularity is decidable for DetCA. In the present paper, we solve this problem for the provably larger class of unambiguous CA.

Personal contribution. I proposed the results, proofs, and ideas of this paper. Colcombet's survey [Col12] was pointed to us by Andreas Krebs, and we used the terminology of this article in the present work. Andreas Krebs also helped in simplifying the proof of Lemma 73 and gave useful comments on the general structure of the paper. McKenzie helped in refining the proofs, in particular that of Theorem 75. The final form of this paper is a concerted work of the authors.

Unambiguous Constrained Automata

Abstract

The class of languages captured by Constrained Automata (CA) that are unambiguous is shown to possess more closure properties than the provably weaker class captured by deterministic CA. Problems decidable for deterministic CA are nonetheless shown to remain decidable for unambiguous CA, and testing for *regularity* is added to this set of decidable problems. Unambiguous CA are then shown incomparable with deterministic reversal-bounded machines in terms of expressivity, and a *deterministic* model equivalent to unambiguous CA is identified.

Introduction

A recent trend in automata theory is to study flavors of nondeterminism, which are introduced to provide a scale of expressiveness in different models (see [Col12] for a survey). The usual goal is to strike a balance between the expressiveness of nondeterministic models and the undecidability properties that often come with nondeterminism. A natural restriction to nondeterminism is *unambiguity*, i.e., the property that despite the underlying nondeterminism, there be at most one way to accept an input word. Within the context of finite automata, unambiguity and nondeterminism are equally expressive, but many open problems concerning the state complexity of unambiguity remain. Within more general contexts, the first question is often whether unambiguity offers more expressiveness than determinism; if so, then the examination of the closure and decidability properties of the new class often reveals that it inherits good properties. Another line of attack is to find a deterministic model equivalent to an unambiguous model, so as to understand how unambiguity affects a given model.

In [KR03], Klaedtke and Rueß studied Constrained Automata (CA),[7] a model whose expressive power lies between regular languages and context-sensitive languages (see Paper I). Klaedtke and Rueß successfully used the CA in the model-checking of hardware circuits, suggesting that CA is a model of interest for real-life applications. The deterministic variant (DetCA) of the CA enjoys more closure properties (e.g., complement) and decidability properties (e.g., universality) than the CA, but is unable to express languages as simple as $\{a, b\}^* \cdot \{a^n b^n \mid n \geq 1\}$ (see Paper I). Buoyed by Colcombet's recent systematic examination of unambiguity [Col12], here we initiate the study of unambiguous CA (UnCA).

We show that UnCA enjoy more closure properties than DetCA, while being more expressive. The class of languages UnCA defines is indeed closed under Boolean operations, inverse morphisms, commutative closure, reversal, and right and left quotient. We show that the problems known to be decidable for DetCA (emptiness, universality, finiteness, inclusion) remain decidable for UnCA. As the main technical result of this paper, we show that regularity is decidable for UnCA; by contrast, regularity is known to be undecidable for CA (see Paper I), while its status was unknown for DetCA. Finally, although DetCA are less powerful than UnCA, we present a natural *deterministic* model equivalent to UnCA; as a result of independent interest, we show that the nondeterministic variant of this model has the same expressive power as CA.

Section 1 in this paper contains preliminaries. Section 2 investigates the closure and expressiveness properties of UnCA. Section 3 compares UnCA and DetRBCM. Section 4 proceeds with the decidability properties of UnCA, showing, as our main result, that regularity is decidable. Section 5 shows that there is a natural equivalent deterministic model to UnCA. Section 6 concludes with a brief discussion.

1 Preliminaries

Let $s \geq 0$ and $p \geq 1$, we define the congruence $\equiv_{s,p}$, by $x \equiv_{s,p} y$ iff $(x = y < s) \vee (x, y \geq s \wedge x = y \pmod{p})$, for $x, y \in \mathbb{N}$; we write $[x]_{s,p}$ for the equivalence class of x under $\equiv_{s,p}$. We extend $\equiv_{s,p}$ component-wise to vectors $\mathbf{x}, \mathbf{y} \in \mathbb{N}^d$ by letting $\mathbf{x} \equiv_{s,p} \mathbf{y}$ iff $x_i \equiv_{s,p} y_i$ for all $1 \leq i \leq d$; similarly, $[\mathbf{x}]_{s,p}$ is the equivalence class of $\mathbf{x}$ under this relation.

Remark. It is decidable whether a given CA is deterministic or unambiguous.

[7] In [KR03], the model under study is called *Parikh automata*. CA are but an effectively equivalent model with an arguably simpler definition.

2 Closure properties and expressiveness of UnCA

In this section, we show closure and nonclosure properties, and we give languages witnessing the strict inclusion chain $\mathscr{L}_{\text{DetCA}} \subsetneq \mathscr{L}_{\text{UnCA}} \subsetneq \mathscr{L}_{\text{CA}}$. We start with a tool that will prove useful when combining UnCA:

CA, DetCA, UnCA: see p. 14

▶ **Lemma 54.** *For any UnCA (A, C), there is an UnCA (A', C') where A' has no ε-transition, $L(A) = L(A')$, and $L(A, C) = L(A', C')$.*

Proof. Let (A, C) be an UnCA with $A = (Q, \Sigma, \delta, q_0, F)$. We first note that for $p, q \in Q$, and $\ell \in \Sigma \cup \{\varepsilon\}$, we may suppose there is at most one way to reach q from p reading ℓ. Indeed, suppose there are two paths π_1, π_2 from p to q labeled ℓ, and suppose there is a path ρ_1 from q_0 to p (otherwise, we may remove p) and a path ρ_2 from q to a final state (otherwise, we may remove q). Then $\rho_1\pi_1\rho_2$ and $\rho_1\pi_2\rho_2$ are two accepting paths with the same label, contradicting the unambiguity of A. In particular, this implies that there is no cycle of ε-transitions, since if π is such a cycle, one may go from and to $\mathsf{From}(\pi)$ reading ε using *two* paths: π and the empty path.

In the same vein, we note that we may suppose that for a state q, if there is a path of ε-transitions from q to a final state q', then it is unique and there is no such path between q and a different final state.

Now we "backward-close" the ε-automaton A. For $p, q \in Q$ and $a \in \Sigma$, define $P(p, a, q)$ to be the only path from p to q labeled a which ends in a transition labeled a; if none exists, set $P(p, a, q)$ to $\perp$. Likewise, define $E(p)$ to be the unique path labeled ε from p to a final state of A, if it exists, and $\perp$ otherwise. Note that $E(p) = \varepsilon$ if $p \in F$. Define $A' = (Q, \Sigma, \delta', q_0, F')$ where:

$$\begin{aligned} \delta' &= \{p \bullet\!\!- a \!\to q \mid a \in \Sigma \wedge P(p, a, q) \neq \perp\} \ , \\ F' &= \{p \mid E(p) \neq \perp\} \ . \end{aligned}$$

Clearly, this automaton has the same language as A. Further, we argue that it is unambiguous. Let $h : \delta'^* \to \delta^*$ be the morphism defined by $h(p \bullet\!\!- a \!\to q) = P(p, a, q)$. We show that h is a bijection from $\mathsf{Run}(A')$ to:

$$\mathsf{Run}(A)^{-\varepsilon} = \mathsf{Run}(A)(\{t \in \delta \mid \mu(t) = \varepsilon\}^*)^{-1} \ ,$$

that is, the initial paths in A ending in a state from which we can reach a final state by following ε-transitions. First note that for a path $\pi' \in \delta'^*$, $h(\pi')$ is a path with the same label, origin, and destination as π'. This implies that if $\pi' \in \mathsf{Run}(A')$, then $h(\pi')$ is an initial path in A ending in a state q such that either $q \in F$ or there is a path of ε-transitions from q to a final state; in both cases, $h(\pi') \in \mathsf{Run}(A)^{-\varepsilon}$.

(h is onto.) Let $\pi \in \mathsf{Run}(A)^{-\varepsilon}$, and write $\pi = \pi_1\pi_2 \cdots \pi_n$ such that each π_i is a path having only one non ε transition, and it is the last transition of π_i. We have that $P(\mathsf{From}(\pi_i), \mu(\pi_i), \mathsf{To}(\pi_i)) = \pi_i$, so for all i, let $t_i = \mathsf{From}(\pi_i)\bullet\!\!-\mu(\pi_i)\!\!\rightarrow\mathsf{To}(\pi_i)$, and note that $t_i \in \delta'$ and $h(t_i) = \pi_i$. Thus $\pi' = t_1t_2 \cdots t_n$ is an initial path in A' such that $h(\pi') = \pi$. Now π (and thus π') ends in a state q such that there is a (possibly empty) ε-labeled path from q to a state in F, i.e., $\mathsf{To}(\pi') \in F'$, and thus $\pi' \in \mathsf{Run}(A)$.

(h is one-to-one.) If π_1 and π_2 are two paths of $\mathsf{Run}(A')$ with the different labels, then $h(\pi_1)$ and $h(\pi_2)$ are different, as they also are paths with different labels. We show that it is not possible for two different paths in $\mathsf{Run}(A')$ to have the same label. Suppose, for a contradiction, that $\pi_1, \pi_2 \in \mathsf{Run}(A')$ are such paths; note that they have the same (nonzero) length as A' has no ε-transition. Let $\rho_1 = E(\mathsf{To}(\pi_1))$ and $\rho_2 = E(\mathsf{To}(\pi_2))$. We have that $h(\pi_1)\rho_1$ and $h(\pi_2)\rho_2$ are accepting paths in A with the same label. Let $\pi_1 = \pi t_1\pi_1'$ and $\pi_2 = \pi t_2\pi_2'$ with $t_1 \neq t_2$ and $\pi, \pi_1, \pi_2 \in \delta'^*$; this split exists as $|\pi_1| = |\pi_2|$. Now $h(t_1)$ and $h(t_2)$ have only one transition not labeled ε, which appears at the end, and as $\mathsf{To}(t_1) \neq \mathsf{To}(t_2)$, those last transitions end in different states, and are thus different. Hence $h(t_1)$ and $h(t_2)$ are different and neither is the prefix of the other. This implies that $h(\pi_1)\rho_1 = h(\pi)h(t_1)h(\pi_1')\rho_1$ and $h(\pi_2)\rho_2 = h(\pi)h(t_2)h(\pi_2')\rho_2$ differ, and are two different accepting paths in A with the same label, contradicting the unambiguity of A. This concludes the proof that h is one-to-one.

Now note that there is a one-to-one correspondence that preserves labels between $\mathsf{Run}(A)$ and $\mathsf{Run}(A)^{-\varepsilon}$ that consists in removing the trailing ε-transitions. As h also preserves labels, this implies that A' is unambiguous and $L(A') = L(A)$. Moreover, with $\pi' \in \mathsf{Run}(A')$, the only path $\pi \in \mathsf{Run}(A)$ with the same label is given by $\pi = h(\pi')E(\mathsf{To}(\pi'))$. We thus define the constraint set C' so that $\mathsf{Pkh}(\pi') \in C'$ iff $\mathsf{Pkh}(\pi) \in C$. For this, we need to know, given the Parikh image of a run in A', in which state q the run ends, so that we can add $\mathsf{Pkh}(E(q))$ to retrieve the Parikh image of the similar path in A:

Fact 55. *Let A be an automaton. For each final state q of A, the set of Parikh images of paths in* $\mathsf{Run}(A)$ *ending in q is effectively semilinear. Moreover, those sets are disjoint.*

Proof sketch. First, we note that $\mathsf{Pkh}(\mathsf{Run}(A))$ is effectively semilinear. This is a simple consequence of Parikh's theorem, which states that the Parikh image of any context-free language is effectively semilinear [Par66]. Given A, the language $\mathsf{Run}(A)$ is effectively regular, thus Parikh's theorem asserts that $\mathsf{Pkh}(\mathsf{Run}(A)) = \mathsf{Pkh}(L(A'))$ is effectively semilinear.

Next, the Parikh image of a path describes a flow, and thus an initial path π on A ends in a state q different from the initial state iff the sum of the $|\pi|_t$ for the t's with $\mathsf{To}(t) = q$ is one more than the same sum for $\mathsf{From}(t) = q$. It ends in the initial state iff the previous sums are equal for all states. Thus the final state of an initial path depends only on its Parikh image.

Now, order $\delta' = \{t'_1, t'_2, \dots, t'_k\}$. Next, for $q \in F'$, let R_q be the semilinear set of Parikh images of initial paths in A' ending in q. Note that Fact 55 implies that for $\pi \in \mathsf{Run}(A')$, $\mathsf{Pkh}(\pi) \in R_q$ iff $\mathsf{To}(\pi) = q$. Then define $C' \subseteq \mathbb{N}^k$ by letting $\mathbf{x} = (x_1, x_2, \dots, x_k) \in C'$ iff:

$$\bigwedge_{q \in F'} \left[(\mathbf{x} \in R_q) \to \left(\sum_{i=1}^{k} x_i \times \mathsf{Pkh}(h(t'_i)) + \mathsf{Pkh}(E(q)) \right) \in C \right] .$$

Concluding the proof of Lemma 54, a word $w \in L(A', C')$ iff $w \in L(A')$ and the Parikh image of the only path labeled w in $\mathsf{Run}(A')$ is in C', that is iff $w \in L(A)$ and the Parikh image of the only path labeled w in $\mathsf{Run}(A)$ is in C, that is iff $w \in L(A, C)$.

▶ **Proposition 56.** *$\mathscr{L}_{\mathrm{UnCA}}$ is closed under union.*

Proof. We start with a simple fact about UnCA:

Fact 57. *For (A, C) an UnCA over Σ, there is an UnCA (A', C') with the same language with $L(A') = \Sigma^*$.*

Proof. Let (A, C) be an UnCA with $A = (Q, \Sigma, \delta, q_0, F)$ and let A' be a deterministic automaton for $\overline{L(A)}$, with $A' = (Q', \Sigma, \delta', q'_0, F')$. We suppose, without loss of generality, that $Q \cap Q' = \emptyset$. Now let $q \notin Q \cup Q'$, $t_1 = q \bullet\!\!-\varepsilon\!\!\to q_0$, $t_2 = q \bullet\!\!-\varepsilon\!\!\to q'_0$, and define A'' as:

$$A'' \quad = \quad (Q \cup Q' \cup \{q\}, \quad \Sigma, \quad \delta \cup \delta' \cup \{t_1, t_2\}, \quad q, \quad F \cup F') \ .$$

Clearly $L(A'') = \Sigma^*$. We show that A'' is unambiguous. Suppose $w \in L(A'')$, then there is a path $t \cdot \pi$ in $\mathsf{Run}(A'')$ with label w, and with $t \in \{t_1, t_2\}$. If $t = t_1$, then $\pi \in \mathsf{Run}(A)$, and thus no other path in $\mathsf{Run}(A'')$ starts with t_1 and has label $\mu(t \cdot \pi)$, by the unambiguity of A. Likewise, if $t = t_2$, then $\pi \in \mathsf{Run}(A')$, and thus no other path in $\mathsf{Run}(A'')$ starts with t_2 and has label $\mu(t \cdot \pi)$, by the determinism of A'. Further, if $t \cdot \pi$ and $t' \cdot \pi'$ are two paths of $\mathsf{Run}(A'')$ with the same label with $t \neq t'$, then $\mu(\pi) = \mu(\pi')$ is a word in both $L(A)$ and $L(A')$, a contradiction. Thus A'' is unambiguous.

Finally, we define D to check whether t_1 or t_2 has been taken, and in the former case check if the rest of the path has its Parikh image in C, while in the latter case reject. In symbols, with the transitions of A'' ordered so that t_1, t_2 come first then the transitions of A then those of A':

$$(x_1, x_2, y_1, y_2, \ldots, y_{|\delta|}, \ldots) \in D \Leftrightarrow x_1 = 1 \wedge (y_1, y_2, \ldots, y_{|\delta|}) \in C \ .$$

Thus the Parikh image of a path in $\mathsf{Run}(A'')$ is accepted by D iff the path started with t_1 and the rest of the path is an accepting path in A whose Parikh image is in C; i.e., $L(A'', D) = L(A, C)$.

Let (A, C) and (B, D) be two UnCA over the same alphabet Σ, and suppose, by Fact 57 followed by Lemma 54, that $L(A) = L(B) = \Sigma^*$ and that neither A nor B has a transition labeled ε. Write $A = (Q_A, \Sigma, \delta_A, q_{0,A}, F_A)$ and $B = (Q_B, \Sigma, \delta_B, q_{0,B}, F_B)$, then define $A \times B$ as the automaton:

$$\begin{aligned} A \times B &= (Q_A \times Q_B,\ \Sigma,\ \delta_{A\times B},\ (q_{0,A}, q_{0,B}),\ F_A \times F_B) \ , \\ \text{where} \quad \delta_{A\times B} &= \{(p,q) \bullet\!\!-a\!\!\to (p', q') \mid p \bullet\!\!-a\!\!\to p' \in \delta_A \wedge q \bullet\!\!-a\!\!\to q' \in \delta_B\} \ . \end{aligned}$$

Now this standard construction is such that a path is accepting in $A \times B$ iff the paths it traces in A and B are both accepting. More precisely, define the two morphisms $p_A : \delta^*_{A\times B} \to \delta^*_A$ and $p_B : \delta^*_{A\times B} \to \delta^*_B$ by $p_A((p,q) \bullet\!\!-a\!\!\to (p', q')) = p \bullet\!\!-a\!\!\to p'$ and $p_B((p,q) \bullet\!\!-a\!\!\to (p', q')) = q \bullet\!\!-a\!\!\to q'$. Then the accepting paths π of $A \times B$ are the paths such that $p_A(\pi) \in \mathsf{Run}(A)$, $p_B(\pi) \in \mathsf{Run}(B)$, and both paths have the same label. As for any word $w \in \Sigma^*$ there is exactly one accepting path labeled w in both $\mathsf{Run}(A)$ and $\mathsf{Run}(B)$, there is exactly one accepting path labeled w in $\mathsf{Run}(A \times B)$, hence $A \times B$ is unambiguous.

Now from the Parikh image of a path on $A \times B$, we may retrieve the Parikh images of the paths it traces in A and B, then force that either the former is in C or the latter is in D. Thus define E as the semilinear set which contains $\mathbf{z} \in \mathbb{N}^{|\delta_{A\times B}|}$ iff there exist $\mathbf{x} \in \mathbb{N}^{|\delta_A|}$ and $\mathbf{y} \in \mathbb{N}^{|\delta_B|}$ such that:

$$\bigwedge_{t\in\delta_A}\Bigg(x_t = \sum_{\substack{t' \in \delta_{A\times B}\\ p_A(t') = t}} z_{t'}\Bigg) \wedge \bigwedge_{t\in\delta_B}\Bigg(y_t = \sum_{\substack{t' \in \delta_{A\times B}\\ p_B(t') = t}} z_{t'}\Bigg) \wedge (\mathbf{x} \in C \vee \mathbf{y} \in D) \ .$$

Then a word is in $L(A \times B, E)$ iff the (one and only) path it traces in A has its Parikh image in C, or similarly for B and D. In other words, $L(A \times B, E) = L(A, C) \cup L(B, D)$.

▶ **Proposition 58.** *$\mathscr{L}_{\text{UnCA}}$ is closed under complement and intersection.*

Proof. Let (A, C) be an UnCA. A word w is *not* in $L(A, C)$ iff either $w \notin L(A)$ or $w \in L(A)$ but the Parikh image of the only path for w in A is rejected by C. Thus:

$$\overline{L(A, C)} = \overline{L(A)} \cup L(A, \overline{C}) \ .$$

Now $\overline{L(A)}$ is regular, thus $\overline{L(A)} \in \mathscr{L}_{\text{UnCA}}$. Moreover, $(A, \overline{C})$ is an UnCA. Thus $\overline{L(A, C)}$ is the union of the languages of two UnCA, and by Proposition 56, it is in $\mathscr{L}_{\text{UnCA}}$. Closure under intersection follows from the closure under union and complement.

For completeness, we add the two following closures:

▶ **Proposition 59.** *$\mathscr{L}_{\text{UnCA}}$ is closed under inverse morphisms and commutative closure.*

Proof. (Inverse morphisms) Let (A, C) be an UnCA over Σ and $h : \mathrm{T}^* \to \Sigma^*$ be a language morphism. Write $A = (Q, \Sigma, \delta, q_0, F)$ and $P(q, u, q')$ the only path in A from q to q' labeled u if it exists, and $\bot$ otherwise. We set $\mathsf{Path}(q, \varepsilon, q) = \varepsilon$ for any q. Define $B = (Q, \mathrm{T}, \delta', q_0, F)$ by:

$$\delta' = \{q \bullet\!\!- a \!\to q' \in Q \times \mathrm{T} \times Q \mid P(q, h(a), q') \neq \bot\} \ .$$

Now B is unambiguous as A is. Define $C' \subseteq \mathbb{N}^{|\delta'|}$ by:

$$\mathbf{x} \in C' \Leftrightarrow \sum_{t = q \bullet\!- a \to q' \in \delta'} x_t \times \mathsf{Pkh}(P(q, h(a), q')) \in C \ .$$

It is clear that $L(B, C') = h^{-1}(L(A, C))$, concluding the proof.

(Commutative closure) It is shown in Paper I, Proposition 18, that the commutative closure of a language in $\mathscr{L}_{\mathrm{CA}}$ is in $\mathscr{L}_{\mathrm{DetCA}}$, and thus in $\mathscr{L}_{\mathrm{UnCA}}$.

Note that $\mathscr{L}_{\mathrm{DetCA}}$ is not closed under reversal, as $\{a, b\}^* \cdot \{a^n b^n \mid n \geq 1\}$ is not in $\mathscr{L}_{\mathrm{DetCA}}$ while its reversal is (see Paper I). Thus it is a curiosity, especially for a class described by a deterministic model (see the forthcoming Theorem 84), that we have:

▶ **Proposition 60.** *$\mathscr{L}_{\mathrm{UnCA}}$ is closed under reversal.*

Proof. Let (A, C) be an UnCA. Let B be the ε-automaton A in which a fresh state q_{f} is set to be the only final state, and with a transition from each former final state to q_{f} labeled ε. Clearly, B is unambiguous. Adjust C into C' so that the added transitions in B do not affect the acceptance of a word, i.e., $L(B, C') = L(A, C)$. Then define D as the ε-automaton B in which every transition is reversed, i.e., $q \bullet\!\!-a\!\!\to q'$ is a transition of B iff $q' \bullet\!\!-a\!\!\to q$ is a transition of D; the order on the transition set of D is the same as that of B. Additionally, set q_{f} as the initial state and the former initial state of B as the only final state. Then D is unambiguous: clearly, $\mathsf{Run}(B)$ is the set of paths which are the reversal of paths in $\mathsf{Run}(D)$ and where each transition is reversed, thus the accepting paths in D labeled w are the reversal of the accepting paths in B labeled w^{R}. As B is unambiguous, only one such path may exist, thus D is unambiguous. Hence $L(D, C') = (L(B, C'))^{\mathrm{R}} = (L(A, C))^{\mathrm{R}}$.

We also note that the prefixes belonging to a CA language can be removed from the language of an UnCA:

▶ **Proposition 61.** *Let $L_1 \in \mathscr{L}_{\mathrm{CA}}$ and $L_2 \in \mathscr{L}_{\mathrm{UnCA}}$. Then $L_1^{-1} L_2 \in \mathscr{L}_{\mathrm{UnCA}}$.*

Proof. Let (A, C) be a CA, (B, D) an UnCA, with $A = (Q_A, \Sigma, \delta_A, q_{0,A}, F_A)$ and $B = (Q_B, \Sigma, \delta_B, q_{0,B}, F_B)$. We suppose, thanks to Lemma 54, that no transition of B is labeled by ε, and that each state of B is reachable from $q_{0,B}$ and can reach a final state. For $q \in Q_B$, define $B^{\to q}$ (resp. $B^{q \to}$) to be the ε-automaton B where the initial state (resp. the only final state) is q, and note that $B^{\to q}$ is unambiguous, as any path from q to a final state can be prefixed with a path from $q_{0,B}$ to q to make an accepting path in B. We first show:

Fact 62. *For any $q_B \in Q_B$, the set $E^{q_B} = \{(\mathsf{Pkh}(\pi), \mathsf{Pkh}(\rho)) \mid \pi \in \mathsf{Run}(A) \wedge \rho \in \mathsf{Run}(B^{q_B \to}) \wedge \mu_A(\pi) = \mu_B(\rho)\}$ is effectively semilinear.*

Proof. Define the morphisms $p_A : (\delta_A \times \delta_B)^* \to \delta_A^*$ and $p_B : (\delta_A \times \delta_B)^* \to \delta_B^*$ by $p_A((t_A, t_B)) = t_A$ and $p_B((t_A, t_B)) = t_B$. Next, let $\perp$ be a symbol not in Σ, and define the morphism $h : (\delta_A \times \delta_B)^* \to (\Sigma \cup \{\perp\})^*$ by $h((t_A, t_B)) = \mu_A(t_A)$ if $\mu_A(t_A) = \mu_B(t_B)$ and $h((t_A, t_B)) = \perp$ otherwise. Now let R be the regular language of the juxtapositions of accepting paths in A and $B^{q_B\rightarrow}$:

$$R = \{\pi \in (\delta_A \times \delta_B)^* \mid p_A(\pi) \in \mathsf{Run}(A) \land p_B(\pi) \in \mathsf{Run}(B^{q_B\rightarrow})\} \ .$$

Now let $L = h^{-1}(L(A) \cap L(B^{q_B\rightarrow})) \cap R$, thus L is the set of words π over $(\delta_A \times \delta_B)$ such that $p_A(\pi) \in \mathsf{Run}(A)$, $p_B(\pi) \in \mathsf{Run}(B^{q_B\rightarrow})$, and those two paths have the same label. Set an order on $(\delta_A \times \delta_B)$, and note that $\mathsf{Pkh}(L)$ is effectively semilinear by Parikh's theorem [Par66]. Then $(\mathbf{x}, \mathbf{y}) \in E^{q_b}$ iff:

$$(\exists \mathbf{z} \in \mathsf{Pkh}(L)) \left[\bigwedge_{t \in \delta_A} \left(x_t = \sum_{t' \in \delta_B} z_{(t,t')} \right) \land \bigwedge_{t' \in \delta_B} \left(y_{t'} = \sum_{t \in \delta_A} z_{(t,t')} \right) \right] \ .$$

A word w is in $(L(A, C))^{-1} L(B, D)$ iff there is a state $q_B \in Q_B$ and a word $u \in L(A, C)$ such that $u \in L(B^{q_B\rightarrow})$, $w \in L(B^{\rightarrow q_B})$, and the Parikh image of one (in fact, the only) path for u in $B^{q_B\rightarrow}$ concatenated with the path for w in $B^{\rightarrow q_B}$ is in D. This is the case iff there is a state $q_B \in Q_B$ and a pair $(\mathbf{x}, \mathbf{y}) \in E^{q_B}$ such that $\mathbf{x} \in C$ and the Parikh image $\mathbf{z}$ of the only path in $B^{\rightarrow q_B}$ labeled w plus $\mathbf{y}$ is in D. In symbols, a word w is in $(L(A, C))^{-1} L(B, D)$ iff it is in:

$$\bigcup_{q_B \in Q_B} L(B^{\rightarrow q_B}, \ \{\mathbf{z} \mid (\exists (\mathbf{x}, \mathbf{y}) \in E^{q_B})[\mathbf{x} \in C \land \mathbf{y} + \mathbf{z} \in D]\}) \ .$$

As $\mathscr{L}_{\text{UnCA}}$ is closed under union (Proposition 56), this implies the result.

Remark. In the previous proof, if (B, D) is a DetCA, then we obtain at the end a set of DetCA, the union of the languages of which is $(L(A, C))^{-1} L(B, C)$. As $\mathscr{L}_{\text{DetCA}}$ is closed under union, this shows that $\mathscr{L}_{\text{DetCA}}$ is also closed under left quotient. Moreover, if (B, D) is an UnCA, then $B^{q\rightarrow}$ is unambiguous: if two accepting paths therein have the same label, then there are two ways to get from $q_{0,B}$ to q reading the same word, and since a final state can be reached from q, the unambiguity of B implies that they are the same paths. Likewise, if (B, D) is a DetCA, then $B^{q\rightarrow}$ is deterministic. Thus a similar proof as the above shows that both $\mathscr{L}_{\text{UnCA}}$ and $\mathscr{L}_{\text{DetCA}}$ are closed under *right* quotient — in the case of DetCA, this settles those questions left open in [KR02]. An alternative proof of the closure under right quotient of $\mathscr{L}_{\text{UnCA}}$ is to note that $L_1(L_2)^{-1} = ((L_2^{\mathrm{R}})^{-1}L_1^{\mathrm{R}})^{\mathrm{R}}$. Thus the closure of $\mathscr{L}_{\text{UnCA}}$ under reversal (Proposition 60) and under left quotient (Proposition 61) implies that $\mathscr{L}_{\text{UnCA}}$ is indeed closed under right quotient.

We introduce an expressiveness lemma inspired by Lemma I.7:

▶ **Lemma 63.** *Let $L \subseteq \Sigma^*$ be in $\mathscr{L}_{\text{CA}}$. There exist $p, \ell \geq 1$ such that for any $v_0, v_1, \ldots, v_\ell \in \Sigma^*$ and $u_1, u_2, \ldots, u_\ell \in \Sigma^{\geq p}$ such that $v_0u_1v_1 \cdots u_\ell v_\ell \in L$, there exist $1 \leq i < j \leq \ell$ and a nonempty $w \in \Sigma^*$ with $|w| \leq p$ such that:*
(1). $u_i = u_{i,1} \cdot w \cdot u_{i,2}$ and $u_j = u_{j,1} \cdot w \cdot u_{j,2}$,
(2). $v_0u_1v_1 \cdots (u_{i,1} \quad \cdot \quad u_{i,2})v_i \cdots (u_{j,1} \cdot w^2 \cdot u_{j,2})v_j \cdots u_\ell v_\ell \in L$,
(3). $v_0u_1v_1 \cdots (u_{i,1} \cdot w^2 \cdot u_{i,2})v_i \cdots (u_{j,1} \quad \cdot \quad u_{j,2})v_j \cdots u_\ell v_\ell \in L$.

Proof. Let $L \subseteq \Sigma^*$ be in $\mathscr{L}_{\text{CA}}$. There is a CA (A, C) such that $L = L(A, C)$. Fix p to be the number of states in A and ℓ to be the number of elementary cycles (i.e., cycles in which no state except the first occurs twice) in A. Let $v_0, u_1, v_1, \ldots, u_\ell, v_\ell$ as in the statement of the lemma. Then there is a path π in $\mathsf{Run}(A)$ such that $\mu(\pi) = v_0u_1v_1 \cdots u_\ell v_\ell$ and $\mathsf{Pkh}(\pi) \in C$. Write this path $\pi = \rho_0\pi_1\rho_1 \cdots \pi_\ell\rho_\ell$ with, for all i, $\mu(\rho_i) = v_i$ and $\mu(\pi_i) = u_i$. Then, by the pigeonhole principle, there exist $i < j$ such that π_i and π_j share the same elementary cycle, i.e., there is a nonempty cycle γ of size less than p such that $\pi_i = \pi_{i,1} \cdot \gamma \cdot \pi_{i,2}$ and $\pi_j = \pi_{j,1} \cdot \gamma \cdot \pi_{j,2}$. This implies that the two paths $\rho_0\pi_1\rho_1 \cdots (\pi_{i,1} \cdot \pi_{i,2})\rho_i \cdots (\pi_{j,1} \cdot \gamma^2 \cdot \pi_{j,2})\rho_j \cdots \pi_\ell\rho_\ell$ and $\rho_0\pi_1\rho_1 \cdots (\pi_{i,1} \cdot \gamma^2 \cdot \pi_{i,2})\rho_i \cdots (\pi_{j,1} \cdot \pi_{j,2})\rho_j \cdots \pi_\ell\rho_\ell$ are accepting paths of A with a Parikh image in C. Thus their labels are in L, and that is the statement of the lemma.

The expressiveness lemma I.7 is akin to the case where all the v_i's are of size greater than p. Define:

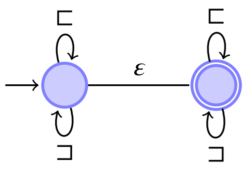
$$P_1 = \{w = w_1w_2 \cdots w_k \in \{\sqsubset, \sqsupset\}^* \mid (\forall i)[|w_1w_2 \cdots w_i|_\sqsubset \geq |w_1w_2 \cdots w_i|_\sqsupset]\} \;,$$

as the prefixes of the semi-Dyck language with one set of parentheses. Then:

▶ **Proposition 64.** *$P_1 \notin \mathscr{L}_{\text{CA}}$ and $\overline{P_1} \in \mathscr{L}_{\text{CA}} \setminus \mathscr{L}_{\text{UnCA}}$.*

Proof. ($P_1 \notin \mathscr{L}_{\text{CA}}$.) We use Lemma 63. Suppose $P_1 \in \mathscr{L}_{\text{CA}}$, and let p, ℓ be as in Lemma 63. Define $v_0 = \varepsilon$ and for all $1 \leq i \leq \ell$, $u_i = \sqsubset^p$, $v_i = \sqsupset^p$. Lemma 63 then asserts that there is $1 \leq k \leq p$ such that $u_1v_1 \cdots \sqsubset^{p-k}\sqsupset^p \cdots \sqsubset^{p+k}\sqsupset^p \cdots u_\ell v_\ell \in P_1$, a contradiction.

($\overline{P_1} \in \mathscr{L}_{\mathrm{CA}}$.) We give a CA (A, C) recognizing $\overline{P_1}$. The ε-automaton guesses a position, and the constraint set checks that, up to this position, the number of $\sqsubset$'s read is less than the number of $\sqsupset$'s. The ε-automaton A is described by:



The constraint set C checks that the loop labeled $\sqsubset$ on the initial state occurs less than the loop labeled $\sqsupset$ on the initial state. We leave the simple proof that $L(A, C) = \overline{P_1}$.

($\overline{P_1} \notin \mathscr{L}_{\mathrm{UnCA}}$.) This is a direct consequence of the closure under complement of $\mathscr{L}_{\mathrm{UnCA}}$ (Proposition 56). Suppose $\overline{P_1} \in \mathscr{L}_{\mathrm{UnCA}}$, then $P_1 \in \mathscr{L}_{\mathrm{UnCA}}$, but $\mathscr{L}_{\mathrm{UnCA}} \subseteq \mathscr{L}_{\mathrm{CA}}$ as an UnCA is a CA, thus $P_1 \in \mathscr{L}_{\mathrm{CA}}$, a contradiction.

▶ **Theorem 65.** $\mathscr{L}_{\mathrm{DetCA}} \subsetneq \mathscr{L}_{\mathrm{UnCA}} \subsetneq \mathscr{L}_{\mathrm{CA}}$.

Proof. ($\mathscr{L}_{\mathrm{DetCA}} \subsetneq \mathscr{L}_{\mathrm{UnCA}}$.) The inclusion follows from the fact that a deterministic automaton is unambiguous, thus a DetCA is an UnCA. The strictness of the inclusion is shown in Paper I: the language $\{a, b\}^* \cdot \{a^n b^n \mid n \geq 1\}$ is in $\mathscr{L}_{\mathrm{UnCA}} \setminus \mathscr{L}_{\mathrm{DetCA}}$. Additionally, we already hinted that $\mathscr{L}_{\mathrm{DetCA}}$ is not closed under reversal while $\mathscr{L}_{\mathrm{UnCA}}$ is (Proposition 60), implying that the two classes differ.

($\mathscr{L}_{\mathrm{UnCA}} \subsetneq \mathscr{L}_{\mathrm{CA}}$.) We already noted that the inclusion is immediate, as an UnCA is a CA. Its strictness comes from Proposition 64, or alternatively, from the fact that $\mathscr{L}_{\mathrm{CA}}$ is not closed under complement while $\mathscr{L}_{\mathrm{UnCA}}$ is.

▶ **Proposition 66.** *$\mathscr{L}_{\mathrm{UnCA}}$ is neither closed under concatenation with a regular language, nor under length-preserving morphisms, nor under starring.*

Proof. (Concatenation.) Let $\Sigma = \{\sqsubset, \sqsupset\}$. The language $L_< = \{w \in \Sigma^* \mid |w|_\sqsubset < |w|_\sqsupset\}$ is in $\mathscr{L}_{\mathrm{DetCA}}$ and such that $\overline{P_1} = L_< \cdot \Sigma^* \notin \mathscr{L}_{\mathrm{UnCA}}$. Thus if $\mathscr{L}_{\mathrm{UnCA}}$ were closed under concatenation, then $\overline{P_1}$ would be in $\mathscr{L}_{\mathrm{UnCA}}$, contradicting Proposition 64.

(Length-preserving morphisms and starring.) Let $\mathrm{T} = \{\sqsubset\!\!|, |\!\!\sqsupset\}$, then $L_< \cdot \mathrm{T}^* \in \mathscr{L}_{\mathrm{UnCA}}$. The length-preserving morphism $h \colon (\Sigma \cup \mathrm{T})^* \to \Sigma^*$ defined by $h(\sqsubset\!\!|) = h(\sqsubset) = \sqsubset$, $h(|\!\!\sqsupset) = h(\sqsupset) = \sqsupset$ is such that $h(L_< \cdot \mathrm{T}^*) = L_< \cdot \Sigma^* \notin \mathscr{L}_{\mathrm{UnCA}}$. For starring, it is shown in Proposition I.18 that with $L = \{a^n b^n \mid n \in \mathbb{N}\} \in \mathscr{L}_{\text{DetCA}}$, $L^* \notin \mathscr{L}_{\text{CA}} \supsetneq \mathscr{L}_{\text{UnCA}}$.

3 UnCA and RBCM

RBCM: see Section 0.3.4, p. 16

It is known that *one-way reversal-bounded counter machines* (RBCM) [Iba78] are as powerful as CA [KR03], while deterministic such machines (DetRBCM) are more powerful than DetCA (see Paper I). In this section, we carry this study further by showing that the expressive power of DetRBCM is incomparable with that of UnCA.

▶ **Proposition 67.** *$\mathscr{L}_{\text{DetRBCM}}$ and $\mathscr{L}_{\text{UnCA}}$ are incomparable.*

Proof. ($\mathscr{L}_{\text{DetRBCM}} \nsubseteq \mathscr{L}_{\text{UnCA}}$.) A DetRBCM can deterministically use extra information provided in the input word to check for a certain property later in the input; this is illustrated by:

$$L = \{a^n w \mid w \in \{\sqsubset, \sqsupset\}^* \wedge |w_1 w_2 \cdots w_n|_{\sqsubset} < |w_1 w_2 \cdots w_n|_{\sqsupset}\} \in \mathscr{L}_{\text{DetRBCM}} \ .$$

Indeed, the DetRBCM starts by counting the number of a's, then decrements this counter while reading w and counting the number of $\sqsubset$'s and $\sqsupset$'s. When the number of a's reaches zero, the machine checks whether the number of $\sqsubset$'s read so far is strictly less than the number of $\sqsupset$'s read, and rejects iff it is not the case.

Suppose $L \in \mathscr{L}_{\text{UnCA}}$. Proposition 61 then asserts that $(\{a\}^*)^{-1} L \cap \{\sqsubset, \sqsupset\}^*$ is in $\mathscr{L}_{\text{UnCA}}$. But this latter language is $\overline{P_1} \notin \mathscr{L}_{\text{UnCA}}$ (Proposition 64), a contradiction.

($\mathscr{L}_{\text{UnCA}} \nsubseteq \mathscr{L}_{\text{DetRBCM}}$.) The language $\{a, b\}^* \cdot \{a^n b^n \mid n \geq 1\}$ is in $\mathscr{L}_{\text{UnCA}}$ but not in $\mathscr{L}_{\text{DetRBCM}}$ (see Paper I).

4 Decision problems for UnCA

We recall the following decidability results, that hold equally well for UnCA:

▶ **Proposition 68** ([KR03], Paper I). *Given a CA, it is decidable whether its language is empty, and whether its language is finite.*

With the closure properties of $\mathscr{L}_{\text{UnCA}}$ of Proposition 58, this implies:

▶ **Proposition 69.** *Given an UnCA, it is decidable whether its language is Σ^*. Given two UnCA, it is decidable whether the language of the first is included in the language of the second.*

The rest of this section is devoted to the main technical result of our paper, namely that it is decidable whether the language of an UnCA is regular. Our technique is in two steps: we first show that it is decidable whether a *bounded* CA language (given additionally a socle of the language) is regular (Lemma 72) then reduce the decision in the general case to the decision with bounded CA languages.

Bounded languages: see p. 6

▶ **Definition 6** ([GS66a]). A set C is *unary* if it is equal to a finite union of linear sets, each period of each linear set having at most one nonzero coordinate.

▶ **Lemma 70** ([GS66a, Theorem 1.3]). *Let $L \subseteq w_1^* w_2^* \cdots w_n^*$. The language L is regular iff $\mathsf{Iter}_{(w_1,w_2,\ldots,w_n)}(L)$ is unary.*

▶ **Lemma 71** ([GS66a, Section 3]). *Given a semilinear set C, it is decidable whether C is unary.*

▶ **Lemma 72.** *Given a CA (A, C) and words $w_1, w_2, \ldots, w_n$ such that $L(A, C)$ is a bounded language and $(w_1, w_2, \ldots, w_n)$ is one of its socles, it is decidable whether $L(A, C)$ is regular.*

Proof. Let (A, C) be a CA with $L(A, C) \subseteq w_1^* w_2^* \cdots w_n^*$. Let T be the set of fresh symbols $\{a_1, a_2, \ldots, a_n\}$ and define the morphism $h : \mathrm{T}^* \to \Sigma^*$ by $h(a_i) = w_i$ for all i. Now let (A', C') be the CA with language $h^{-1}(L(A, C)) \cap a_1^* a_2^* \cdots a_n^*$ obtained by the (effective) closures of CA. Then for $\mathbf{i} \in \mathbb{N}^n$, $a_1^{i_1} a_2^{i_2} \cdots a_n^{i_n} \in L(A', C')$ iff $w_1^{i_1} w_2^{i_2} \cdots w_n^{i_n} \in L(A, C)$. Hence:

$$\mathsf{Pkh}(L(A', C')) = \mathsf{Iter}_{(w_1,w_2,\ldots,w_n)}(L(A, C)) \enspace .$$

Now $\mathsf{Pkh}(L(A', C'))$ is a semilinear set that can be (effectively) obtained [Kar04, Theorem 3.21], and we may thus check whether it is unary using Lemma 71. This amounts to deciding, by Lemma 70, whether $L(A, C)$ is regular.

▶ **Lemma 73.** *Let C be unary. Then there exist $s \geq 0$ and $p \geq 1$ such that for any $s' \geq 0$, $p' \geq 1$, C is the (finite) union of sets of the form $[\mathbf{x}]_{s+s',pp'}$.*

Proof. Let $C = \bigcup_{1\leq i\leq \ell}(\mathbf{a}_i + \mathbf{b}_{i,1} \times \mathbb{N} + \mathbf{b}_{i,2} \times \mathbb{N} + \cdots + \mathbf{b}_{i,k_i} \times \mathbb{N}) \subseteq \mathbb{N}^d$, where each $k_i \geq 0$ and only one entry in any $\mathbf{b}_{i,m}$ is nonzero.

Let $s - 1 \in \mathbb{N}$ be the maximum entry of any $\mathbf{a}_i$, p be the least common multiple of all the entries of all the $\mathbf{b}_{i,m}$, and $s' \geq 0$, $p' \geq 1$. For $\mathbf{x} \in \mathbb{N}^d$, let $\mathbf{x}^-$ be the only element in $[\mathbf{x}]_{s+s',pp'} \cap \{0, 1, \ldots, s+s'+pp'-1\}^d$. We show that $\mathbf{x} \in C$ iff $\mathbf{x}^- \in C$.

Let $\mathbf{x} \in C$. For $1 \leq m \leq d$, if $x_m \geq s + s' + pp'$, then subtracting pp' from x_m yields another tuple in C. As $\mathbf{x}^-$ is obtained by this repeated process it is in C. Conversely, suppose $\mathbf{x}^- \in C$. For $1 \leq m \leq d$, if $x^-_m \geq s + s'$, then adding pp' to x_m yields another tuple in C. As $x_m \equiv_{s+s',pp'} x^-_m$, $\mathbf{x}$ is obtained by this repeated process, thus $\mathbf{x} \in C$.

Hence:

$$C = \bigcup_{\mathbf{x}\in C\cap\{0,1,\ldots,s+s',pp'\}^d} [\mathbf{x}]_{s+s',pp'} \enspace .$$

Remark. The converse of the previous lemma is true, but will not be needed in what follows.

We continue with a lemma that allows us to focus on languages of *paths*:

▶ **Lemma 74.** *The language of an UnCA (A, C) is regular iff $\mathsf{Run}(A)\restriction_C$ is regular.*

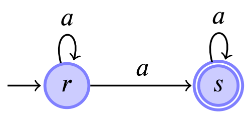
Proof. First, suppose $\mathsf{Run}(A)\restriction_C$ is regular, for a CA (A, C). As by definition $L(A, C) = \mu(\mathsf{Run}(A)\restriction_C)$ and regular languages are closed under morphisms, we have that $L(A, C)$ is regular. This part does not rely on unambiguity.

Second, consider an UnCA (A, C). We remark that if an accepting path of A is labeled by a word in $L(A, C)$, then it is in $\mathsf{Run}(A)\restriction_C$ (the converse is true of any CA). Indeed, since a path labeled by a word w in $L(A, C)$ is, by unambiguity, the only path labeled w in $\mathsf{Run}(A)$, it has its Parikh image in C. In other words:

$$\mathsf{Run}(A)\restriction_C = \mu^{-1}(L(A, C)) \cap \mathsf{Run}(A) \enspace .$$

Now, as the class of regular languages is closed under inverse morphisms and intersection, if $L(A, C)$ is regular then $\mathsf{Run}(A)\restriction_C$ is regular.

Remark. The inclusion $\mathsf{Run}(A)\restriction_C \supseteq \mu^{-1}(L(A, C)) \cap \mathsf{Run}(A)$ is crucial to the proof of Lemma 74 and to the decidability of regularity for UnCA. Indeed, both this inclusion and Lemma 74 fail for CA — in fact, regularity is undecidable for CA (see Paper I). For example, let A be the automaton:



Next, define C to constrain the two loops on r and s to occur the same number of times. Then $L(A,C) = \{a^{2n+1} \mid n \in \mathbb{N}\}$, a regular language. But with t_1, t_2, t_3 the three transitions of A, from left to right, $\mathsf{Run}(A)\restriction_C = \{t_1^n t_2 t_3^n \mid n \in \mathbb{N}\}$, a nonregular language.

As $\mathsf{Run}(A)$ is effectively obtainable from A, we need only focus on the decidability of the regularity of $\mathsf{Run}(A)\restriction_C$. We already noted in different expressiveness lemmata for CA and DetCA (e.g., Lemma 63) that "moving a cycle" within a run affects neither its being an accepting path, nor its Parikh image. Repeatedly moving the cycles to the leftmost position in the run at which they can occur will be a key ingredient in the following proof. This operation, in particular, will allow to convert the language of runs in an ε-automaton to a set of *bounded* languages, with the property that a path is accepting iff the repeated moving of cycles leads to a path in one of the bounded languages. We will then see that this moving can be done by a simple process which an automaton can somewhat simulate. With the addition of restrictions (the $\restriction$ operation), we will see that the restriction of the language of runs is regular iff the restriction of each of those bounded languages is regular. We then rely on Lemma 72 to decide regularity.

▶ **Theorem 75.** *It is decidable whether the language of an UnCA is regular.*

Proof. Let (A, C) be a UnCA with $A = (Q, \Sigma, \delta, q_0, F)$. Lemma 74 implies that we need only show the decidability of the regularity of $R = \mathsf{Run}(A)\restriction_C$.

We first formalize the discussion made before this theorem. In the following, we use Latin letters b, u, v, w to denote *paths*, and more generally words over δ, as we no longer consider words over Σ. We use the term *cycle* for nonempty paths starting and ending in the same state and with no other state appearing twice, i.e., an elementary cycle in the underlying multigraph. Fix an ordering on the cycles of A: $\{b_1, b_2, \ldots, b_\ell\} \subseteq \delta^*$. Let S be the set of initial paths in A, including the empty path. For $w \in S$, define $\mathsf{States}(w)$ as the set of states visited by w. We see the empty path as from and to q_0, so that $\mathsf{States}(\varepsilon) = \{q_0\}$ and $\mathsf{From}(\varepsilon) = \mathsf{To}(\varepsilon) = q_0$.

Define $\alpha\colon S \to (S \times \mathbb{N}^\ell)$ by $\alpha(\varepsilon) = (\varepsilon, \mathbf{0})$ and, for $u \cdot t \in S$ where $t \in \delta$ and $\alpha(u) = (v, \mathbf{x})$:

$$\alpha(u \cdot t) = \begin{cases} (v', \mathbf{x} + \mathbf{e}_i) & \text{if } v \cdot t = v' b_i \wedge \mathsf{States}(b_i) \subseteq \mathsf{States}(v') \ , \\ (v \cdot t, \mathbf{x}) & \text{otherwise} \ . \end{cases}$$

Note that α is well-defined and that, for any $u \in S$, $\alpha(u) = (w, \mathbf{x})$ is such that w is indeed in S.

In words, applying α removes most of the cycles in a path, and counts them. Hence, if we see $\alpha(u) = (w, \mathbf{x})$ as the path w in which b_i is placed x_i times on the first occurrence of $\mathsf{From}(b_i)$ in w, we may interpret the action of α as "moving to the left" each cycle read, while "removing their nesting." Additionally, this path is in R iff u is in R.

In order to make the preceding intuition formal, we define the different bounded languages that represent $\mathsf{Run}(A)$ when the cycles are moved to the leftmost position where they fit. First, for $q \in Q$, fix a compatible ordering on the cycles with q as their origin: $\{b_{(q,1)}, b_{(q,2)}, \ldots, b_{(q,\ell_q)}\}$, i.e., if $b_i = b_{(q,i')}$, $b_j = b_{(q,j')}$, and $i < j$ then $i' < j'$. We write, as usual, $\mathbf{b}_q$ for $(b_{(q,1)}, b_{(q,2)}, \ldots, b_{(q,\ell_q)})$. Define, for $q \in Q$, the regular language $B_q = b^*_{(q,1)} b^*_{(q,2)} \cdots b^*_{(q,\ell_q)}$. Now for $w \in S$, let $(q_0, q_1, \ldots, q_n)$ be an ordering of $\mathsf{States}(w)$ such that if q_i is first met before q_j in w, then $i < j$ — that is, the q_i's are ordered in their order of first appearance in w. Further, let $1 = i_0, i_1, \ldots, i_n$ be the positions in w of the first appearance of $q_0, q_1, \ldots, q_n$, respectively. Then we define the bounded regular language $E_w \subseteq S$:

$$E_w = B_{q_0} \cdot w_{[i_0, i_1 - 1]} \cdot B_{q_1} \cdot w_{[i_1, i_2 - 1]} \cdots B_{q_n} \cdot w_{[i_n, |w|]} \ ,$$

where $w = w_1 w_2 \cdots w_{|w|}$ and $w_{[a,b]} = w_a w_{a+1} \cdots w_b$. In particular, $E_\varepsilon = B_{q_0}$. Let C_w be the iteration set $\mathsf{Iter}_{(\mathbf{b}_{q_0}, w_{[i_0,i_1-1]}, \ldots, \mathbf{b}_{q_n}, w_{[i_n,|w|]})}(E_w \cap R)$ and define I_w using C_w and focusing on the cycles, i.e., for $\mathbf{x} \in \mathbb{N}^\ell$, $\mathbf{x} \in I_w$ iff:

$$(\mathbf{x}_{q_0}, 1, \mathbf{x}_{q_1}, 1, \ldots, \mathbf{x}_{q_n}, 1) \in C_w \wedge (\forall q \in Q \setminus \{q_0, q_1, \ldots, q_n\})[\mathbf{x}_q = \mathbf{0}] \ ,$$

where $\mathbf{x}_q \in \mathbb{N}^{\ell_q}$, and $x_{(q,i)}$ is understood as the variable x_j for which $b_j = b_{(q,i)}$. Note that if $I_w \neq \emptyset$, then $w \in \mathsf{Run}(A)$. We are now ready to clarify the informal discussion made before the theorem:

Fact 76. *For all $u \in S$, $u \in R$ iff $\alpha(u) \in \{(w, \mathbf{x}) \mid \mathbf{x} \in I_w\}$.*

Proof. Let $u \in S$ and write $\alpha(u) = (w, \mathbf{x})$. By induction on $|u|$, we show that (i). $\mathsf{Pkh}(u) = \mathsf{Pkh}(w) + \sum_i x_i \times \mathsf{Pkh}(b_i)$; (ii). $\mathsf{To}(w) = \mathsf{To}(u)$; (iii). if $x_i \neq 0$ then $\mathsf{From}(b_i) \in \mathsf{States}(w)$. If $|u| = 0$, this is clear. So let $u \cdot t \in S$ such that $t \in \delta$, and write $\alpha(u \cdot t) = (w, \mathbf{x})$. Let $\alpha(u) = (v, \mathbf{y})$ which, by induction hypothesis, verifies (i–iii).

Case 1: suppose $v \cdot t$ can be written $v' b_j$ for some $v' \in S$, $1 \leq j \leq \ell$, and with $\mathsf{States}(b_j) \subseteq \mathsf{States}(v')$. Then $w = v'$ and $\mathbf{x} = \mathbf{y} + \mathbf{e}_j$. Then (i) is verified, as $\mathsf{Pkh}(u \cdot t) = \mathsf{Pkh}(u) + \mathsf{Pkh}(t) = \mathsf{Pkh}(v) + \sum_i y_i \times \mathsf{Pkh}(b_i) + \mathsf{Pkh}(t) = \mathsf{Pkh}(v') + \sum_i x_i \times \mathsf{Pkh}(b_i)$. Also, (ii) is verified since $\mathsf{To}(t) = \mathsf{To}(b_j) = \mathsf{To}(v')$, as $v' b_j$ is a path and b_j a cycle. Finally, (iii) is also verified since $\mathsf{States}(b_j) \subseteq \mathsf{States}(v')$ implies that $\mathsf{States}(v') = \mathsf{States}(v' b_j) = \mathsf{States}(v \cdot t)$, which, by induction hypothesis, is $\mathsf{States}(u \cdot t)$.

Case 2: suppose otherwise. Then $w = v \cdot t$ and $\mathbf{x} = \mathbf{y}$, and (i–iii) are satisfied by induction hypothesis.

This concludes the proof by induction. Now define $u' \in S$ as follows. Start with $u' \leftarrow w$. For each $i = 1, 2, \ldots, \ell$, in that order, if $x_i \neq 0$, then add $b_i^{x_i}$ at the position in u' where $\mathsf{From}(b_i)$ is first met — (iii) asserts that it is met at least once. Then (ii) implies that $\mathsf{To}(u') = \mathsf{To}(u)$. Moreover, (i) implies that $\mathsf{Pkh}(u) = \mathsf{Pkh}(u')$. Thus $u \in R$ iff $u' \in R$. Now note that thanks to the compatibility of the different orderings of the cycles, $u' \in E_w$. Thus $u' \in R$ iff $u' \in E_w \cap R$, that is, iff $\mathbf{x} \in I_w$.

If R is regular, then any $E_w \cap R$ is regular. We will show, using the previous fact as a decision procedure for R, that if all the $E_w \cap R$ are regular, then R is regular. The function α gives a hint of an automaton for R; however, the "accepting set" of Fact 76 clearly establishes that the state set is infinite. To circumvent this problem, we show that we can consider only finite objects with the two following facts, the second being a consequence of Lemma 70.

Fact 77. *There is a computable finite set S^{fin} such that any word w appearing as $\alpha(u) = (w, \cdot)$ is in S^{fin}.*

Proof. We show that, for $u \in S$, $\alpha(u) = (w, \cdot)$ is such that a cycle b_i cannot appear twice in w. This is shown by induction on $|u|$. If $|u| = 0$, then this is clearly true. So let $u \cdot t \in S$, with $t \in \delta$. Let $\alpha(u) = (v, \cdot)$, where, by induction hypothesis, no cycle is repeated in v. With $\alpha(u \cdot t) = (w, \cdot)$, note that w is a prefix of $v \cdot t$. If $v \cdot t$ does not contain the same cycle twice, then we are done. So suppose $v \cdot t$ contains the same cycle b twice. Then b should appear at the end of $v \cdot t$, for otherwise, the repeated b cycles appear in v. Thus $v \cdot t = v'bv''b$ for some path v', v''. In this case, $\mathsf{States}(b) \subseteq \mathsf{States}(v'bv'')$, and thus $\alpha(u \cdot t) = (v'bv'', \cdot)$, and as $v'bv''$ is a prefix of v, no cycle appears twice in it.

Now let $w \in S$ of length greater than $(\ell + 1) \times |Q|$. Then each of the first $\ell + 1$ blocks of $|Q|$ transitions of w contain a cycle. Hence by the pigeonhole principle, a cycle gets repeated in w. This implies, by the previous discussion, that no $u \in S$ is such that $\alpha(u) = (w, \cdot)$.

Fact 78. *Suppose that for all $w \in S^{\mathrm{fin}}$, $E_w \cap R$ is regular. There exist $s \geq 0$, $p \geq 1$ such that for any $\mathbf{x} \in \mathbb{N}^\ell$, $\mathbf{x} \in I_w$ iff $[\mathbf{x}]_{s,p} \subseteq I_w$.*

Proof. Suppose that for all $w \in S^{\mathrm{fin}}$, $E_w \cap R$ is regular. We show that each I_w, $w \in S^{\mathrm{fin}}$, is the union of some $[\mathbf{x}]_{s,p}$, implying the Fact. For any $w \in S^{\mathrm{fin}}$, I_w is a unary set by Lemma 70. By Lemma 73, there exist integers s_w and p_w such that I_w is a finite union of sets of the form $[\mathbf{x}]_{s_w,p_w}$. Now let $s = \sum_{w \in S^{\mathrm{fin}}} s_w$ and $p = \prod_{w \in S^{\mathrm{fin}}} p_w$. The same Lemma 73 then asserts that each I_w, $w \in S^{\mathrm{fin}}$, can be written as a finite union of sets of the form $[\mathbf{x}]_{s,p}$.

Suppose that for all $w \in S^{\mathrm{fin}}$, $E_w \cap R$ is regular, and let s, p be given by Fact 78. We define a deterministic automaton B for R by:

$$B = (S^{\mathrm{fin}} \times (\mathbb{N}^{|\delta|}/{\equiv_{s,p}}), \quad \delta, \quad \Delta, \quad (\varepsilon, [\mathbf{0}]_{s,p}), \quad T) \ ,$$
$$\Delta = \{(u, [\mathbf{x}]_{s,p}) \bullet\!\!-t\!\!\rightarrow (u', [\mathbf{x} + \mathbf{e}]_{s,p}) \mid u \cdot t \in S \wedge \alpha(u \cdot t) = (u', \mathbf{e})\} \ ,$$
$$T = \{(w, [\mathbf{x}]_{s,p}) \mid [\mathbf{x}]_{s,p} \subseteq I_w\} \ .$$

The set Δ is well-defined as $\mathbf{x} \equiv_{s,p} \mathbf{x}'$ implies $\mathbf{x} + \mathbf{e} \equiv_{s,p} \mathbf{x}' + \mathbf{e}$. Also, for any word $u \in S$ (and only for them) there is a path from the initial state labeled u.

Fact 79. *Suppose that for all $w \in S^{\mathrm{fin}}$, $E_w \cap R$ is regular. Let $u \in S$, $\alpha(u) = (w, \mathbf{x})$, and Π be the initial path on B labeled u. Then $\mathsf{To}(\Pi) = (w, [\mathbf{x}]_{s,p})$.*

Proof. This stems from the following property: if $u \cdot t \in S$ with $t \in \delta$, and if we write $\alpha(u) = (v, \mathbf{x})$ and $\alpha(v \cdot t) = (v', \mathbf{e})$, then $\alpha(u \cdot t) = (v', \mathbf{x} + \mathbf{e})$.

We show the fact by induction on $|u|$. If $|u| = 0$ then this is clear. So let $u \cdot t \in S$ with $t \in \delta$. Let Π be the unique initial path on B labeled u. With $\alpha(u) = (v, \mathbf{x})$, we have, by induction hypothesis, that $\mathsf{To}(\Pi) = (v, [\mathbf{x}]_{s,p})$. Write $\alpha(v \cdot t) = (v', \mathbf{e})$, then the ending state of the unique initial path on B labeled $u \cdot t$ is $(v', [\mathbf{x}+\mathbf{e}]_{s,p})$, and as $\alpha(u \cdot t) = (v', \mathbf{x}+\mathbf{e})$, this concludes the proof.

Let $u \in S$ and $\alpha(u) = (w, \mathbf{x})$. Then $u \in L(B)$ iff, by the Fact 79, $(w, [\mathbf{x}]_{s,p}) \in T$, that is, iff $[\mathbf{x}]_{s,p} \subseteq I_w$. By Fact 78, this is the case iff $\mathbf{x} \in I_w$. By Fact 76, this is the case iff $u \in R \cap S$, i.e., iff $u \in R$. Thus $L(B) = R$ and R is regular.

We now conclude the proof of Theorem 75. As R is regular iff all the $E_w \cap R$ are regular, for $w \in S^{\text{fin}}$, it is sufficient to check whether the latter part is true. Now, for $w \in S^{\text{fin}}$, we can construct a CA for $E_w \cap R$ and we know a socle of $E_w \cap R$ (as we know a socle for E_w); hence Lemma 72 allows to check whether $E_w \cap R$ is regular.

A DetCA is an UnCA; moreover, DetCA are effectively equivalent [KR03] to deterministic extended automata over $(\mathbb{Z}^k, +, \mathbf{0})$ (defined in [MS01]). Thus:

▶ **Corollary 80.** *Given a DetCA or an extended automaton over* $(\mathbb{Z}^k, +, \mathbf{0})$*, it is decidable whether its language is regular.*

5 A deterministic form of UnCA

We present a *deterministic* model equivalent to UnCA. This model is a restriction of the affine Parikh automaton and can be seen as a simple register automaton. As a result of independent interest, we show that CA are equivalent to the nondeterministic variant of this model, and that a seemingly more powerful model (so-called *finite-monoid affine Parikh automata* of Paper II) is in fact equivalent to CA (resp. UnCA) in its nondeterministic (resp. deterministic) form.

▶ **Definition 7** (APA variants)**.** An APA (A, U, C) is said to be:

- *Finite-monoid (FM-APA, FM-DetAPA)* if $\mathscr{M}(U)$ is finite;
- *Moving (M-APA, M-DetAPA)* if for all transition t of A, $U_t = (M, \mathbf{v})$ is such that M is a 0-1-matrix with exactly one 1 per row.

APA: see p. 15; all the APA here are $\mathbb{N}$-APA

We will only consider finite-monoid and moving (Det)APA in the present work. We write $\mathscr{L}_{\text{FM-APA}}$, $\mathscr{L}_{\text{FM-DetAPA}}$, $\mathscr{L}_{\text{M-APA}}$, and $\mathscr{L}_{\text{M-DetAPA}}$ for the classes of languages recognized by FM-APA, FM-DetAPA, M-APA, and M-DetAPA respectively.

Remark. An M-(Det)APA of dimension d can be seen as a finite-state *(deterministic) register automaton* with d registers $r_1, r_2, \dots, r_d$: each transition performs actions of the type $r_i \leftarrow r_{j_i} + k_i$, with $k_i \in \mathbb{N}$, $1 \leq j_i \leq d$, for $1 \leq i \leq d$, and the device accepts iff the underlying automaton accepts and the values of the registers at the end of the computation belong to a prescribed semilinear set.

We first tackle the nondeterministic case, then focus on the deterministic one.

▶ **Theorem 81.** $\mathscr{L}_{\text{CA}} = \mathscr{L}_{\text{M-APA}} = \mathscr{L}_{\text{FM-APA}}$.

Proof. ($\mathscr{L}_{\text{CA}} \subseteq \mathscr{L}_{\text{M-APA}}$.) Given a CA (A, C) where $A = (Q, \Sigma, \delta, q_0, F)$ and $\delta = \{t_1, t_2, \ldots, t_n\}$, we define an M-APA (A, U, C) by setting, for all $t_i \in \delta$, $U_{t_i}(\mathbf{x}) = \mathbf{x} + \mathsf{Pkh}(t_i)$. For a path $\pi \in \delta^*$, we have that $U_\pi(\mathbf{0}) = \mathsf{Pkh}(\pi)$. This implies that $L(A, U, C) = \mu(\{\pi \in \mathsf{Run}(A) \mid \mathsf{Pkh}(\pi) \in C\}) = L(A, C)$, and moreover, that $U_t = (M, \mathbf{v})$ is such that M is the identity matrix, thus (A, U, C) is an M-APA.

($\mathscr{L}_{\text{M-APA}} \subseteq \mathscr{L}_{\text{FM-APA}}$.) Composing 0-1-matrices with exactly one 1 per row results in the same type of matrices. Thus the multiplicative monoid $\mathscr{M}(U)$ of an M-APA (A, U, C) is finite, i.e., (A, U, C) is an FM-APA.

($\mathscr{L}_{\text{FM-APA}} \subseteq \mathscr{L}_{\text{CA}}$.) Let (A, U, C) be an FM-APA with $A = (Q, \Sigma, \delta, q_0, F)$. For $t \in \delta$, we write $U_t = (M_t, \mathbf{v}_t)$, and for $t_1 t_2 \cdots t_n \in \delta^+$, we let $M_{t_1 t_2 \cdots t_n} = M_{t_n} \cdots M_{t_2} \cdot M_{t_1}$. As it is consistent to do, we set $M_\varepsilon = Id$, the identity matrix.

We show that $L(A, U, C)$ can be expressed as the union of the languages of a finite number of CA, and that those CA are unambiguous if A is deterministic. We work in 3 steps. (1.) We start by devising a finite set of automata and show that they recognize the runs π on A while "knowing" M_π (Fact 82). (2.) We show that this extra knowledge allows for the extraction of $U_\pi(\mathbf{0})$ when π is read (Fact 83). Accordingly, we design a semilinear set to constrain this extracted value by C. (3.) We conclude that replacing the labels t of those CA by $\mu_A(t)$ gives a finite set of CA recognizing $L(A, U, C)$. (We could have made this replacement directly within the construction of Step 1 but felt that the proof is best presented taking this extra step.)

Step 1: Automata for the Paths of A. The simplest way to construct an automaton for $\mathsf{Run}(A)$ is by replacing the label of each transition t of A by t itself, i.e., we obtain the automaton $(Q, \delta, \Delta, q_0, F)$ where $t = q \bullet\!\!- a \!\rightarrow q' \in \delta \Leftrightarrow q \bullet\!\!- t \!\rightarrow q' \in \Delta$. This is the first idea of the present construction. The second idea is that we want, when in a state q, all the possible M_π's for π accepted from q to be the same. We will see in Step 2 that this allows for the extraction of the values of U.

Write $\mathscr{M} = \mathscr{M}(U)$. We define, for $q \in Q$ and $M \in \mathscr{M}$:

$$B^{\rightarrow(q,M)} = (Q \times \mathscr{M},\ \delta,\ \Delta,\ (q, M),\ F \times \{M_\varepsilon\})\ ,$$

where:

$$\Delta = \{(q, M) \bullet\!\!- t \!\rightarrow (q', M') \mid t = q \bullet\!\!- \mu(t) \!\rightarrow q' \in \delta \wedge M'.M_t = M\} \ .$$

It is important to note that even if A is deterministic, $B^{\to(q,M)}$ may not be deterministic. Indeed, let Z be the all-zero matrix, and suppose that, for some $t \in \delta$, $M_t = Z$. Then *any* matrix M' verifies $M'.M_t = Z$, thus from the state $(\mathsf{From}(t), Z)$ there is a transition labeled t to *any* state $(\mathsf{To}(t), M')$ for $M' \in \mathscr{M}$. We now show that these automata indeed recognize the paths π in A, while "knowing" M_π. In order to produce a simple statement, write $A^{\to q}$ for A where the initial state is set to q, then:

Fact 82. *For any $q \in Q$ and $M \in \mathscr{M}$, $L(B^{\to(q,M)}) = \{\pi \in \mathsf{Run}(A^{\to q}) \mid M_\pi = M\}$. In particular, $\mathsf{Run}(A) = \bigcup_{M \in \mathscr{M}} L(B^{\to(q_0,M)})$.*

Proof. By induction on $|\pi|$, $\pi \in \delta^*$. Let $L^{(q,M)} = \{\pi \in \mathsf{Run}(A^{\to q}) \mid M_\pi = M\}$.

Let $\pi = \varepsilon$. If $\pi \in L(B^{\to(q,M)})$, then (q, M) is final in $B^{\to(q,M)}$, i.e., $q \in F$ and $M = M_\varepsilon$. The former implies that $\varepsilon \in \mathsf{Run}(A^{\to q})$, and the latter that $M_\pi = M$, concluding the base case in that direction. Conversely, if $\pi \in L^{(q,M)}$, then $q \in F$ and $M = M_\pi = M_\varepsilon$, showing that (q, M) is final in $B^{\to(q,M)}$, hence $\pi \in L(B^{\to(q,M)})$.

Let $\pi \in \delta^*$ such that $|\pi| > 0$ and write $\pi = t\rho$ with $t \in \delta$ and $\rho \in \delta^*$. Suppose $\pi \in L(B^{\to(q,M)})$, so that there is a transition $(q, M) \bullet\!\!- t \!\rightarrow (q', M')$ in Δ with ρ in $L(B^{\to(q',M')})$. By induction hypothesis, $M' = M_\rho$ and ρ is either the empty path, in which case $q' \in F$, or ρ is a path from q' to a state in F. Moreover, by construction $M = M'.M_t$, $q = \mathsf{From}(t)$, and $q' = \mathsf{To}(t)$. Thus $M = M_\rho.M_t = M_\pi$ and π is a path from q to a state in F, i.e., $\pi \in \mathsf{Run}(A^{\to q})$. This concludes this direction. Conversely, suppose $\pi \in L^{(q,M)}$. Then t is a transition from q, and, as $M = M_\pi = M_\rho.M_t$, the transition $(q, M) \bullet\!\!- t \!\rightarrow (\mathsf{To}(t), M_\rho)$ is in Δ. As $\rho \in \mathsf{Run}(A^{\to \mathsf{To}(t)})$, by induction hypothesis $\rho \in L(B^{\to(\mathsf{To}(t),M_\rho)})$, and this implies that $\pi \in L(B^{\to(q,M)})$.

Step 2: Retrieving $U_\pi(\mathbf{0})$. In this step, we argue that our previous construction helps in retrieving the value of $U_\pi(\mathbf{0})$ when π is read over some $B^{\to(q,M)}$. The main ingredient is the following simple property:

$$\text{for } t \in \delta \text{ and } \rho \in \delta^*, \quad U_{t\rho}(\mathbf{0}) = M_\rho.\mathbf{v}_t + U_\rho(\mathbf{0}) \ . \tag{5}$$

We now show a property on paths *over* $B^{\to(q,M)}$. To avoid confusion, elements of Δ^* will be written in capital letters (e.g., Π, P — read "capital rho" — for paths and T for transitions). First, identify Δ with $\{T_1, T_2, \dots, T_n\}$, and each T_i with $(q_i, M_i)\bullet\!\!-t_i\!\!\to(q_i', M_i')$; next, write μ_B for the μ function of one of the $B^{\to(q,M)}$'s — this morphism does not depend on the choice of (q, M). Then:

Fact 83. *For any $q \in Q$, $M \in \mathscr{M}$, and $\Pi \in \mathsf{Run}(B^{\to(q,M)})$:*

$$U_{\mu_B(\Pi)}(\mathbf{0}) = \sum_{i=1}^{n} |\Pi|_{T_i} \times (M_i'.\mathbf{v}_{t_i}) \ .$$

Proof. By induction on $|\Pi|$. Let $q \in Q$ and $M \in \mathscr{M}$.

Let $\Pi = \varepsilon$. Then $U_{\mu_B(\Pi)}(\mathbf{0}) = \mathbf{0}$, and, likewise, the sum of the right-hand side is zero.

Let $\Pi \in \mathsf{Run}(B^{\to(q,M)})$ such that $|\Pi| > 0$ and write $\Pi = T_j\mathrm{P}$ with $T_j \in \Delta$ and $\mathrm{P} \in \Delta^*$. Write $\rho = \mu_B(\mathrm{P})$. Then (5) asserts that:

$$U_{\mu_B(\Pi)}(\mathbf{0}) = U_{t_j\rho}(\mathbf{0}) = M_\rho.\mathbf{v}_{t_j} + U_\rho(\mathbf{0}) \ .$$

Now P is an accepting path of $B^{\to(q_j',M_j')}$, implying that $\rho \in L(B^{\to(q_j',M_j')})$. Fact 82 then says that M_ρ is M_j', hence:

$$\begin{aligned} U_{\mu_B(\Pi)}(\mathbf{0}) &= M_j'.\mathbf{v}_{t_j} + \sum_{i=1}^{n} |\mathrm{P}|_{T_i} \times (M_i'.\mathbf{v}_{t_i}) && \text{(by induction hypothesis)} \\ &= \sum_{i=1}^{n} |\Pi|_{T_i} \times (M_i'.\mathbf{v}_{t_i}) \ . \end{aligned}$$

Now define $C' \subseteq \mathbb{N}^n$ by $(x_1, x_2, \dots, x_n) \in C' \Leftrightarrow \left(\sum_{i=1}^n x_i \times (M_i'.\mathbf{v}_{t_i})\right) \in C$. Fact 82 and Fact 83 imply that, for $q \in Q$ and $M \in \mathscr{M}$:

$$L(B^{\to(q,M)}, C') = \{\pi \in \mathsf{Run}(A^{\to q}) \mid M_\pi = M \wedge U_\pi(\mathbf{0}) \in C\} \ . \tag{6}$$

Step 3: from Paths to their Labels. For $q \in Q$ and $M \in \mathscr{M}$, define $D^{\to(q,M)}$ to be the automaton $B^{\to(q,M)}$ where a transition labeled t in $B^{\to(q,M)}$ is relabeled $\mu_A(t)$ in $D^{\to(q,M)}$. Equation (6) then implies:

$$L(D^{\to(q,M)}, C') = \{\mu(\pi) \in \mathsf{Run}(A^{\to q}) \mid M_\pi = M \wedge U_\pi(\mathbf{0}) \in C\} \ .$$

Since $\mathsf{Run}(A) = \bigcup_{M\in\mathscr{M}} B^{\to(q_0,M)}$, we have:

$$L(A, U, C) = \bigcup_{M\in\mathscr{M}} L(D^{\to(q_0,M)}, C') \ .$$

As $\mathscr{M}$ is finite by hypothesis, $L(A, U, C)$ is the finite union of CA languages. The closure of $\mathscr{L}_{\text{CA}}$ under union [KR03] implies that $L(A, U, C) \in \mathscr{L}_{\text{CA}}$.

▶ **Theorem 84.** $\mathscr{L}_{\text{UnCA}} = \mathscr{L}_{\text{M-DetAPA}} = \mathscr{L}_{\text{FM-DetAPA}}$.

Proof. ($\mathscr{L}_{\text{UnCA}} \subseteq \mathscr{L}_{\text{M-DetAPA}}$.) It is shown in Paper II, Lemma 48 that $\mathscr{L}_{\text{UnCA}} \subseteq \mathscr{L}_{\text{FM-DetAPA}}$.[8] Therein, the remark is made that the matrices used to construct an FM-DetAPA from a UnCA are 0-1-matrices with *at most* one 1 per row. It is easily seen that we can use 0-1-matrices with *exactly* one 1 per row. Indeed, let (A, U, C) be an FM-DetAPA of dimension d, with δ the transition set of A, and for all $t \in \delta$, $U_t = (M, \mathbf{v})$ is such that M is a 0-1-matrix with at most one 1 per row. Then define (A, U', C') as an M-DetAPA of dimension $d+1$ with the same language as follows. Let $t \in \delta$ and $U_t = (M, \mathbf{v})$, and let $\mathbf{r}_i \in \mathbb{N}^d$, $1 \le i \le d$, be the i-th row of M. Define $M' \in \mathbb{N}^{(d+1)\times(d+1)}$ where the i-th row of M', $1 \le i \le d$, is $(\mathbf{r}_i, 0)$ if $\mathbf{r}_i$ has a 1 and $\mathbf{e}_{d+1}$ otherwise; and the $d+1$-th row of M' is $\mathbf{e}_{d+1}$. We then let $U'_t = (M', (\mathbf{v}, 0))$. Thus for a path π in A, $U'_\pi(0^{d+1}) = (U_\pi(0^d), 0)$. We accordingly set $C' = C \times \{0\}$, and thus $L(A, U, C) = L(A, U', C')$, and (A, U', C') is an M-DetAPA.

($\mathscr{L}_{\text{M-DetAPA}} \subseteq \mathscr{L}_{\text{FM-DetAPA}}$.) As in the proof of Theorem 81, an M-DetAPA is an FM-DetAPA.

($\mathscr{L}_{\text{FM-DetAPA}} \subseteq \mathscr{L}_{\text{UnCA}}$.) We add an extra step to the proof of the inclusion $\mathscr{L}_{\text{FM-APA}} \subseteq \mathscr{L}_{\text{CA}}$ of Theorem 81. Using the same notations, we show that if A is deterministic, then the CA constructed are unambiguous. $\mathscr{L}_{\text{UnCA}}$ being closed under union (Proposition 56) this proves the inclusion.

Step 4: from APA Determinism to CA Unambiguity. Suppose that A is deterministic. We argue that for any $q \in Q$ and $M \in \mathscr{M}$, $D^{\to(q,M)}$ is unambiguous. We show by induction on the size of $w \in L(D^{\to(q,M)})$ that there is only one accepting path labeled w in $D^{\to(q,M)}$. If $w = \varepsilon$, then ε is the only path labeled w (recall that $D^{\to(q,M)}$ has no ε-transition). If $|w| > 0$, then write $w = av$ where $a \in \Sigma$ and $v \in \Sigma^*$. As $A^{\to q}$ is deterministic, there is only one t labeling a transition leaving (q, M) in $B^{\to(q,M)}$

[8]In Paper II, so-called *constraint-deterministic CA* are used instead of UnCA; it is easily seen that UnCA are a special case of constraint-deterministic CA.

with $\mu_A(t) = a$. Thus the transitions labeled a leaving (q, M) in $D^{\to(q,M)}$ are of the form $(q, M_\rho.M_t) \bullet\!\!-a\!\!\to (q', M_\rho)$ where $M_\rho.M_t = M$ and $q' = \mathsf{To}(t)$. Now there is only one path π in $A^{\to q'}$ recognizing v, implying that there is only one M' such that $v \in L(D^{\to(q',M')})$, in fact $M' = M_\pi$. Hence only the transition $(q, M) \bullet\!\!-a\!\!\to (q', M')$ can be taken to recognize w. The induction hypothesis then asserts that only one accepting path from (q', M') is labeled v, thus only one accepting path from (q, M) is labeled w.

Remark. Theorems 81 and 84 are *effective*, in the sense that one can go from one model to another following an algorithm. This implies in particular, from Theorem 75 that regularity is decidable for FM-DetAPA; we note that it is not decidable for DetAPA (see Paper I), which describes a class of languages strictly larger than that of UnCA though expected to be incomparable with that of CA.

6 Conclusion

We showed that $\mathscr{L}_{\text{UnCA}}$ is a class of languages that is closed under the Boolean operations, inverse morphisms, commutative closure, reversal, and right and left quotient, and that provably fails to be closed under concatenation with a regular language, length-preserving morphisms, and starring. Further, the following problems are decidable for $\mathscr{L}_{\text{UnCA}}$: emptiness, universality, finiteness, inclusion, and regularity. Deciding regularity for UnCA and DetCA is our main result.

We propose three future research avenues. First, the properties of UnCA indicate its suitability for model-checking, and we could envisage real-world applications of verification using UnCA. Second, we translated unambiguous CA to a natural model of *deterministic* register automata; the close inspection of this translation can lead to further advances in our understanding of unambiguity, in particular in the open problems dealing with unambiguous finite automata [Col12]. Third, we note that the closure properties of $\mathscr{L}_{\text{UnCA}}$ imply that this class can be described by a natural algebraic object (see [BKR11]). This will certainly help in linking UnCA to a first-order logic framework, and thus to some Boolean circuit classes. Hence we hope that UnCA can shed a new light on the classes of circuit complexity.

Acknowledgement

We thank Andreas Krebs for stimulating discussions and comments concerning this work and the anonymous referees for their careful reading. The first author thanks Benno Salwey and Dave Touchette for comments on early versions of this paper.

Discussion

The present paper gives a thorough study of the closure, decidability, and expressiveness properties of: constraint-deterministic CA (of Paper II), unambiguous CA, finite-monoid DetAPA, and moving DetAPA, while showing that all these models express the same languages.

The main technical result of this paper, the decidability of regularity, uses a technique to reduce the problem in the general case to the problem with bounded languages, which is usually easier to work with. This result was the starting point of this paper. While formalizing this result in the context of DetCA, it occurred to us that the only property needed for it to work was that of Lemma 74, i.e., that $L(A, C)$ is regular iff $\mathsf{Run}(A) \upharpoonright_C$ is regular. Unambiguous CA then appeared to us as the model with the largest expressive power verifying this property.

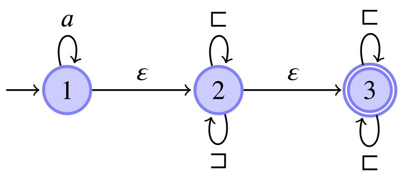
We note that we relied on the Dyck languages to prove separation of UnCA and CA. It would feel more natural to separate those classes using expressiveness lemmata, instead of closure properties. In particular, the language $\{a, b\}^* \cdot \{a^n b^n\} \cdot \{a, b\}^*$ appears to be outside of $\mathscr{L}_{\text{UnCA}}$, and it would separate more accurately, or rather, more intuitively, the two classes.

This paper did not rely on the definition of Parikh automata; however, we can devise a definition for unambiguous Parikh automata. A Parikh automaton (A, C) over Σ is unambiguous if A is unambiguous and for any word $w \in \Sigma^*$, there is at most one $\omega \in L(A)$ with $\Phi(\omega) = w$. It is easily seen that such unambiguous PA are equivalent to UnCA, using arguments similar to Paper I, Theorem 6.

An interesting further model to study, as suggested by Andreas Krebs, is that of a nondeterministic CA for which the Parikh image of *at most* one accepting path per word belongs to the constraint set. Let us use call *one-success* (OneCA) those constrained automata, and write $\mathscr{L}_{\text{OneCA}}$ for the class of languages they recognized. Clearly, $\mathscr{L}_{\text{UnCA}} \subseteq \mathscr{L}_{\text{OneCA}}$. Now, this inclusion is strict, as the language:

$$L = \{a^n w \mid w \in \{\sqsubset, \sqsupset\}^* \wedge |w_1 w_2 \cdots w_n|_{\sqsubset} < |w_1 w_2 \cdots w_n|_{\sqsupset}\} \ ,$$

of Proposition 67 is expressible by a OneCA, and is not expressible by UnCA. This latter fact is proved in Proposition 67, while expressivity by OneCA is shown by the following construction. Let A be described by:



and define the set $C \subseteq \mathbb{N}^7$ to constrain the number of a's read on state 1 to be the number of $\sqsubset$'s and $\sqsupset$'s read on state 2; it should moreover check that the number of $\sqsubset$'s read on 2 is strictly smaller than the number of $\sqsupset$'s read there.

Now, it is clear that $L(A, C) = L$. Also, this is a OneCA: the first nondeterministic choice on ε can only be taken at the end of the a's, and the only way to read the input word that is accepted by C is when the number of characters read on 2 is exactly the number of a's. Thus $\mathscr{L}_{\text{UnCA}} \subsetneq \mathscr{L}_{\text{OneCA}}$.

On the other hand, we conjecture that $\mathscr{L}_{\text{OneCA}} \subsetneq \mathscr{L}_{\text{CA}}$. Languages in $\mathscr{L}_{\text{CA}}$ not belonging to $\mathscr{L}_{\text{OneCA}}$ would be languages with possibly multiple witnesses of membership (i.e., accepting splitting) for words; for instance, $\overline{P_1}$, where the word $w = \sqsupset\sqsupset$ has two witnesses of membership: the first position, and the second one, both positions i verifying $|w_1 \cdots w_i|_{\sqsubset} < |w_1 \cdots w_i|_{\sqsupset}$. The language $\{a, b\}^* \cdot \{a^n b^n \mid n \in \mathbb{N}\} \cdot \{a, b\}^*$ is another example.

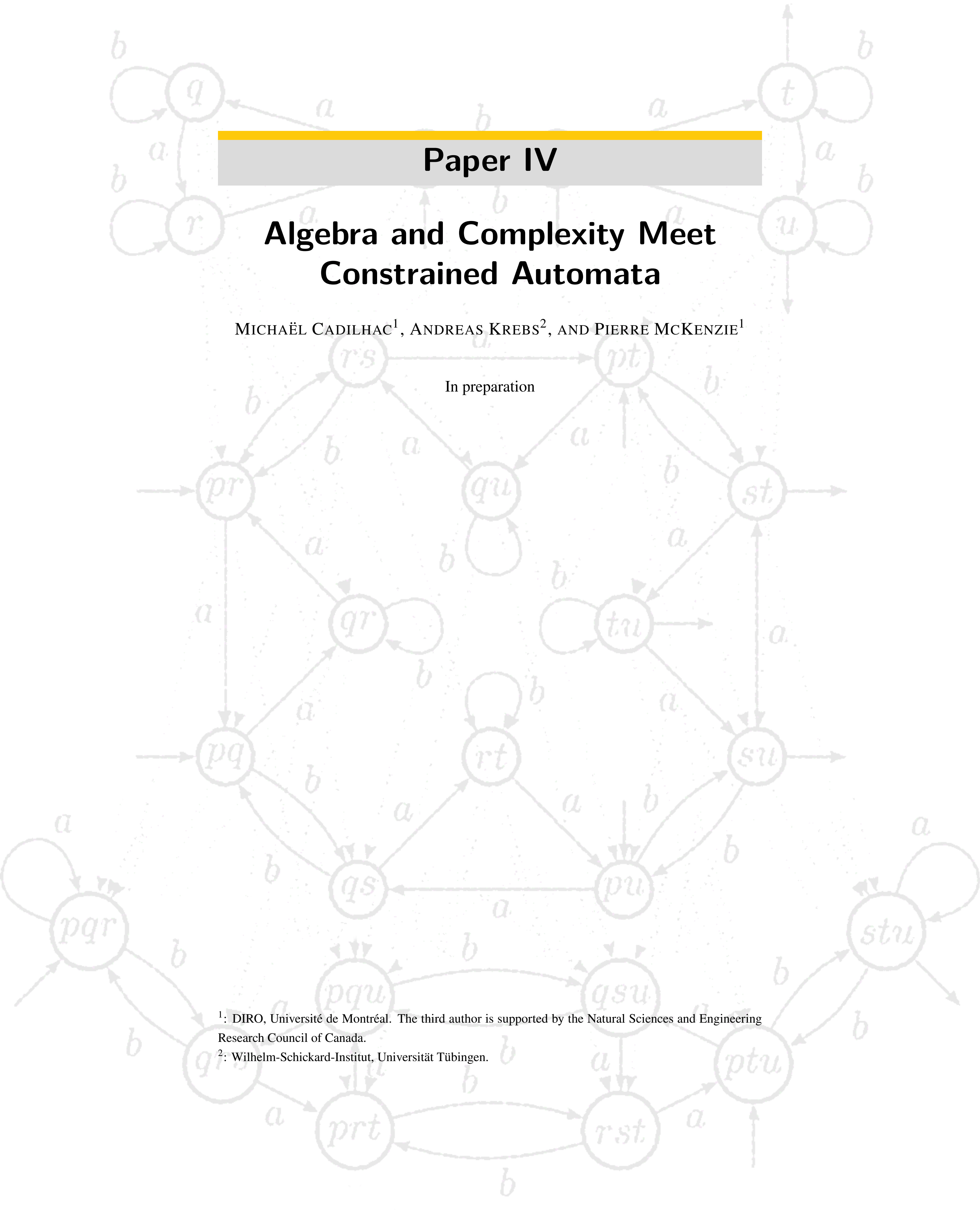
Further, we note that this model does not verify Lemma 74. Of course, for any OneCA (A, C), if $\mathsf{Run}(A)\restriction_C$ is regular, then $L(A, C)$ is regular — this is true of any CA. Now the Remark appearing on p. 106 presents a CA (A, C) with the property that $L(A, C)$ is regular while $\mathsf{Run}(A)\restriction_C$ is not: this CA is in fact a OneCA. However, the proof that regularity is undecidable of CA (Proposition I.17) seems not to apply for OneCA: it would require $\mathscr{L}_{\text{OneCA}}$ to be closed under inverse rational transduction, which is equivalent to being closed under morphisms, inverse morphisms, and intersections with regular languages, and this would imply that $\mathscr{L}_{\text{OneCA}} = \mathscr{L}_{\text{CA}}$.

This leaves interesting questions for further study.

Another natural model in the vein of unambiguity is the unambiguous APA (UnAPA), i.e., APA (A, U, C) where A is unambiguous. We note that most closure proofs of the present paper work equally well for UnAPA, showing:

▶ **Proposition 85.** *The class of languages recognized by UnAPA is closed under the Boolean operations and inverse morphisms.*

The case of reversal is tackled in Paper IV, Corollary 96, and further study of this model is done in Paper IV.



Paper IV

Algebra and Complexity Meet Constrained Automata

Michaël Cadilhac[1], Andreas Krebs[2], and Pierre McKenzie[1]

In preparation

[1]: DIRO, Université de Montréal. The third author is supported by the Natural Sciences and Engineering Research Council of Canada.
[2]: Wilhelm-Schickard-Institut, Universität Tübingen.

Presentation

When tackling decidability questions within a model of automata, one possible endeavor is to characterize the model using algebraic tools. In language theory, we say that a language $L \subseteq \Sigma^*$ is recognized by a monoid M if there is a morphism $h\colon \Sigma^* \to M$ such that $L = h^{-1}(S)$ for some $S \subseteq M$. In the 1960s, it had long been known that the regular languages correspond to the languages recognized by finite monoids, but the notion of recognizability only gained widespread acceptance after the appearance of Schützenberger's theorem (see [MP71, p. 99] for an historical account). Indeed, Schützenberger's theorem, which states that the star-free languages are those recognized by aperiodic monoids, provided the first example of an open decidability question solved with algebraic tools — in this case, whether it is possible to decide if a regular language is star-free.

Following this trend, our first goal with the present paper is to characterize classes of languages recognized by CA and APA in an algebraic framework. However, the classical theory is of no use when considering nonregular languages. For instance, the smallest monoid recognizing $\{w \mid |w|_a = |w|_b\}$ is isomorphic to $(\mathbb{Z}, +)$ and already this monoid recognizes undecidable languages. Instead, we turn to *typed monoids*, an algebraic structure introduced by Krebs, Lange, and Reifferscheid [KLR07] precisely to get around this problem. It was our hope that this would help in showing the decidability of regularity for DetCA, which we solved using different techniques in Paper III.

Our second goal is to tackle the question of the separation of DetAPA and APA. We take two paths to provide answers to this question. First, we consider separation over the *unary languages*. Indeed, as the nonregular language $\{a^{2^n} \mid n \in \mathbb{N}\}$ is in $\mathscr{L}_{\mathrm{PA}}$, showing that the only unary languages that DetAPA recognize are regular would imply the above separation. In fact, this property of DetAPA does not hold: using a characterization of DetAPA by means of rational power series, we show that there are nonregular unary languages in $\mathscr{L}_{\text{DetAPA}}$. Second, we rely on the simulation of Turing machines by APA of Paper I to show that, given natural computational complexity assumptions, DetAPA and APA define distinct classes of languages. Along this latter path, we show some new unconditional closure properties.

We use the typed variants of the wreath and block products in our characterizations. For the sake of completeness, we now give the definitions of the unilateral and bilateral semidirect products of monoids; these are central to the wreath and block product. See [TT07] for a survey of the links between algebra and language theory pertaining to our study, including the following definitions.

Let M and N be finite monoids. To distinguish the operation of M and N, we denote the operation of M as $+$ and its identity element as 0 (although this operation is not necessarily commutative) and the operation of N implicitly and its identity element as 1. A left action of N on M is a function mapping pairs $(n, m) \in N \times M$ to $nm \in M$ and satisfying $n(m_1 + m_2) = nm_1 + nm_2$, $n_1(n_2m) = (n_1n_2)m$, $n0 = 0$ and $1m = m$. Right actions are defined symmetrically. If we have both a right and a left action of N on M that further satisfy $n_1(mn_2) = (n_1m)n_2$, we define the *bilateral semidirect product* $M ** N$ as the monoid with elements in $M \times N$ and multiplication defined as $(m_1, n_1)(m_2, n_2) = (m_1n_2 + n_1m_2, n_1n_2)$. This operation is associative and $(0, 1)$ acts as an identity for it:

$$\begin{aligned}
(m_1, n_1)((m_2, n_2)(m_3, n_3)) &= (m_1, n_1)(m_2n_3 + n_2m_3, n_2n_3)\\
&= (m_1n_2n_3 + n_1m_2n_3 + n_1n_2m_3, n_1n_2n_3)\\
&= (m_1n_2 + n_1m_2, n_1n_2)(m_3, n_3)\\
&= ((m_1, n_1)(m_2, n_2))(m_3, n_3)\ ,\\
(0, 1)(m, n) &= (0n + 1m, 1n) = (m, n)\ ,\\
(m, n)(0, 1) &= (m1 + n0, n1) = (m, n)\ .
\end{aligned}$$

Given only a left action, the *unilateral semidirect product* $M*N$ is the bilateral semidirect product $M ** N$ where the right action on M is trivial ($mn = m$).

Now, the fundamental property of the iterated multiplication is the following. Let $(m_1, n_1), (m_2, n_2), \ldots, (m_k, n_k) \in S ** T$. Then:

$$\prod_{i=1}^{k}(m_i, n_i) = \left(\sum_{i=1}^{k}(n_1 \cdots n_{i-1}) \cdot m_i \cdot (n_{i+1} \cdots n_k), n_1n_2 \cdots n_k\right)\ .$$

This is clear for $k = 1$, taking $(n_1 \cdots n_{k-1})$ to be 1. Now associativity allows us to bracket the well-defined $\prod_{i=1}^{k}(m_i, n_i)$ in the most favorable way, which in this case is to write it as $\left(\prod_{i=1}^{k-1}(m_i, n_i)\right)(m_k, n_k)$; then the induction step is direct from the definition.

Personal contribution. Most of the results on the typed monoid characterizations and the conditional separations were worked out by Krebs and me during a stay at Tübingen. Krebs remarked the link between DetAPA and weighted automata that appears in a simplified form here. I formalized and wrote the proofs, with the help of McKenzie. The paper is still in preparation.

Algebra and Complexity Meet Constrained Automata

Abstract

We propose several algebraic characterizations of models related to the constrained automaton, from which we derive expressiveness and complexity results. We also investigate how computational complexity assumptions help in solving the open problem of separating deterministic and nondeterministic affine Parikh automata.

Introduction

The works of Büchi [Büc60] and Schützenberger [Sch65] unveiled the first of many fascinating correspondences between formal language theory (and its models), algebra, logic, and circuit complexity. The pursuit of such correspondences has led to even more remarkable results, including separation and decidability properties (see, e.g., [Str94]). In particular, it appears that an algebraic characterization of a model of computation extracts the very essence of its abilities and limitations, which are then described as a set of (pseudo)identities.

Constrained automata [KR03] were introduced in the context of model-checking by Klaedtke and Rueß[9] as a model whose expressive power lies between regular and context-sensitive languages. A *Constrained Automaton* (CA) is given by an automaton A and a semilinear set C; its language is the set of labels of the accepting paths π of A such that the Parikh image of π is in C. This model has been extended (Paper I) to *Affine Parikh Automata* (APA). An APA is given by an automaton A, a function U

[9]In [KR03], the model under study is called *Parikh automata*. Constrained automata are but an effectively equivalent model with an arguably simpler definition.

labeling the transitions of A by affine functions, and a semilinear set C; its language is the set of labels of the accepting paths π of A for which the successive composition of the affine functions given by U maps the all-zero vector to a vector in C. In this paper, we present an algebraic view on CA and APA, and place these models in their natural complexity context.

However, in order to give an algebraic characterization of a class of nonregular languages, a careful extension to the classical algebraic theory of regular languages has to be made. Indeed, in the classical framework, the syntactic monoid of the language of words having more a's than b's is $\mathbb{Z}$, but any unary language — even undecidable ones — can be recognized by $\mathbb{Z}$ with a suitable accepting set. A solution to this problem, *typed monoids*, has been introduced by Krebs, Lange and Reifferscheid [KLR07]. These are (possibly infinite) monoids paired with a finite Boolean algebra of subsets of the monoid called *types*; a language is recognized by a typed monoid if it is the inverse morphic image of a type. Using this framework, they characterized some circuit complexity classes, and unveiled an algebra–circuits–logic equivalence. The intuition given by the algebraic characterizations helped in showing separation results in some logics [BKR11]. In our present work, we show that the framework of typed monoids is also able to naturally characterize classes of languages coming from automata theory. This has two main benefits. On the one hand, this provides logical and circuit characterizations of some of our classes of automata, and on the other hand, this makes available the tools of algebra to show additional results.

The main results of this paper are threefold. First, we characterize deterministic and unambiguous CA and APA using the theory of typed monoids, and deterministic APA using rational power series. Second, we show that unambiguous CA correspond to a natural logic, and deduce circuit complexity upper bounds for the class of languages they recognize. Third, we investigate possible ways to separate deterministic APA and APA by showing some closure properties of these models, some of them depending on computational complexity assumptions.

Section 1 in this paper contains preliminaries. Section 3 investigates the typed monoid characterizations and provides, as applications, a logical characterization of unambiguous CA, and a closure property of deterministic and unambiguous APA. Section 4 describes deterministic APA using power series and show a separation between deterministic APA and CA over *unary* languages. Section 5 gives conditional separations of APA and deterministic APA, and shows some additional (non)closure properties of deterministic APA and APA.

1 Preliminaries

We write $\mathbb{Z}_0^-$ for the set of nonpositive integers. For a monoid M, we write e_M for the identity element of M. We let $\mathscr{M}_{\mathbb{Z}}(k)$, for $k \geq 1$, be the monoid of square matrices of dimension $k \times k$ with values in $\mathbb{Z}$ and with the operation being the inverted matrix multiplication. That is, a morphism $h\colon \{a,b\}^* \to \mathscr{M}_{\mathbb{Z}}(k)$ is such that $h(ab) = h(b).h(a)$ with . the usual matrix multiplication. We write Ψ_i for the projection on the i-th component.

Wreath and block products. Let S, T be two monoids. The *wreath product* of S and T, written $S \wr T$, is defined as the unilateral semidirect product of S^T and T, where the left action of T on S^T is given by $(t \cdot f)(t') = f(t't)$, for $f : T \to S$ and $t, t' \in T$. The *block product* of S and T, written $S \Box T$, is defined as the bilateral semidirect product of $S^{T\times T}$ and T, where the right (resp. left) action of T on $S^{T\times T}$ is given by $(f \cdot t)(t_1, t_2) = f(t_1, tt_2)$ (resp. $(t \cdot f)(t_1, t_2) = f(t_1 t, t_2)$), for $f : T \times T \to S$ and $t, t_1, t_2 \in T$.

Uni/bi-lateral semidirect product: see p. 124

Transition monoid. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete deterministic automaton. For $a \in \Sigma$, define $f_a\colon Q \to Q$ by $f_a(q) = q'$ iff $q \bullet\!\!-a\!\!\to q' \in \delta$. The *transition monoid* M of A is the closure under composition of the set $\{f_a \mid a \in \Sigma\}$. The monoid M acts on Q naturally by $q^m = m(q)$, $m \in M$, $q \in Q$. Write $\eta\colon \Sigma^* \to M$ for the canonical surjective morphism associated, that is, the morphism defined by $\eta(a) = f_a$, $a \in \Sigma$. Then $q^{\eta(w)}$ is the state reached by reading $w \in \Sigma^*$ from the state $q \in Q$.

2 Two normal forms on CA and APA

We present several technical lemmata on CA and APA, that will help us in devising concise proofs for the algebraic characterizations that follow.

CA, APA: see pp. 14, 15; all the APA here are $\mathbb{N}$-APA

Recall (e.g., [End72]) that for any semilinear set $C \subseteq \mathbb{Z}^d$, there is a Boolean combination of expressions of the form: $\sum_{1\leq i\leq d} \alpha_i x_i > c$ and $\sum_{1\leq i\leq d} \alpha_i x_i \equiv_p c$, with $\alpha_i, c \in \mathbb{Z}$ and $p > 1$, which is true iff $(x_1, x_2, \ldots, x_d) \in C$. Note that the α_i may be zero. We define two notions which refine this point of view:

▶ **Definition 8.** We say that a semilinear set C is *modulo-free* if it can be expressed as a Boolean combination of expressions of the form $\sum_i \alpha_i x_i > c$, for $\alpha_i \in \mathbb{Z}$. We say that C is *basic* if it can further be expressed as a *positive* Boolean combination of expressions of the form $\sum_i \alpha_i x_i > 0$.

▶ **Lemma 86.** *If $C \subseteq \mathbb{Z}^d$ is modulo-free, then for all $1 \leq d' < d$ there is a basic set $B \subseteq \mathbb{Z}^d$ such that for all $\mathbf{x} \in \mathbb{Z}^{d'}$ and $1 \leq i \leq d - d'$:*

$$(\mathbf{x}, \mathbf{e}_i) \in B \Leftrightarrow (\mathbf{x}, \mathbf{e}_i) \in C \enspace .$$

Proof. Let C be expressed as a Boolean combination of expressions of the form $\sum_i \alpha_i x_i > c$. Let the negations be pushed to the lowest level of the Boolean combination, now:

$$\neg\left(\sum_t \alpha_t x_t > c\right) \sim \sum_i \alpha_i x_i < c + 1 \sim \sum_i (-\alpha_i) x_i > -(c+1) \enspace ,$$

thus we may suppose that C is a *positive* Boolean combination. Now replace each expression $\sum_i \alpha_i x_i > c$ by $\sum_i \alpha'_i x_i > c$ where $\alpha'_i = \alpha_i$ if $i \leq d'$ and $\alpha'_i = \alpha_i - c$ otherwise. Thus let $\mathbf{x} \in \mathbb{Z}^{d'} \times \{\mathbf{e}_i \mid 1 \leq i \leq d - d'\}$. It holds that $\sum_i \alpha_i x_i > c$ iff $\sum_i \alpha'_i x_i > 0$, thus the set resulting from the replacement is a basic set with the property of the statement of the lemma.

The first normal form concerns DetCA and UnCA:

▶ **Lemma 87.** *Every DetCA (resp. UnCA) has the same language $L \subseteq \Sigma^+$ as another DetCA (resp. UnCA) (A, C) with $L(A) = \Sigma^*$ and C a basic set.*

Proof. Let (A, C) be a CA. Let C be expressed as a Boolean combination of expressions that use the relation symbols $<$, $\equiv_p$, function symbol $+$, and constants from $\mathbb{N}$. We first remove the $\equiv_p$ relation symbols. Consider an expression in the definition of C of the form $\sum_t \alpha_t \times x_t \equiv_p c$, for $\alpha_t \in \mathbb{Z}$, where the sum ranges over all transitions. Then define A' to be p copies $A_0, A_1, \ldots, A_{p-1}$ of A, the initial state being that of A_0, and the final states being the final states of all the A_i's. On taking a transition t of A in the copy A_i, the automaton A' jumps to the copy $A_{i+\alpha_t \pmod p}$. Thus A' ends its computation in the copy A_c iff the numbers x_t of times each transition t of A has been taken in any copy are such that $\sum_t \alpha_t \times x_t \equiv_p c$. Now, given $\mathsf{Pkh}(\pi)$ for a path π, a basic set can check in which state π ended (this is the only state entered more often than exited, see, e.g., Paper III, Fact 55), thus the expression under consideration can be replaced by an expression checking whether the path ended in A_c. The other expressions are simply adjusted to be oblivious to the copies. By induction, we may suppose that C is thus defined without the relations $\equiv_p$, i.e., that C is modulo-free.

Thus let (A, C) be a CA with C modulo-free. We rely on Lemma 86 to turn C into a basic set. Write $A = (Q, \Sigma, \delta, q_0, F)$, then define $A' = (Q \cup \{s\}, \Sigma, \delta', s, F)$

where $s \notin Q$, $\delta' = \delta \cup T$ with $T = \{s \xrightarrow{a} q \mid q_0 \xrightarrow{a} q \in \delta\}$. Order δ' so that the transitions of T appear last. Now define C' as being C where each expression $\sum_{t\in\delta} \alpha_t x_t > c$ is replaced by $\sum_{t\in\delta'} \alpha_t x_t > 0$ where α_t for $t = s \xrightarrow{a} q \in T$ is $\alpha_{q_0 \xrightarrow{a} q}$. Clearly $L(A, C) = L(A', C')$. Now exactly one of the transitions in T will be taken in a nonempty run in A', thus $L(A', C') = L(A, C' \cap \{\mathbf{e}_i \mid 1 \leq i \leq |T|\})$, and by Lemma 86, there is a basic set B such that $L(A', C') = L(A', B)$.

Note that all the operations made on the automaton part of the CA do not change whether it is deterministic or unambiguous. We now make sure that the language of the underlying automaton is Σ^*.

Let (A, C) be a DetCA with C basic. Following Karianto [Kar04], we can set all the states of A to be final if we tweak C to check that the ending state of the path is final in A. This is equivalent to checking which state was entered more often than exited, which is expressed, for a state q different from the initial state as $\sum_{t\in q^-} x_t - \sum_{t\in q^+} x_t > 0$, where q^- (resp. q^+) is the set of transitions going to (resp. leaving) the state q. Thus the set K of Parikh images of accepting paths in A is basic, and we can replace C by $C \cap K$, a basic set.

For the UnCA case, this is shown in Paper III, Fact 57. Note that the modification made to C (which is essentially to replace it with $C \times \mathbb{Z}^+$) preserves the fact that the constraint set is basic.

We also note the following simple fact:

▶ **Lemma 88.** *For $(A, C_1 \cap C_2)$ a DetCA or an UnCA it holds that:*

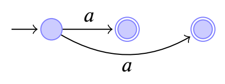
$$L(A, C_1 \cap C_2) = L(A, C_1) \cap L(A, C_2) \ .$$

The same holds for $\cup$.

Proof. For $\cup$, this is true of DetCA, UnCA, and CA. For $\cap$, the inclusion from left to right holds for the three models; the converse inclusion only holds for DetCA and UnCA. Let $w \in L(A, C_1) \cap L(A, C_2)$ for A a deterministic or unambiguous automaton. Then there is only one accepting path π in A with label w, and thus $\mathsf{Pkh}(\pi) \in C_1 \cap C_2$, and in turn $w \in L(A, C_1 \cap C_2)$.

Example 17

Let A be the nondeterministic automaton:



and $C_1 = \binom{1}{0}$, $C_2 = \binom{0}{1}$, then $L(A, C_i) = \{a\}$, $i = 1, 2$, and $L(A, C_1 \cap C_2) = \emptyset$.

We show more in the context of APA to allow the forthcoming proofs of characterization to translate smoothly from CA to APA. In the following, we consider that a matrix $M \in \mathscr{M}_{\mathbb{Z}}(k)$ is in a set $C \subseteq \mathbb{Z}^{k^2}$ if the vector consisting of the columns of M is in C.

▶ **Lemma 89.** *Let $L \subseteq \Sigma^+$ be in $\mathscr{L}_{\text{DetAPA}}$. There is a morphism $h\colon \Sigma^* \to \mathscr{M}_{\mathbb{Z}}(k)$, for some k, and a set $\mathcal{X} \subseteq \mathbb{Z}^{k^2}$ expressible as a Boolean combination of expressions $x_i > 0$, such that $L = h^{-1}(\mathcal{X})$.*

Similarly, let $L \subseteq \Sigma^+$ be in $\mathscr{L}_{\text{APA}}$ (resp. in $\mathscr{L}_{\text{UnAPA}}$). There is an automaton (resp. unambiguous automaton) A with transition set δ, a morphism $h\colon \delta^* \to \mathscr{M}_{\mathbb{Z}}(k)$, for some k, and a set $\mathcal{X} \subseteq \mathbb{Z}^{k^2}$ expressible as a Boolean combination of expressions $x_i > 0$, such that $L = \mu_A(h^{-1}(\mathcal{X}) \cap \mathsf{Run}(A))$.

Proof. Using Paper I, Lemma 21, then Paper I, Proposition 38, we obtain that for any $L \subseteq \Sigma^+$ in $\mathscr{L}_{\text{DetAPA}}$, there is a morphism $g : \Sigma^* \to \mathscr{M}_{\mathbb{Z}}(n)$, for some n, a vector $\mathbf{s} \in \mathbb{Z}^k$, and a modulo-free set C such that:

$$L = \{w \mid g(w).\mathbf{s} \in C\} \ .$$

Similarly, using Paper I, Lemma 22, then Paper I, Lemma 21, we obtain that for any $L \subseteq \Sigma^+$ in $\mathscr{L}_{\text{APA}}$ (resp. $\mathscr{L}_{\text{UnAPA}}$), there is an automaton (resp. unambiguous automaton) A with transition set δ, a morphism $g\colon \delta^* \to \mathscr{M}_{\mathbb{Z}}(n)$, for some n, a vector $\mathbf{s} \in \mathbb{Z}^k$, and a modulo-free set C such that:

$$L = \mu_A(\{\pi \in \mathsf{Run}(A) \mid g(\pi).\mathbf{s} \in C\}) \ .$$

The rest of this proof will focus on reaching the statement of the lemma for DetAPA; the proof is identical for UnAPA and APA, as A is left unchanged.

We first show that we can turn C into a basic set using Lemma 86. Extend g to a morphism $g' : \Sigma^* \to \mathscr{M}_{\mathbb{Z}}(n+1)$ defined by:

$$g'(a) = \begin{pmatrix} \boxed{g(a)} & \begin{matrix}0\\ \vdots\end{matrix} \\ 0 \ \cdots & 1 \end{pmatrix} ,$$

and let $\mathbf{s}'$ be the vector $(\mathbf{s}, 1)$. Then for any nonempty path π in A, $g'(\pi).\mathbf{s}' = (g(\pi).\mathbf{s}, 1)$. Moreover, by Lemma 86, there is a basic set B such that $B \cap \mathbb{Z}^n \times \{1\} = C \times \{1\}$. Thus $g(\pi).\mathbf{s} \in C$ iff $g'(\pi).\mathbf{s}' \in B$.

Thus we suppose that C is a basic set. We show that we can turn C into a set of dimension n' expressible as a Boolean combination of expressions $x_i > 0$. Suppose $\sum_i \alpha_i x_i > 0$ appears in the expression defining C. We extend g so that this sum is computed by the matrices. For $a \in \Sigma$, write the lines of $g(a)$ as $L_1, L_2, \ldots, L_n$. Then define $g' : \Sigma^* \to \mathscr{M}_{\mathbb{Z}}(n+1)$ as the morphism mapping $a \in \Sigma$ to the matrix consisting of lines $(L_1, 0), (L_2, 0), \ldots, (L_n, 0), (\sum_{i \le n} \alpha_i L_i, 0)$. This is such that, for

$w \in \Sigma^*$, the last component of $g(w).(\mathbf{s}, 0)$ is indeed the sum under consideration. Thus the formula $\sum_i \alpha_i x_i > 0$ can be replaced by $x_{n+1} > 0$. Proceeding inductively results in a set $C \subseteq \mathbb{Z}^{n'}$ expressible as a Boolean combination of expressions of the form $x_i > 0$.

Thus we suppose that C is expressible as a Boolean combination of expressions $x_i > 0$. We show that the product $g(w).\mathbf{s}$ can be computed within the matrices. For $a \in \Sigma$, write the lines of $g(a)$ as $L_1, L_2, \ldots, L_n$, and define $h \colon \Sigma^* \to \mathcal{M}_{\mathbb{Z}}(n+1)$ as the morphism mapping $a \in \Sigma$ to the matrix consisting of lines $(L_1, L_1.\mathbf{s}), (L_2, L_2.\mathbf{s}), \ldots, (L_n, L_n.\mathbf{s}), \mathbf{0}$. Then for $w \in \Sigma^+$, we have that $h(w).\mathbf{e}_{n+1} = (g(w).\mathbf{s}, 0)$. Thus let $\mathscr{Z} \subseteq \mathbb{Z}^{(n+1)\times(n+1)}$ be defined as $\mathscr{Z} = \mathbb{Z}^{n\times(n+1)} \times C \times \mathbb{Z}$. Then $g(w).\mathbf{s} \in C$ iff $h(w) \in \mathscr{Z}$, proving the lemma.

3 Finitely typed monoids characterizations

In this section we characterize DetCA, UnCA, DetAPA, and UnAPA using the theory of (finitely) typed monoids [KLR07].

▶ **Definition 9** (Typed monoid [KLR07]). A *typed monoid* is a pair $(S, \mathfrak{S})$ where S is a finitely generated monoid and $\mathfrak{S}$ is a finite Boolean algebra of subsets of S whose elements are called *types*. We write $(S, \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\})$ for the typed monoid $(S, \mathfrak{S})$ where $\mathfrak{S}$ is generated by the $\mathcal{S}_i$'s. If $n = 1$, we simply write $(S, \mathcal{S}_1)$. For two typed monoids $(M, \mathfrak{M})$, $(N, \mathfrak{N})$, their Cartesian product $(S, \mathfrak{S}) = (M, \mathfrak{M}) \times (N, \mathfrak{N})$ is defined by $S = M \times N$, and $\mathfrak{S}$ is the Boolean algebra generated by $\{\mathcal{M} \times \mathcal{N} \mid \mathcal{M} \in \mathfrak{M} \wedge \mathcal{N} \in \mathfrak{N}\}$.

We view a finite monoid M as the typed monoid $(M, 2^M)$, and write $\mathbf{M}$ for the class of typed finite monoids.

▶ **Definition 10** (Language recognition [KLR07]). A typed monoid $(S, \mathfrak{S})$ *recognizes* a language $L \subseteq \Sigma^*$ if there is a morphism $h \colon \Sigma^* \to S$ and a type $\mathcal{S} \in \mathfrak{S}$ such that $L = h^{-1}(\mathcal{S})$. We write $H^{-1}((S, \mathfrak{S}))$ for the class of languages, over any alphabet, recognized by $(S, \mathfrak{S})$.

The usual wreath product (resp. block product) of M and N, i.e., the unilateral (resp. bilateral) semidirect product of M^N (resp. $M^{N\times N}$) and N, results, in the infinite monoid case, in monoids with uncountably many elements, failing to fall within the definition of typed monoid. Thus these products are restricted to *type-respecting* functions, that is, functions that only depend on the type of their arguments (multiplied by some constants):

▶ **Definition 11** (Type-respecting [KLR07]). Let $(M, \mathfrak{M})$ be a typed monoid and S any set. A function $f : M \times M \to S$ is *type-respecting* if it has a finite image and for each $s \in S$, the set $\{(m_1, m_2) \mid f(m_1, m_2) = s\}$ can be described by a finite Boolean combination of conditions of the form $m_1.c_1 \in \mathcal{M}_1$, $c_2.m_2 \in \mathcal{M}_2$, $m_1.c_3.m_2 \in \mathcal{M}_3$, where $c_1, c_2, c_3 \in M$ are constants and $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3 \in \mathfrak{M}$ are types. A function $f : M \to S$ is *type-respecting* if there is a type-respecting function $f' : M \times M \to S$ such that for all $m \in M$, $f(m) = f'(m, e_M)$.

Remark. We will seldom use the previous technical definition. In most of our cases, the typed monoid $(M, \mathfrak{M})$ will be finite (that is, M is a finite monoid and $\mathfrak{M}$ is the powerset of M); in this case, *any* function in $S^{M\times M}$ (or S^M) is type respecting, as its value on (m_1, m_2) depends only on whether m_1 and m_2 belong to types of the form $\{m\}$, for $m \in M$.

▶ **Definition 12** (Block product [BKM07]). Let $(M, \mathfrak{M})$, $(N, \mathfrak{N})$ be typed monoids, and let V_2 be the set of all type-respecting functions from $N \times N$ to M. The *(finitely typed) block product* $(S, \mathfrak{S})$ of $(M, \mathfrak{M})$ and $(N, \mathfrak{N})$, written $(M, \mathfrak{M})\Box(N, \mathfrak{N})$, is such that S is the bilateral semidirect product $V_2 ** N$ with the actions of the untyped block product, and $\mathfrak{S} = \{\mathcal{S}_{\mathcal{M}} \mid \mathcal{M} \in \mathfrak{M}\}$ where:

$$\mathcal{S}_{\mathcal{M}} = \{(f, n) \in S \mid f(e_N, e_N) \in \mathcal{M}\} \ .$$

The primary focus of the theory of typed monoids was the algebraic characterization of classes of circuit complexity, which are naturally closed under reversal. We need a finer notion of product to characterize classes which are not closed under reversal, thus we naturally derive the following definition from the definitions of typed block product and untyped wreath product:

▶ **Definition 13** (Wreath product). Let $(M, \mathfrak{M})$, $(N, \mathfrak{N})$ be typed monoids, and let V_1 be the set of all type-respecting functions from N to M. The *(finitely typed) wreath product* $(S, \mathfrak{S})$ of $(M, \mathfrak{M})$ and $(N, \mathfrak{N})$, written $(M, \mathfrak{M}) \wr (N, \mathfrak{N})$, is such that S is the unilateral semidirect product $V_1 * N$ with the action of the untyped wreath product, and $\mathfrak{S} = \{\mathcal{S}_{\mathcal{M}} \mid \mathcal{M} \in \mathfrak{M}\}$ where:

$$\mathcal{S}_{\mathcal{M}} = \{(f, n) \in S \mid f(e_N) \in \mathcal{M}\} \ .$$

Remark. The definitions of the typed block and wreath products are only slight modifications of those in the classical untyped case (see, e.g., [Str94]). In particular, the fundamental property of the multiplication in both cases still holds: for typed monoids $(M, \mathfrak{M})$, $(N, \mathfrak{N})$, and $(f_i, n_i) \in (M, \mathfrak{M}) \wr (N, \mathfrak{N})$, $i = 1, \dots, k$:

$$\prod_{i=1}^{k}(f_i, n_i) = \left(\sum_{i=1}^{k}(n_1 \cdots n_{i-1}) \cdot f_i, n_1 n_2 \cdots n_k \right) ,$$

where the sum of two functions is their pointwise product over M. Similarly, if the (f_i, n_i)'s are in $(M, \mathfrak{M}) \Box (N, \mathfrak{N})$:

$$\prod_{i=1}^{k}(f_i, n_i) = \left(\sum_{i=1}^{k}(n_1 \cdots n_{i-1}) \cdot f_i \cdot (n_{i+1} \cdots n_k), n_1 n_2 \cdots n_k \right) .$$

Also, we define the wreath product (resp. block product) of two sets of typed monoids to be the set of typed monoids which are the wreath product (resp. block product) of a typed monoid in the first set with a typed monoid in the second set.

Remark. We will frequently focus on languages which do not contain the empty word. This is a technical simplification which introduces no loss of generality, as all our typed monoid classes recognize $\{\varepsilon\}$ and are closed under union.

Let $\mathbf{Z}^+$ be the set of typed monoids $\{(\mathbb{Z}, \mathbb{Z}^+)^k \mid k \geq 1\}$.

▶ **Theorem 90.** *The class of languages recognized by* $\mathbf{Z}^+ \wr \mathbf{M}$ *is* $\mathscr{L}_{\text{DetCA}}$. *In symbols,* $H^{-1}(\mathbf{Z}^+ \wr \mathbf{M}) = \mathscr{L}_{\text{DetCA}}$.

Proof. ($\mathscr{L}_{\text{DetCA}} \subseteq H^{-1}(\mathbf{Z}^+ \wr \mathbf{M})$) We first show that $H^{-1}(\mathbf{Z}^+ \wr \mathbf{M})$ is closed under union and intersection. Let $L_1, L_2 \in H^{-1}(\mathbf{Z}^+ \wr \mathbf{M})$ be two languages over Σ, that is, for $i = 1, 2$, there exist a finite monoid M_i, an integer k_i, a morphism $h_i \colon \Sigma^* \to \mathbb{Z}^{k_i} \wr M_i$, and a type $\mathscr{T}_i$ of $(\mathbb{Z}, \mathbb{Z}^+)^{k_i}$ such that $L_i = h_i^{-1}(\mathscr{T}_i)$.

Consider the typed monoid $(\mathbb{Z}, \mathbb{Z}^+)^{k_1+k_2} \wr (M_1 \times M_2) \in \mathbf{Z}^+ \wr \mathbf{M}$. This monoid recognizes both the intersection and union of L_1 and L_2 as follows. Define $h \colon \Sigma^* \to \mathbb{Z}^{k_1+k_2} \wr (M_1 \times M_2)$ by $h(a) = (f_a, (\Psi_2(h_1(a)), \Psi_2(h_2(a))))$ where $a \in \Sigma$ and $f_a((m_1, m_2)) = ([\Psi_1(h_1(a))](m_1), [\Psi_1(h_2(a))](m_2)) \in \mathbb{Z}^{k_1+k_2}$. This function is type-respecting. Now let $\oslash \in \{\cup, \cap\}$. We define $\mathscr{T}_\oslash = (\mathscr{T}_1 \times \mathbb{Z}^{k_2}) \oslash (\mathbb{Z}^{k_1} \times \mathscr{T}_2)$, and thus $h^{-1}(\mathscr{T}_\oslash) = L_1 \oslash L_2$.

Now let (A, C) be a DetCA with $A = (Q, \Sigma, \delta, q_0, F)$, and suppose (by Lemma 87) that $F = Q$ and that the constraint set is expressed by a positive Boolean combination of clauses of the form $\sum_{t\in\delta} \alpha_t x_t > 0$. Closure of $H^{-1}(\mathbf{Z}^+ \wr \mathbf{M})$ under $\cup$ and $\cap$ together with Lemma 88 imply that it is enough to argue the case in which C is defined by a single such clause.

Let M be the transition monoid of A, $\eta\colon \Sigma^* \to M$ the canonical morphism associated, and for $m \in M$, write q^m for the action of m on q (i.e., $q^{\eta(w)}$ is the state reached reading w from the state q in A). We now define $h\colon \Sigma^* \to \mathbb{Z} \wr M$ as follows. Let $\tau\colon M \times \Sigma \to \delta$ be defined by $\tau(m,a) = q_0^m \bullet\!\!-a\!\!\to q_0^{m\eta(a)}$. Then:

$$h(a) = (f_a, \eta(a)),\ \text{where}\ f_a(m) = \alpha_{\tau(m,a)}\ .$$

Now for $w = w_1 w_2 \cdots w_n \in \Sigma^*$ and $\pi = \pi_1\pi_2\cdots\pi_n$ the unique accepting path in A from q_0 labeled w, we have:

$$h(w) = (f_{w_1} + \eta(w_1)\cdot f_{w_2} + \cdots + \eta(w_1 w_2 \cdots w_{n-1})\cdot f_{w_n}, \eta(w))$$

$$[\Psi_1(h(w))](\eta(\varepsilon)) = \alpha_{\tau(\eta(\varepsilon),w_1)} + \sum_{i=2}^{n} \alpha_{\tau(\eta(w_1\cdots w_{i-1}),w_i)}\ ,$$

note that $q_0^{\eta(w_1\cdots w_{i-1})}$ is $\mathsf{From}(\pi_{i-1})$ and thus $\tau(\eta(w_1\cdots w_{i-1}), w_i) = \pi_i$, hence:

$$[\Psi_1(h(w))](\eta(\varepsilon)) = \sum_{i=1}^{n} \alpha_{\pi_i} = \sum_{t\in\delta} |\pi|_t \times \alpha_t\ .$$

Thus, $\mathsf{Pkh}(\pi) \in C$ iff $[\Psi_1(h(w))](\eta(\varepsilon)) > 0$. Hence with the type $\mathcal{T} = \{(f,m) \in (\mathbb{Z},\mathbb{Z}^+) \wr M \mid f(\eta(\varepsilon)) > 0\}$, which is indeed a type of $(\mathbb{Z},\mathbb{Z}^+) \wr M$, we have that $h^{-1}(\mathcal{T}) = L(A,C)$.

($H^{-1}(\mathbf{Z}^+ \wr \mathbf{M}) \subseteq \mathcal{L}_{\text{DetCA}}$) Let $L \subseteq \Sigma^*$ be recognized by $(\mathbb{Z},\mathbb{Z}^+)^k \wr M$ using a type $\mathcal{T}$ and a morphism $h\colon \Sigma^* \to (\mathbb{Z}^k)^M \times M$, and write for convenience $h_i(w) = \Psi_i(h(w))$, $i = 1,2$. Let A be the automaton $(M, \Sigma, \delta, e_M, M)$, where:

$$\delta = \{m \bullet\!\!-a\!\!\to m' \mid m \in M \wedge a \in \Sigma \wedge m' = m.h_2(a)\}\ .$$

Now as $\mathcal{L}_{\text{DetCA}}$ is closed under union and intersection, we may suppose that the type $\mathcal{T}$ is of the following form:

$$\mathcal{T} = \prod_{i=1}^{k} \{(f,m) \mid f(e_M) \in \mathcal{T}_i\}\ ,$$

where each $\mathcal{T}_i \in \{\emptyset, \mathbb{Z}_0^-, \mathbb{Z}^+, \mathbb{Z}\}$. Define $T = \mathcal{T}_1 \times \mathcal{T}_2 \times \cdots \times \mathcal{T}_k$, and the semilinear set C consisting of elements:

$$(x_{t_1}, x_{t_2}, \ldots, x_{t_{|\delta|}})\ \text{s.t.}\ \sum_{t\in\delta} x_t \times [h_1(\mu(t))](\mathsf{From}(t)) \in T\ .$$

We claim that the language of the DetCA (A,C) is L. Let $w = w_1 w_2 \cdots w_n \in \Sigma^*$. There is an (accepting) path in A labeled w going through the states $e_M = h_2(\varepsilon)$,

$h_2(w_1)$, $h_2(w_1w_2)$, …, $h_2(w_1w_2\cdots w_n)$. Thus the sum computed by the semilinear set is $h_1(w_1) + h_2(w_1)\cdot h_1(w_2) + \cdots + h_2(w_1w_2\cdots w_n)\cdot h_1(w_n)$, taken at the point e_M. This is precisely $[h_1(w)](e_M)$, and thus checking whether it belongs to T is equivalent to checking whether $h(w) \in \mathscr{T}$. Hence $L = L(A, C)$.

Now $\mathscr{L}_{\mathrm{DetCA}}$ is a *variety of languages*, in the sense of $*$-varieties of [Eil76], i.e., it is closed under the Boolean operations, inverse morphisms, and quotient by a word. The Eilenberg-type theorem of typed monoids [BKR11, Thm. 2] states that varieties of languages and varieties of typed monoids (see [BKR11, Sec. 6]) are in one-to-one correspondence. This implies that the closure of $\mathbf{Z}^+ \wr \mathbf{M}$ under direct product, typed submonoid (see [BKR11, Def. 3]), and inverse image by a typed morphism (see [BKR11, Def. 2]) is *the* variety of typed monoids corresponding to $\mathscr{L}_{\mathrm{DetCA}}$. We may naturally ask whether this class is closed under wreath product, and we show this is not the case. Let $U_1 = (\{0,1\}, \times)$, then:

Theorem 91. *There is a language $L \notin \mathscr{L}_{\mathrm{CA}}$ recognized by $U_1 \wr (\mathbb{Z}, \mathbb{Z}^+)$ and by $(\mathbb{Z}, \mathbb{Z}^+) \wr (\mathbb{Z}, \mathbb{Z}^+)$.*

Proof. We treat the case $(\mathbb{Z}, \mathbb{Z}^+) \wr (\mathbb{Z}, \mathbb{Z}^+)$, this is similar for $U_1 \wr (\mathbb{Z}, \mathbb{Z}^+)$. We consider the language:

$$P_1 = \{w = w_1w_2\cdots w_k \in \{\sqsubset, \sqsupset\}^* \mid (\forall i)[|w_1w_2\cdots w_i|_\sqsubset \geq |w_1w_2\cdots w_i|_\sqsupset]\} \ ,$$

which is not in $\mathscr{L}_{\mathrm{CA}}$ (Paper III, Prop. 64). Define $h\colon \{\sqsubset, \sqsupset\}^* \to \mathbb{Z} \wr \mathbb{Z}$ by:

$$h(\sqsubset) = (f, -1)$$
$$h(\sqsupset) = (f, 1)$$
$$f(n) = \begin{cases} 0 & \text{if } n \leq 0 \ , \\ 1 & \text{otherwise.} \end{cases}$$

Note that f is indeed type-respecting, as its value depends only on whether its argument is in the type $\mathbb{Z}^+$ of the typed monoid $(\mathbb{Z}, \mathbb{Z}^+)$. Now $h(w) = (f, |w|_\sqsupset - |w|_\sqsubset)$ and $f(0)$ is zero or less iff for all i it holds that $|w_1w_2\cdots w_i|_\sqsubset \geq |w_1w_2\cdots w_i|_\sqsupset$. Thus:

$$P_1 = h^{-1}(\{(f, n) \mid f(0) \in \mathbb{Z}_0^-\}) \ ,$$

and hence $P_1 \in H^{-1}((\mathbb{Z}, \mathbb{Z}^+) \wr (\mathbb{Z}, \mathbb{Z}^+))$.

▶ **Theorem 92.** *The class of languages recognized by* $\mathbf{Z}^+\Box\mathbf{M}$ *is* $\mathscr{L}_{\text{UnCA}}$. *In symbols,* $H^{-1}(\mathbf{Z}^+\Box\mathbf{M}) = \mathscr{L}_{\text{UnCA}}$.

Proof. ($\mathscr{L}_{\text{UnCA}} \subseteq H^{-1}(\mathbf{Z}^+\Box\mathbf{M})$) We first note that $H^{-1}(\mathbf{Z}^+\Box\mathbf{M})$ is closed under union and intersection; this is the same proof as in Theorem 90 except that f_a is now defined as:

$$f_a((m_1, m_2), (m_1', m_2')) = ([\Psi_1(h_1(a))](m_1, m_1'), [\Psi_1(h_2(a))](m_2, m_2')) \ .$$

Next consider an UnCA (A, C) with $A = (Q, \Sigma, \delta, q_0, F)$, and suppose (Lemma 87) that $L(A) = \Sigma^*$ and that the constraint set is expressed by a positive Boolean combination of clauses of the form $\sum_{t\in\delta} \alpha_t x_t > 0$. Closure of $H^{-1}(\mathbf{Z}^+\Box\mathbf{M})$ under $\cup$ and $\cap$ together with Lemma 88 imply that it is enough to argue the case in which C is defined by a single such clause.

Let M be the transition monoid of the deterministic version of A, obtained using the powerset construction. Let A' be defined as A with all transitions inverted (i.e., $p \bullet\!\!-a\!\!\to q$ is in A iff $q \bullet\!\!-a\!\!\to p$ is in A'). Let M' be the transition monoid of the deterministic version of A', using again the powerset construction, and let M^{c} be the monoid defined on the same elements as M' but with the operation reversed (i.e., $m_1 \circ_{M'} m_2$ in M' is $m_2 \circ_{M^{\text{c}}} m_1$ in M^{c}; this is still a monoid as $\circ_{M^{\text{c}}}$ is still associative). We will show that $L(A, C)$ is recognized by $(S, \mathfrak{S}) = (\mathbb{Z}, \mathbb{Z}^+)\Box(M \times M^{\text{c}})$.

Write η and η^{c} for the canonical morphisms associated with M and M^{c}; for $m \in M$ and $R \subseteq Q$, write R^m for the action of m on R, and likewise for M^{c}. We first note that for $w \in \Sigma^*$, $\{q_0\}^{\eta(w)}$ is the set of states of A that can be reached in A reading w from q_0, and, likewise, that $F^{\eta^{\text{c}}(w)}$ is the set of states in A from which reading w leads to a final state.

Now for $m_1 \in M$, $a \in \Sigma$, and $m_2 \in M^{\text{c}}$, let $\tau(m_1, a, m_2)$ be the unique transition in A from a state in $\{q_0\}^{m_1}$ to a state in F^{m_2} labeled a. We show that τ is well-defined. Let w_1, w_2 such that $\eta(w_1) = m_1$ and $\eta^{\text{c}}(w_2) = m_2$; this means that there are w_1-labeled paths in A from q_0 to *any* state in $\{q_0\}^{m_1}$, and, likewise, w_2-labeled paths in A from *any* state in F^{m_2} to a final state. *(Existence):* as $w_1 a w_2$ is in $\Sigma^* = L(A)$, there is a transition in A from a state in $\{q_0\}^{m_1}$ to a state in F^{m_2} labeled a. *(Uniqueness):* if two transitions $p \bullet\!\!-a\!\!\to p'$ and $q \bullet\!\!-a\!\!\to q'$ are such that $p, q \in \{q_0\}^{m_1}$ and $p', q' \in F^{m_2}$, this means that there are multiple accepting paths in A labeled $w_1 a w_2$, contradicting the unambiguity of A.

We now define $h\colon \Sigma^* \to S$ by:

$$\begin{aligned} h(a) &= (f_a, (\eta(a), \eta^{\mathrm{c}}(a))), \text{ where} \\ f_a((m_1, m_2), (m'_1, m'_2)) &= \alpha_{\tau(m_1, a, m'_2)} \ . \end{aligned}$$

Now let $w = w_1 w_2 \cdots w_i \in \Sigma^*$ and π be the unique path in A from q_0 to a final state labeled w. Then:

$$\begin{aligned} \pi &= \pi_1 \pi_2 \cdots \pi_n \quad \text{where} \\ \pi_i &= \tau(\eta(w_1 w_2 \cdots w_{i-1}), w_i, \eta^{\mathrm{c}}(w_{i+1} w_{i+2} \cdots w_n)) \ , \end{aligned}$$

and thus:

$$[\Psi_1(h(w))]((\eta(\varepsilon), \eta^{\mathrm{c}}(\varepsilon))) = \sum_{t \in \delta} |\pi|_t \times \alpha_t \ .$$

Thus $\mathsf{Pkh}(\pi) \in C$ iff $[\Psi_1(h(w))]((\eta(\varepsilon), \eta^{\mathrm{c}}(\varepsilon))) > 0$. Hence with the type $\mathscr{S} = \{(f, m) \in S \mid f((\eta(\varepsilon), \eta^{\mathrm{c}}(\varepsilon))) \in \mathbb{Z}^+\}$, which is indeed a type in $\mathfrak{S}$ as $\mathbb{Z}^+$ is a type of $(\mathbb{Z}, \mathbb{Z}^+)$, we have that $h^{-1}(\mathscr{S}) = L(A, C)$.

($\boldsymbol{H^{-1}}(\mathbf{Z^+ \Box M}) \subseteq \mathscr{L}_{\text{UnCA}}$) Let $L \subseteq \Sigma^*$ be recognized by $(\mathbb{Z}, \mathbb{Z}^+)^k \Box M$ using a type $\mathscr{T}$ and a morphism $h = (h_1, h_2)$ with $h_1\colon \Sigma^* \to \mathbb{Z}^{M \times M}$ and $h_2\colon \Sigma^* \to M$. Let $A(s_1, s_2)$ be the automaton $(M \times M, \Sigma, \delta, (s_1, s_2), M \times \{e_M\})$ where:

$$\begin{aligned} \delta = &\{(m_1, m_2) \bullet\!\!-a\!\!\to (m'_1, m'_2) \mid \\ &\quad m'_1 = m_1.h_2(a) \wedge h_2(a).m'_2 = m_2 \in M \wedge a \in \Sigma\} \ . \end{aligned}$$

Note that $w \in L(A(s_1, s_2))$ implies $h_2(w) = s_2$. We argue that $A(s_1, s_2)$ is unambiguous for any $s_1, s_2 \in M$. It is clear that if $w = \varepsilon$, every $A(s_1, s_2)$ has at most one accepting path labeled w. Now let $w = a \cdot v$ for $v \in \Sigma^*$. Suppose $w \in L(A(s_1, s_2))$. This implies that $h_2(w) = s_2$. The states that can be reached from $(s_1, h_2(w))$ reading a are all of the form $(s_1.h_2(a), m)$, $m \in M$. Now v should be accepted by the automaton A where the initial state is set to one of these states; thus there is only one state fitting, $(s_1.h_2(a), h_2(v))$. By induction hypothesis, there is only one path in $A(s_1.h_2(a), h_2(v))$ recognizing v, thus there is only one path in $A(s_1, h_2(w))$ recognizing w. This shows that for any s_1, s_2, $A(s_1, s_2)$ is unambiguous.

Now, with $e = (e_M, e_M)$, and as $\mathscr{L}_{\text{UnCA}}$ is closed under union and intersection, we may suppose that the type $\mathscr{T}$ is of the following form:

$$\mathscr{T} = \prod_{i=1}^{k} \{(f, m) \mid f(e, e) \in \mathscr{T}_i\} \ ,$$

where each $\mathcal{T}_i \in \{\emptyset, \mathbb{Z}_0^-, \mathbb{Z}^+, \mathbb{Z}\}$. Define $T = \mathcal{T}_1 \times \mathcal{T}_2 \times \cdots \times \mathcal{T}_k$, and the semilinear set C consisting of elements:

$$(x_{t_1}, x_{t_2}, \ldots, x_{t_{|\delta|}}) \text{ s.t. } \sum_{t \in \delta} x_t \times [h_1(\mu(t))](\Psi_1(\mathsf{From}(t)), \Psi_2(\mathsf{To}(t))) \in T \enspace .$$

We show that $\bigcup_{m \in M} L(A(e_M, m), C)$ is L. Let $w = w_1 w_2 \cdots w_n \in \Sigma^*$. There is a unique accepting path in $A(e_M, h_2(w))$ (and in no other $A(e_M, m)$) labeled w, and it is going successively through the states $(h_2(\varepsilon), h_2(w)) = (e_M, h_2(w))$, $(h_2(w_1), h_2(w_2 \cdots w_n))$, …, $(h_2(w), e_M) = (h_2(w), h_2(\varepsilon))$. For this path, the sum computed by the semilinear set is:

$$\sum_{i=1}^{n} h_2(w_1 \cdots w_{i-1}) \cdot h_1(w_i) \cdot h_2(w_{i+1} \cdots w_n) \enspace ,$$

at the point (e_M, e_M). This is precisely $[h_1(w)](e_M, e_M)$, and checking whether it is in T amounts to checking whether $h(w) \in \mathcal{T}$, hence $L = \bigcup_{m \in M} L(A(e_M, m), C)$.

We derive an interesting property of the logical characterization and circuit complexity of UnCA. Let MSO[<] be the monadic second-order logic with < as the unique numerical predicate, and FO+G[<] be the first-order logic with group quantifiers and < as the unique numerical predicate. Both logics express exactly the regular languages (these are respectively the classical results of Büchi [Büc60] and Barrington, Immerman, Straubing [BIS90]). Now define the *extended majority* quantifier $\widehat{\text{Maj}}$, introduced in [BKM07], as: $w \models \widehat{\text{Maj}}\ x\ \langle \phi_i \rangle_{i=1,\ldots,m}$ iff $\sum_{x=1}^{|w|} |\{i \mid w \models \phi_i(x)\}| - |\{i \mid w \not\models \phi_i(x)\}| > 0$. Then:

▶ **Corollary 93.** *A language is in $\mathcal{L}_{\text{UnCA}}$ iff it can be expressed as a Boolean combination of formulas of the form:*

$$\widehat{\text{Maj}}\ x\ \langle \phi_i \rangle_{i=1,\ldots,m}$$

where each ϕ_i is an MSO[<] *formula or an* FO+G[<] *formula. Hence, $\mathcal{L}_{\text{UnCA}} \subsetneq \text{NC}^1$.*

Proof. We first show that the languages recognized by $(\mathbb{Z}, \mathbb{Z}^+)$ are those expressible as a formula of the form (or negation of) $\widehat{\text{Maj}}\ x\ \langle Q_{A_i} x \rangle_{i=1,\ldots,m}$ where $A_i \subseteq \Sigma$, and $Q_{A_i} x$ is short for $\bigvee_{a \in A_i} Q_a x$.

Let $L \in H^{-1}((\mathbb{Z}, \mathbb{Z}^+))$, i.e., let $h\colon \Sigma^* \to \mathbb{Z}$ be a morphism and suppose $L = h^{-1}(\mathbb{Z}^+)$ (if $L = h^{-1}(\mathbb{Z}_0^-)$, then the negation of the formula we obtain here will describe L). We suppose moreover, w.l.o.g., that each $h(a)$, $a \in \Sigma$, is even. Now let m be $\max\{|h(a)| \mid a \in \Sigma\}$ and define, for $1 \leq i \leq m$:

$$A_i = \{a \in \Sigma \mid m + h(a) \geq 2 \times i\} \ .$$

Now let $w \in \Sigma^*$ be a word and $1 \leq x \leq |w|$. Then it holds that:

$$h(w_x) = \underbrace{|\{i \mid w_x \in A_i\}|}_{(m+h(a))/2} - \underbrace{|\{i \mid w_x \notin A_i\}|}_{m-(m+h(a))/2} \ .$$

Thus for $w \in \Sigma^*$, $h(w) > 0$ iff $w \models \widehat{\mathrm{Maj}}\ x\ \langle Q_{A_i} x\rangle_{i=1,\dots,m}$, thus the language expressed by this latter formula is $h^{-1}(\mathbb{Z}^+) = L$.

Conversely, consider a formula $\widehat{\mathrm{Maj}}\ x\ \langle Q_{A_i} x\rangle_{i=1,\dots,m}$. Then let $h\colon \Sigma^* \to \mathbb{Z}$ be the morphism defined by $h(a) = |\{i \mid a \in A_i\}| - |\{i \mid a \notin A_i\}|$, for $a \in \Sigma$. We have that for $w \in \Sigma^*$, $h(w) > 0$ iff the formula under consideration holds true, implying that the language recognized by the formula is $h^{-1}(\mathbb{Z}^+)$.

It follows that the languages recognized by $\mathbf{Z}^+$ are the Boolean combinations of languages expressible as such formulas. Now the languages (with one free variable) recognized by finite monoids are those recognized by MSO[<] or FO+G[<] formulas. Thus the *block product principle* [Kre08, Theorem 3.40] implies that the languages of $\mathscr{L}_{\text{UnCA}} = H^{-1}(\mathbf{Z}^+\Box\mathbf{M})$ are those expressible as Boolean combinations of formulas of the form of the statement of the lemma. As for circuits, the regular languages (with one free variable, in the sense of $\mathscr{V}$-structures [Str94, p. 14]) are recognized by NC^1 circuits, and a formula or negation of a formula of the form $\widehat{\mathrm{Maj}}\ x\ \langle Q_{A_i} x\rangle_{i=1,\dots,m}$ can be expressed by a threshold circuit. Now [Kre08, Lemma 4.29] implies that $\mathscr{L}_{\text{UnCA}} \subseteq \mathrm{NC}^1$. Strictness is implied by Theorem 91, as the Dyck languages are in NC^1.

Remark. The characterization of $\mathscr{L}_{\text{UnCA}}$ by means of block products allows for an alternative algebra-based proof that $\mathscr{L}_{\text{UnCA}}$ is closed under reversal (the proof of Paper III, Proposition 60, is automata-based). Let $L \in \mathscr{L}_{\text{UnCA}}$, there is a finite monoid M, an integer k, a type $\mathscr{T}$ of $(\mathbb{Z}, \mathbb{Z}^+)^k\Box M$, and a morphism $h\colon \Sigma^* \to \mathbb{Z}^k\Box M$ such that $L = h^{-1}(\mathscr{T})$. Define M^{c} as M where the operation is reversed, and $h^{\mathrm{c}}\colon \Sigma^* \to \mathbb{Z}^k\Box M^{\mathrm{c}}$ by:

$$h^{\mathrm{c}}(a) = (f_a^{\mathrm{c}}, \Psi_2(h(a))) \text{ where}$$
$$f_a^{\mathrm{c}}(m_1, m_2) = [\Psi_1(h(a))](m_2, m_1) \ .$$

Then $L^{\mathrm{R}} = (h^{\mathrm{c}})^{-1}(\mathscr{T})$ and, as M^{c} is also a finite monoid, $L^{\mathrm{R}} \in H^{-1}(\mathbf{Z}^{+}\Box\mathbf{M}) = \mathscr{L}_{\mathrm{UnCA}}$.

Write $\mathfrak{Z}^{+}(k)$ for the type set of $(\mathbb{Z}, \mathbb{Z}^{+})^{k}$, that is, the sets expressible as a Boolean combination of expressions of the form $x_i > 0$. Let $\mathbf{ZMat}^{+}$ be the set of typed monoids $\{(\mathscr{M}_{\mathbb{Z}}(k), \mathfrak{Z}^{+}(k \times k)) \mid k \geq 1\}$, then:

▶ **Theorem 94.** *The class of languages recognized by* $\mathbf{ZMat}^{+}$ *is* $\mathscr{L}_{\mathrm{DetAPA}}$. *In symbols,* $H^{-1}(\mathbf{ZMat}^{+}) = \mathscr{L}_{\mathrm{DetAPA}}$.

Proof. ($\mathscr{L}_{\mathrm{DetAPA}} \subseteq H^{-1}(\mathbf{ZMat}^{+})$) This is a direct consequence of Lemma 89.

($H^{-1}(\mathbf{ZMat}^{+}) \subseteq \mathscr{L}_{\mathrm{DetAPA}}$) Given $k \geq 1$, a type $\mathscr{Z}$ of $(\mathbb{Z}, \mathbb{Z}^{+})^{k\times k}$, and a morphism $h\colon \Sigma^{*} \to \mathscr{M}_{\mathbb{Z}}(k)$, we build a two-state DetAPA of dimension k^2 for $h^{-1}(\mathscr{Z})$. First, let $h'\colon \Sigma^{*} \to \mathscr{M}_{\mathbb{Z}}(k^2)$ be such that $h'(a)$ is the Kronecker product of the identity matrix of dimension k and $h(a)$. Define $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k)$ where each $\mathbf{e}_i$ is of dimension k. Then for any word w, $h(w) \in \mathscr{Z}$ iff $h'(w).\mathbf{e} \in \mathscr{Z}$. Now let $A = (\{r, s\}, \Sigma, \delta, r, \{s\})$, with $\delta = \{r, s\} \times \Sigma \times \{s\}$. Then let $U\colon \delta^{*} \to \mathscr{F}_{k^2}$ for $q \in \{r, s\}$, $a \in \Sigma$, and $\mathbf{x} \in \mathbb{Z}^{k^2}$ be defined by:

$$U_{q \xleftarrow{} a \to s}(\mathbf{x}) = \begin{cases} h'(a).\mathbf{e} & \text{if } q = r, \\ h'(a).\mathbf{x} & \text{otherwise.} \end{cases}$$

This implies that for $w \in \Sigma^{+}$ and π its unique accepting path in A, it holds that $U_{\pi}(\mathbf{0}) = h'(w).\mathbf{e}$. Thus $L(A, U, \mathscr{Z}) = h^{-1}(\mathscr{Z})$.

▶ **Theorem 95.** *The class of languages recognized by* $\mathbf{ZMat}^{+}\Box\mathbf{M}$ *is* $\mathscr{L}_{\mathrm{UnAPA}}$. *In symbols,* $H^{-1}(\mathbf{ZMat}^{+}\Box\mathbf{M}) = \mathscr{L}_{\mathrm{UnAPA}}$.

Proof. $\mathscr{L}_{\mathrm{UnAPA}} \subseteq H^{-1}(\mathbf{ZMat}^{+}\Box\mathbf{M})$ is the same as $\mathscr{L}_{\mathrm{UnCA}} \subseteq H^{-1}(\mathbf{Z}^{+}\Box\mathbf{M})$ in Theorem 92, thanks to Lemma 89.

$H^{-1}(\mathbf{ZMat}^{+}\Box\mathbf{M}) \subseteq \mathscr{L}_{\mathrm{UnAPA}}$ is the same as $H^{-1}(\mathbf{Z}^{+}\Box\mathbf{M}) \subseteq \mathscr{L}_{\mathrm{UnCA}}$ in Theorem 92 for the automaton part, and the same as Theorem 94 for the constraint set and affine function parts.

In both cases, those characterizations make the following new closure properties easy to show:

▶ **Corollary 96.** *Both* $\mathscr{L}_{\mathrm{DetAPA}}$ *and* $\mathscr{L}_{\mathrm{UnAPA}}$ *are closed under reversal.*

Proof. For a matrix M, write M^{T} for the transpose of M. We extend this notation to types in $\mathfrak{Z}^{+}(k)$ naturally. Let $L \in H^{-1}(\mathbf{ZMat}^{+})$, there are $h: \Sigma^{*} \to \mathscr{M}_{\mathbb{Z}}(k)$ and $\mathscr{Z} \in \mathfrak{Z}^{+}(k \times k)$ such that $L = h^{-1}(\mathscr{Z})$. Define $h': \Sigma^{*} \to \mathscr{M}_{\mathbb{Z}}(k)$ by $h'(a) = (h(a))^{\mathrm{T}}$. Then for a word w, $h(w) = (h'(w^{\mathrm{R}}))^{\mathrm{T}}$, and thus $h'(w) \in \mathscr{Z}^{\mathrm{T}}$ iff $h(w^{\mathrm{R}}) \in \mathscr{Z}$. Hence $L^{\mathrm{R}} = (h')^{-1}(\mathscr{Z}^{\mathrm{T}}) \in H^{-1}(\mathbf{ZMat}^{+})$.

In the UnAPA case, we apply the same argument as the remark on page 141, with the transposes as in the previous case. Let $L \in H^{-1}(\mathbf{ZMat}^{+}\Box\mathbf{M})$, there is a finite monoid M, a type $\mathscr{T}$ of $(\mathscr{M}_{\mathbb{Z}}(k), \mathfrak{Z}^{+}(k))\Box M$, and a morphism $h: \Sigma^{*} \to \mathscr{M}_{\mathbb{Z}}(k)\Box M$ such that $L = h^{-1}(\mathscr{T})$. Now $\mathscr{T}$ is of the form $\{(f, m) \mid f(e_M, e_M) \in \mathscr{Z}\}$ for some $\mathscr{Z} \in \mathfrak{Z}^{+}(k)$. Define $\mathscr{T}^{\mathrm{T}}$ as the type $\{(f, m) \mid f(e_M, e_M) \in \mathscr{Z}^{\mathrm{T}}\}$. Now define M^{c} as M where the operation is reversed, and $h^{\mathrm{c}}: \Sigma^{*} \to \mathscr{M}_{\mathbb{Z}}(k)\Box M^{\mathrm{c}}$ by:

$$h^{\mathrm{c}}(a) = (f_a^{\mathrm{c}}, \Psi_2(h(a))) \text{ where}$$
$$f_a^{\mathrm{c}}(m_1, m_2) = ([\Psi_1(h(a))](m_2, m_1))^{\mathrm{T}} \ .$$

Then $L^{\mathrm{R}} = (h^{\mathrm{c}})^{-1}(\mathscr{T}^{\mathrm{T}}) \in H^{-1}(\mathbf{ZMat}^{+}\Box\mathbf{M})$.

4 Formal power series characterization

In this section, we show that the languages of DetAPA are those expressible as a Boolean combination of positive supports of $\mathbb{Z}$-valued rational series. This helps us derive a separation over the unary languages between $\mathscr{L}_{\mathrm{CA}}$ and $\mathscr{L}_{\mathrm{DetAPA}}$ — the separation was known (Paper I, Proposition 25), but not over unary languages.

We start with the definition of series and of linear representation:

▶ **Definition 14** (e.g., [BR10]). We call functions from Σ^{*} into $\mathbb{Z}$ ($\mathbb{Z}$)-*series*. For such a series r, it is customary to write (r, w) for $r(w)$. We write $\mathsf{supp}_{+}(r)$ for the set $\{w \mid (r, w) > 0\}$.

A *linear representation* of dimension $k \geq 1$ is a triple $(\mathbf{s}, h, \mathbf{g})$ such that $\mathbf{s} \in \mathbb{Z}^{k}$ is a *row* vector, $\mathbf{g} \in \mathbb{Z}^{k}$ is a *column* vector, and $h: \Sigma^{*} \to (\mathbb{Z}^{k \times k}, .)$ is a monoid morphism, where . is the *usual* matrix multiplication. It defines the series $r = ||(\mathbf{s}, h, \mathbf{g})||$ with $(r, w) = \mathbf{s}.h(w).\mathbf{g}$.

A series is said to be *rational* if it is defined by a linear representation. We write $\mathbb{Z}^{\mathrm{rat}}\langle\langle\Sigma^{*}\rangle\rangle$ for the set of rational series.

For a class $\mathscr{C}$ of languages, write $\mathsf{BC}(\mathscr{C})$ for the Boolean closure of $\mathscr{C}$. Then:

▶ **Theorem 97.** $\mathscr{L}_{\mathrm{DetAPA}} = \mathsf{BC}(\mathsf{supp}_{+}(\mathbb{Z}^{\mathrm{rat}}\langle\langle\Sigma^{*}\rangle\rangle))$.

Proof. ($\mathscr{L}_{\text{DetAPA}} \subseteq \mathsf{BC}(\mathsf{supp}_+(\mathbb{Z}^{\text{rat}}\langle\langle\Sigma^*\rangle\rangle))$) First note that there is a rational series r such that $\mathsf{supp}_+(r) = \{\varepsilon\}$. Let L be in $\mathscr{L}_{\text{DetAPA}}$; we may thus suppose that $\varepsilon \notin L$. By the same token as in the proof of Theorem 94, there is a morphism $h\colon \Sigma^* \to \mathscr{M}_{\mathbb{Z}}(k)$, for some k, a vector $\mathbf{v} \in \{0,1\}^k$, and a type $\mathscr{Z}$ of $(\mathbb{Z}, \mathbb{Z}^+)^k$ such that:

$$L = \{w \mid h(w).\mathbf{v} \in \mathscr{Z}\} \ .$$

Further, similar to Lemma 88, $L(A, U, C_1 \oslash C_2) = L(A, U, C_1) \oslash L(A, U, C_2)$, for $\oslash \in \{\cup, \cap\}$ and any DetAPA $(A, U, C_1 \oslash C_2)$. Moreover, $L(A, U, \overline{C}) = \overline{L(A, U, C)} \cap L(A)$. We may thus suppose that $\mathscr{Z}$ is reduced to $\mathbb{Z}^{i-1} \times \mathbb{Z}^+ \times \mathbb{Z}^{k-i}$ for some i.

Now let h' be the morphism from Σ^* to $(\mathbb{Z}^{k\times k}, .)$, with . the *usual* matrix multiplication, where $h'(a) = (h(a))^{\mathrm{T}}$, with $a \in \Sigma$ and M^{T} the transpose of M. Note that $h(a_1a_2) = h(a_2).h(a_1) = ((h(a_1))^{\mathrm{T}}.(h(a_2))^{\mathrm{T}})^{\mathrm{T}}$, which is $(h'(a_1a_2))^{\mathrm{T}}$; more generally, $h(w) = (h'(w))^{\mathrm{T}}$. Thus we have that $\mathbf{v}^{\mathrm{T}}.h'(w) = (h(w).\mathbf{v})^{\mathrm{T}}$. Hence with $\mathbf{s} = \mathbf{v}^{\mathrm{T}}$ and $\mathbf{g}$ the column vector $\mathbf{e}_i$, $\mathbf{s}.h'(w).\mathbf{g} > 0$ iff $h(w).\mathbf{v} \in \mathscr{Z}$.

Now the triple $(\mathbf{s}, h', \mathbf{g})$ is a linear representation of a rational series which associates w to $\mathbf{s}.h'(w).\mathbf{g}$, and this concludes the proof.

($\mathsf{BC}(\mathsf{supp}_+(\mathbb{Z}^{\text{rat}}\langle\langle\Sigma^*\rangle\rangle)) \subseteq \mathscr{L}_{\text{DetAPA}}$) As $\mathscr{L}_{\text{DetAPA}}$ is closed under union, complement, and intersection, we need only show that $\mathsf{supp}_+(\mathbb{Z}^{\text{rat}}\langle\langle\Sigma^*\rangle\rangle) \subseteq \mathscr{L}_{\text{DetAPA}}$.

Let $(\mathbf{s}, h, \mathbf{g})$ be a linear representation of dimension k of a rational series r over the alphabet Σ. Define $h'\colon \Sigma^* \to \mathscr{M}_{\mathbb{Z}}(k)$ by letting $h'(a) = (h(a))^{\mathrm{T}}$, for $a \in \Sigma$. Then for $w \in \Sigma^*$, $h(w) = (h'(w))^{\mathrm{T}}$. Now the rest of the proof is similar to that of Theorem 94: define $A = (\{r,t\}, \Sigma, \delta, r, \{r,t\})$, with $\delta = \{r,t\} \times \Sigma \times \{t\}$. Then let $U\colon \delta^* \to \mathscr{F}_k$ for $q \in \{r,t\}$, $a \in \Sigma$, and $\mathbf{x} \in \mathbb{Z}^k$, be defined by:

$$U_{q \leftarrow a \rightarrow t}(\mathbf{x}) = \begin{cases} h'(a).\mathbf{s} & \text{if } q = r, \\ h'(a).\mathbf{x} & \text{otherwise.} \end{cases}$$

This implies that for $w \in \Sigma^*$ and π its unique accepting path in A, it holds that $U_\pi(\mathbf{0}) = \mathbf{s}.h(w)$. Thus letting $C = \{\mathbf{x} \mid \mathbf{x}.\mathbf{g} > 0\}$, with $\mathbf{x}$ a row vector and $\mathbf{g}$ a column vector, we have that $L(A, U, C) = \mathsf{supp}_+(r)$.

▶ **Corollary 98.** *There is an unary language in $\mathscr{L}_{\text{DetAPA}}$ which is not in $\mathscr{L}_{\text{CA}}$.*

Proof. For $\Sigma = \{a\}$, and a series r over Σ, write $c_n = (r, a^n)$. Recall that r is rational iff the sequence of c_n's satisfies a linear recurrence relation [BR10]. Now let r be specified by the linear recurrence relation and initial values:

$$c_n = 2 \times c_{n-1} - 5 \times c_{n-2}, \quad c_0 = 0, c_1 = 1 \ .$$

Then $c_n = \frac{1}{4}i((1-2i)^n - (1+2i)^n) = \frac{1}{2}5^{n/2}\sin(n\tan^{-1}(2))$. Write $x = \tan^{-1}(2)$, and note that x/π is not a rational number: this is a consequence of Niven's theorem [Niv56, Cor. 3.12], asserting that if x/π is rational, then the only rational values of $\tan(x)$ are 0 and ± 1. Now the sign of c_n only depends on the sign of $\sin(nx)$. Suppose $\mathsf{supp}_+(r)$ is in $\mathscr{L}_{\text{CA}}$. As any language in $\mathscr{L}_{\text{CA}}$ is semilinear [KR03] (implying that all the unary languages of $\mathscr{L}_{\text{CA}}$ are regular), there is a set $E = \{k_1 + t.k_2 \mid t \in \mathbb{N}\}$, with $k_1 \in \mathbb{N}$, $k_2 \in \mathbb{N}^+$, such that for any $e \in E$, $c_e > 0$. Now this means that $\sin(k_1 x + t \times k_2 x)$ should be positive for any $t \in \mathbb{N}$. Let $z \in \mathbb{N}$, then note that for any real v:

$$\sin(k_1 x + \frac{v - k_1 x + 2\pi z}{k_2 x} \times k_2 x) = \sin(v) \ .$$

Thus, for any $z \in \mathbb{N}$, any integer t in the range:

$$\left]\frac{-\pi - k_1 x + 2\pi z}{k_2 x}, \frac{-k_1 x + 2\pi z}{k_2 x}\right[\ ,$$

is such that $\sin(k_1 x + t \times k_2 x) < 0$. We reach a contradiction by showing that one such range contains an integer. Indeed, write $A = \frac{2\pi}{k_2 x}$, $B = \frac{-\pi - k_1 x}{k_2 x}$, and $C = \frac{\pi}{k_2 x}$; we are searching for a $z \in \mathbb{N}$ such that there is an integer in $[z.A + B, z.A + B + C]$. Now, as A is irrational, for any real interval $[a, b] \subseteq [0, 1]$ there is an integer z such that the fractional part of $z.A$ is in $[a, b]$ (see, e.g., [Niv63, Th. 3.2]). Thus we can find a z such that there is an integer t with the property that $z.A + B \in [t - C/2 - C/4, t - C/2 + C/4]$. This implies that $z.A + B < t < z.A + B + C$, proving the claim.

5 Conditional separations of APA and DetAPA

We first note the following fact:

▶ **Lemma 99.** *There is a polynomial p such that if $L \subseteq \Sigma^*$ is decided by a nondeterministic Turing machine in time-constructible time $t(n)$, then there is an alphabet T and a language $R \subseteq \bigcup_n (\Sigma^n \cdot \{\sharp\} \cdot \mathrm{T}^{p(t(n))})$ in $\mathscr{L}_{\text{DetAPA}}$, with $\Sigma \cap \mathrm{T} = \varnothing$ and $\sharp \notin \Sigma \cup \mathrm{T}$, such that: $R(\{\sharp\}\mathrm{T}^*)^{-1} = L$.*

Proof. Let $L \subseteq \Sigma^*$ be a language decided by a nondeterministic Turing machine M in time-constructible time $t(n)$. Suppose, w.l.o.g. as $t(n)$ is time-constructible, that M accepts or rejects any word w in exactly $t(|w|)$ steps. Let R be the language $\{w\sharp\rho \mid w \in L \wedge \rho \text{ is an accepting run for } w\}$, where ρ is written using the encoding of Paper I, Lemma 29: ρ is a sequence of instantaneous descriptions of the machine M, separated by a specific symbol $\natural$, and padded so that each description has the same length (say, again, $t(|w|)$). We may suppose that the alphabet T for the runs has an empty intersection with Σ, and that $\sharp$ is in none of the alphabets. Write $\tau(w)$ for the initial instantaneous description of the Turing machine M with input w.

Now note that $I = \{w\sharp\tau(w)\natural \mid w \in \Sigma^*\} \cdot \mathrm{T}^*$ is in $\mathscr{L}_{\mathrm{DetAPA}}$ (similar to Paper I, Proposition 25), that the language of the accepting runs $R' \subseteq \mathrm{T}^*$ of M is in $\mathscr{L}_{\mathrm{DetAPA}}$ (Paper I, Lemma 29), that $R'' = \Sigma^* \cdot \{\sharp\} \cdot R'$ is in $\mathscr{L}_{\mathrm{DetAPA}}$, and that $\mathscr{L}_{\mathrm{DetAPA}}$ is closed under intersection (Paper I, Proposition 24). This implies that $R = I \cap R'' \in \mathscr{L}_{\mathrm{DetAPA}}$.

We note that this implies the following unconditional nonclosure properties:

▶ **Corollary 100.** *None of $\mathscr{L}_{\mathrm{APA}}$, $\mathscr{L}_{\mathrm{UnAPA}}$, and $\mathscr{L}_{\mathrm{DetAPA}}$ is closed under quotient.*

Proof. Recall that $\mathscr{L}_{\mathrm{APA}} \subseteq \mathrm{NP}$. Let L be a language complete for NEXP. Lemma 99 asserts that there is an $R \in \mathscr{L}_{\mathrm{DetAPA}}$ such that there is a regular language S with $RS^{-1} = L$. As $L \notin \mathrm{NP}$, $L \notin \mathscr{L}_{\mathrm{APA}}$, implying the nonclosure by $S \in \mathscr{L}_{\mathrm{DetAPA}}$.

▶ **Corollary 101.** *There are* NP*-complete languages in $\mathscr{L}_{\mathrm{APA}}$.*

Proof. Let L be an NP-complete language decided by a nondeterministic Turing machine in polynomial time $t(n)$. With the notations of Lemma 99 applied to L, let $h\colon (\Sigma \cup \mathrm{T} \cup \{\sharp\})^* \to (\Sigma \cup \{\sharp\})^*$ be the nonerasing morphism mapping all the letters of T to $\sharp$, and leaving the other letters unchanged. Then:

$$h(R) = \{w\sharp^{t(|w|)^2+1} \mid w \in L\} \ .$$

Now $h(R) \in \mathscr{L}_{\mathrm{APA}}$, as $\mathscr{L}_{\mathrm{APA}}$ is closed under nonerasing morphisms, and $h(R)$ is clearly NP-complete.

Corollary 101 and the fact that $\mathscr{L}_{\mathrm{APA}} \subseteq \mathrm{NP}$ imply:

▶ **Corollary 102.** *If* co-NP $\neq$ NP *then $\mathscr{L}_{\mathrm{APA}}$ is not closed under complement.*

As $\mathscr{L}_{\text{DetAPA}}$ is closed under complement, if co-NP $\neq$ NP then $\mathscr{L}_{\text{DetAPA}} \neq \mathscr{L}_{\text{APA}}$. In fact, it is also a direct consequence of $\mathscr{L}_{\text{DetAPA}} \subseteq$ P and of Corollary 101 that:

▶ **Corollary 103.** *If* P $\neq$ NP *then* $\mathscr{L}_{\text{DetAPA}} \neq \mathscr{L}_{\text{APA}}$.

This implies, as any language of $\mathscr{L}_{\text{APA}}$ is the image by a length-preserving morphism of a language in $\mathscr{L}_{\text{DetAPA}}$, that if P $\neq$ NP then $\mathscr{L}_{\text{DetAPA}}$ is not closed under length-preserving morphism.

▶ **Proposition 104.** *If* NEXP $\neq$ EXP *then* $\mathscr{L}_{\text{DetAPA}}$ *is not closed under commutative closure.*

Proof. Let $E \in \{0,1\}^*$ be in NEXP. Define $\mathsf{un}(w)$, $w \in \{0,1\}^*$, to be the unary encoding of the binary number $1w$ over the alphabet $\Sigma = \{v\}$. Let $L = \mathsf{un}(E)$, it is decided by a nondeterministic Turing machine in polynomial time $t(n)$. Consider the language $R \in \mathscr{L}_{\text{DetAPA}}$ given by Lemma 99 applied to L; we proceed with the notations of Lemma 99. Write $\mathrm{T} = \{a_1, a_2, \ldots, a_k\}$, and define $h\colon \mathrm{T}^* \to \{x,y\}^*$ by $h(a_i) = x^i y^k x^{k-i}$. Now each word $h(a_i)$ is unique (among the other $h(a_j)$) and has exactly k letters x and y. Extend h over $(\Sigma \cup \mathrm{T} \cup \{\sharp\})^*$ by leaving the other letters unchanged. Then $h(R) \in \mathscr{L}_{\text{DetAPA}}$, as it suffices to replace each transition labeled $a \in \mathrm{T}$ of the DetAPA for R by a sequence of k transitions labeled $h(a)$, and this preserves determinism. Now:

$$c(h(R)) \cap v^* \cdot \{\sharp\} \cdot \{x\}^* \cdot \{y\}^* = \{w\sharp x^{k\times p(t(|w|))} y^{k\times p(t(|w|))} \mid w \in L\} \ .$$

Let X be the above language, then if $\mathscr{L}_{\text{DetAPA}}$ is closed under commutative closure, $X \in \mathscr{L}_{\text{DetAPA}}$, thus $X \in$ P, and as p and t are polynomials, $L \in$ P, implying that $E \in$ EXP, hence NEXP = EXP.

Thus if NEXP $\neq$ EXP, $\mathscr{L}_{\text{DetAPA}}$ and $\mathscr{L}_{\text{APA}}$ differ as:

▶ **Proposition 105.** $\mathscr{L}_{\text{APA}}$ *is closed under commutative closure.*

Proof. Let (A, U, C) be an APA of dimension d over the alphabet $\Sigma = \{a, b\}$. Write $A = (Q, \Sigma, \delta, q_0, F)$. Let A' be two copies of A, the second one having its transition labels inverted (i.e., a transition is labeled a iff it was labeled b in A). The initial state of A' is that of A, and the final states that of A and its copy. For a state $q \in A$, we write q^{c} for the equivalent state in the second copy. We add to A' the following transitions:

$$T = \{p \bullet\!\!-b\!\!\to q^{\mathrm{c}}, p^{\mathrm{c}} \bullet\!\!-a\!\!\to q \mid p \bullet\!\!-a\!\!\to q \in \delta\} \ .$$

Write $A' = (Q', \Sigma, \delta', q_0, F')$. First, we define $\tau\colon \delta' \to \delta$ by:

$$\tau(t) = \begin{cases} p \bullet\!\!-\ell\!\!\to q & \text{if } t = p \bullet\!\!-\ell\!\!\to q \text{ or } t = p^{\mathrm{c}} \bullet\!\!-\ell\!\!\to q,\ p, q \in Q\ , \\ p \bullet\!\!-a\!\!\to q & \text{if } t = p \bullet\!\!-b\!\!\to q^{\mathrm{c}} \text{ or } t = p^{\mathrm{c}} \bullet\!\!-b\!\!\to q^{\mathrm{c}},\ p, q \in Q\ , \\ p \bullet\!\!-b\!\!\to q & \text{if } t = p \bullet\!\!-a\!\!\to q^{\mathrm{c}} \text{ or } t = p^{\mathrm{c}} \bullet\!\!-a\!\!\to q^{\mathrm{c}},\ p, q \in Q\ , \end{cases}$$

that is, $\tau(t)$ is the noninverted transition t, in A. We now define $U'\colon (\delta')^* \to \mathscr{F}(d{+}1)$ by:

$$U'_t(\mathbf{x}, n) = \begin{cases} (U_{\tau(t)}(\mathbf{x}), n) & \text{if } \mu(t) = \mu(\tau(t)), \\ (U_{\tau(t)}(\mathbf{x}), n-1) & \text{if } \mu(t) = a \wedge \mu(\tau(t)) = b, \\ (U_{\tau(t)}(\mathbf{x}), n+1) & \text{if } \mu(t) = b \wedge \mu(\tau(t)) = a. \end{cases}$$

The extra component stores the number of times an a has been taken instead of a b minus the number of times a b has been taken instead of an a. If these two numbers are equal, then the numbers of a's and b's in the word read correspond to the numbers of a's and b's that would have been read without the transition labels inverted; this is the crucial point we show in what follows.

We will show that (1) for any path π in A from q_0 to a state $q \in Q$ and any word $w \in \{a, b\}^*$, there is a w-labeled path π' in A' from q_0 to either q or q^{c} such that $U'_{\pi'}(\mathbf{0}) = (U_\pi(\mathbf{0}), |\mu(\pi)|_a - |w|_a)$; conversely, we will show that (2) for any path π' in A' from q_0 to a state q or q^{c}, for $q \in Q$, $\pi = \tau(\pi')$ is a path in A from q_0 to q such that $U'_{\pi'}(\mathbf{0}) = (U_\pi(\mathbf{0}), |\mu(\pi)|_a - |\mu(\pi')|_a)$.

We first show how (1) and (2) imply the proposition. Define $C' = C \times \{0\}$, then we claim that $L(A', U', C') = c(L(A, U, C))$. Indeed, let $\pi \in \mathsf{Run}(A)$ such that $U_\pi(\mathbf{0}) \in C$, and let $w \in c(\mu(\pi))$. Then, by (1) there is an w-labeled accepting path π' in A' such that $U_{\pi'}(\mathbf{0}) = (U_\pi(\mathbf{0}), 0)$, which is in C', and thus $w \in L(A', U', C')$. Conversely, let $\pi' \in \mathsf{Run}(A')$ such that $U'_{\pi'}(\mathbf{0}) \in C'$. Then, by (2) $\pi = \tau(\pi')$ is in $\mathsf{Run}(A)$, $|\mu(\pi)|_a = |\mu(\pi')|_a$, implying that the two words have the same Parikh image, and $U_\pi(\mathbf{0}) \in C$. Hence $\mu(\pi) \in L(A, U, C)$, which implies that $\mu(\pi') \in c(L(A, U, C))$.

We now show (1) by induction. This is clear if π is the empty path, from q_0 to q_0. Now let $\pi = \rho t$ and $w = u\ell$, and set $q = \mathsf{To}(\rho)$. Now suppose, by the induction hypothesis, that there is a u-labeled path ρ' in A' from q_0 to either q or q^{c} such that $U'_{\rho'}(\mathbf{0}) = (U_\rho(\mathbf{0}), |\mu(\rho)|_a - |u|_a)$. If $\mu(t) = \ell$, then define $\pi' = \rho'(\mathsf{To}(\rho') \bullet\!\!-\ell\!\!\to \mathsf{To}(t))$, otherwise define $\pi' = \rho'(\mathsf{To}(\rho') \bullet\!\!-\ell\!\!\to (\mathsf{To}(t))^{\mathrm{c}})$. It is now straightforward to check that π' fulfills the conditions of (1).

We show (2) by induction. Again, this is clear if $\pi' = \varepsilon$. Let $\pi' = \rho' t'$, and let the ending state of ρ' be q or q^{c}, for $q \in Q$. By the induction hypothesis, $\rho = \tau(\rho')$ is a path in A from q_0 to q such that $U'_{\rho'}(\mathbf{0}) = (U_\rho(\mathbf{0}), |\mu(\rho)|_a - |\mu(\rho')|_a)$. Now $\pi = \rho\tau(t') = \tau(\pi')$ is such that $U'_{\pi'}(\mathbf{0}) = (U_\pi(\mathbf{0}), |\mu(\rho)|_a - |\mu(\rho')|_a + k)$ where $k = 0$ if $\mu(t') = \mu(\tau(\pi'))$, $k = 1$ if $\mu(t') = b$ and $\mu(\tau(\pi')) = a$ (thus if $|\mu(\pi)|_a - |\mu(\pi')|_a = |\mu(\rho)|_a - |\mu(\rho')|_a + 1$), and $k = -1$ if $\mu(t') = a$ and $\mu(\tau(\pi')) = b$ (thus if $|\mu(\pi)|_a - |\mu(\pi')|_a = |\mu(\rho)|_a - |\mu(\rho')|_a - 1$). In all cases we indeed have that $U'_{\pi'}(\mathbf{0}) = (U_\pi(\mathbf{0}), |\mu(\pi)|_a - |\mu(\pi')|_a)$. It is also clear that π fulfills the other conditions of (2).

We indicate the two modifications that should be made for an alphabet size $n > 2$. First, instead of 2 copies of A, there should be $n!$ copies, each for a permutation of the alphabet. Second, the functions U'_t should have n registers, where the i-th is incremented if $\mu(t)$ is the i-th letter and decremented if $\mu(\tau(t))$ is the i-th letter. All those counters should reach 0 for acceptance.

We add the following closure property of $\mathscr{L}_{\text{APA}}$, which we conjecture does not hold for $\mathscr{L}_{\text{DetAPA}}$:

▶ **Proposition 106.** *$\mathscr{L}_{\text{APA}}$ is closed under starring.*

Proof. We first present the idea of the proof. Let (A, U, C) be an APA of dimension d, and suppose, to simplify this sketch, that C is linear, i.e., $C = \{\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k \mid \mathbf{a} \in \mathbb{N}^k\}$. Let $C(\mathbf{a})$ for $\mathbf{a} \in \mathbb{N}^k$ be the value $\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k$. The construction of the proof is then as follows. The automaton A guesses during a run π a value $\mathbf{a} \in \{0, 1, \ldots, c^{|\pi|} - 1\}^k$, for c a big enough constant. Then, at the end of its computation, and before it is reset, the value $\mathbf{z} = U_\pi(\mathbf{0}) - C(\mathbf{a})$ is added to an additional register $\mathbf{v}$ (of dimension d). Now if $U_\pi(\mathbf{0})$ is in C, there is an $\mathbf{a}$ such that $\mathbf{z} = \mathbf{0}$; conversely, if $\mathbf{z} = \mathbf{0}$, then $U_\pi(\mathbf{0})$ is in C. Thus if the right guesses are taken and $\mu(\pi) \in L(A, U, C)$, $\mathbf{0}$ will be added to $\mathbf{v}$, and conversely, no sequence of guesses exist that would make $\mathbf{0}$ to be added to $\mathbf{v}$ if $\mu(\pi) \notin L(A, U, C)$. Now we rely on the technique presented in Paper I, Lemma 28 to make the conjunction of an unbounded number of conditions by maintaining a nonzero flag: each time a transition is taken, the value of $\mathbf{v}$ is multiplied by c; this ensures that if at any reset a nonzero value $\mathbf{z}$ is added to $\mathbf{v}$, this latter vector will stay nonzero for the rest of the computation. Thus a word in $(L(A))^*$ is in $(L(A, U, C))^*$ iff $\mathbf{v}$ is all-zero at the end of the computation.

We now present formal arguments. Let (A, U, C) be an APA of dimension d with $A = (Q, \Sigma, \delta, q_0, F)$. Write $C = C_1 \cup C_2 \cup \cdots \cup C_n$ where all the C_i's are linear.

Let $C_i = \{\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_\ell\mathbf{v}_\ell \mid \mathbf{a} \in \mathbb{N}^\ell\}$, with $\mathbf{v}_j \in \mathbb{N}^d$, and define k as the maximum of the ℓ's for all C_i. We write $C_i(\mathbf{a})$ for $\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_\ell\mathbf{v}_\ell$, with $\mathbf{a} \in \mathbb{N}^k$. Note that if $\mathbf{v} \in C_i$ then there is a vector $\mathbf{a} \in \{0, 1, \ldots, \max \mathbf{v}\}^k$ such that $\mathbf{v} = C_i(\mathbf{a})$. We let c be a constant such that for any path π of A, any component of $U_\pi(\mathbf{0})$ is strictly less than $c^{|\pi|}$ (see, e.g., Paper I, Proposition 27). In particular, this implies that $U_\pi(\mathbf{0}) \in C_i$ iff there is $\mathbf{a} \in \{0, \ldots, c^{|\pi|} - 1\}^k$ such that $U_\pi(\mathbf{0}) = C_i(\mathbf{a})$.

Define $A' = (Q, \Sigma, \delta', q_0, F)$ as the starring of A, i.e.:

$$\delta' = \delta \cup \{q \bullet\!\!-a\!\!\rightarrow p \mid a \in \Sigma \wedge q \in F \wedge q_0 \bullet\!\!-a\!\!\rightarrow p \in \delta\} \ .$$

We may suppose that all the added transitions did not appear in δ originally (this could require that we duplicate the automaton and jump from one copy to the other); we call these transitions the *resetting transitions*. Next, define A'' as n copies of A', and let each resetting transition choose nondeterministically one of the n copies to continue the computation; we call these copies the *linear copies* and let the first one be initial (i.e., the initial state of A'' is that of the first linear copy). Now define A^{g} to be c^k copies of A'', which we refer to using vectors in $\{0, 1, \ldots, c-1\}^k$. Each time a transition is taken, it chooses nondeterministically to continue the computation in one of the copies; we call these copies the *period copies* and again let the first one (corresponding to the all-zero vector) be initial. Write the elements of the transition set δ^{g} of A^{g} as $t_{i,\mathbf{x}}$, for $t \in \delta'$, $1 \le i \le n$, $\mathbf{x} \in \{0, 1, \ldots, c-1\}^k$, if it goes in the $\mathbf{x}$-th period copy of the i-th linear copy. Now define $U^{\mathrm{g}} : (\delta^{\mathrm{g}})^* \to \mathscr{F}(2d + k)$, with $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^d$ and $\mathbf{a} \in \mathbb{Z}^k$ by letting $U^{\mathrm{g}}_{t_{i,\mathbf{x}}}(\mathbf{u}, \mathbf{v}, \mathbf{a})$ be:

$$\begin{cases} (U_t(\mathbf{u}), c \times \mathbf{v}, c \times \mathbf{a} + \mathbf{x}) & \text{if } t \text{ is not resetting,} \\ (U_{q_0 \bullet\!-a\!\rightarrow p}(\mathbf{0}), c \times \mathbf{v} + \mathbf{u} - C_i(\mathbf{a}), \mathbf{x}) & \text{if } t = q \bullet\!\!-a\!\!\rightarrow p \text{ is resetting.} \end{cases}$$

Let π^{g} be a path in A^{g} starting from the initial state to a final state not going through any reset. Then $U^{\mathrm{g}}_{\pi^{\mathrm{g}}}(\mathbf{0}, \mathbf{x}, \mathbf{0}) = (\mathbf{u}, \mathbf{v}, \mathbf{a})$ is such that $\mathbf{u} = U_\pi(\mathbf{0})$, with π the equivalent path in A, $\mathbf{v} = c^{|\pi^{\mathrm{g}}|} \times \mathbf{x}$, and $\mathbf{a}$ is a vector of numbers in $\{0, 1, \ldots, c^{|\pi^{\mathrm{g}}|-1}\}$. In fact, for any path π in A, there is an equivalent path in A^{g} for any value of $\mathbf{a}$. Thus, we view the work of A^{g} and U^{g} as taking a path π in A while guessing a vector $\mathbf{a}$ and multiplying $\mathbf{x}$ by $c^{|\pi|}$. Now when a resetting transition is taken, $\mathbf{u}$ and $\mathbf{a}$ are reset, and $U_\pi(\mathbf{0}) - C_1(\mathbf{a})$ is added to $\mathbf{x}$. This value is 0 if $U_\pi(\mathbf{0}) \in C_1$ and the right guess of the values $\mathbf{a}$ has been made, i.e., $U_\pi(\mathbf{0}) = C_1(\mathbf{a})$; as previously mentioned, if $U_\pi(\mathbf{0}) \in C_1$, these values exist. As in Paper I, Lemma 28 the component $\mathbf{x}$ is such that if any nonzero value is added to it, it will stay nonzero throughout the computation. This behavior is similar if the path starts in the i-th linear copy, but with C_i instead of C_1.

Now augment U^{g} so that an additional component indicates the number $1 \leq i \leq n$ of the linear copy in which the computation ended: $U'_{t_{i,\mathbf{x}}}(\cdot, j) = (U^{\mathrm{g}}_{t_{i,\mathbf{x}}}(\cdot), i)$. Then define:

$$C^{\mathrm{g}} = \{(\mathbf{u}, \mathbf{v}, \mathbf{a}, i) \mid \mathbf{v} = 0 \wedge C_i(\mathbf{a}) = \mathbf{u}\} \enspace .$$

This is such that $L(A^{\mathrm{g}}, U', C^{\mathrm{g}}) = L(A, U, C_1) \cdot (L(A, U, C))^*$. Now with A^{g}_i the automaton A^{g} but with the i-th linear copy initial, this leads to:

$$\{\varepsilon\} \cup \bigcup_{1 \leq i \leq n} L(A^{\mathrm{g}}_i, U', C^{\mathrm{g}}) = (L(A, U, C))^* \enspace .$$

6 Conclusion

We showed that the classes of languages defined by deterministic and unambiguous CA and APA are recognized by natural algebraic objects in the context of typed monoids. We showed that deterministic APA are strongly linked to $\mathbb{Z}$-valued rational series. We investigated closure properties of APA and deterministic APA in the light of computational complexity assumptions, and showed, under these assumptions, separation of the classes of languages they define.

We propose several avenues of research. First, the algebraic characterizations we give may lead to a finer understanding of the circuit complexity of the classes of languages we studied. We hope that further study will help shed further light on small circuit complexity classes. Second, we show only *conditional* separation results of DetAPA and APA. The characterization of DetAPA using rational series could be the starting point of an unconditional separation, in particular over unary languages. Indeed, we do not believe that the language $\{a^{2^n} \mid n \in \mathbb{N}\}$, a language of $\mathscr{L}_{\mathrm{APA}}$, is in $\mathscr{L}_{\mathrm{DetAPA}}$; or, equivalently (similarly to Corollary 98), we do not believe that there is a sequence defined by a linear recurrence relation which is positive only on powers of two.

Discussion

This paper in preparation answers primarily the question: what are the algebraic structures corresponding to the classes of languages we have defined and studied over the previous papers. However, two obvious missing classes are $\mathscr{L}_{\text{CA}}$ and $\mathscr{L}_{\text{APA}}$. These two classes are not closed or expected to be closed under complement, thus they do not fit in the framework of typed monoids. One possible solution for this characterization would be to relax the requirement of having a Boolean algebra of types in typed monoids, that is, going in the direction of the *pointed-monoids* of Sakarovitch (e.g., [Sak76]). However, this seems to introduce technical difficulties and a better framework appears to be that of a typed variant of *ordered monoids*. An ordered monoid is a monoid equipped with a partial order $\preceq$, and it recognizes a language L if L is the inverse morphic image of an *upper subset* of the monoid (that is, a set S such that $x \in S$ and $x \preceq y$ implies $y \in S$). In this context, it has been suggested by Krebs that the *Schützenberger product* (see, e.g., [Sim93]), which is a natural algebraic interpretation of language product, could be used to characterize $\mathscr{L}_{\text{CA}}$. We note that Pin [Pin03] recently proposed a definition of the Schützenberger product for ordered monoids. Also, we saw that $\mathscr{L}_{\text{DetCA}}$ is a variety of languages, and [BKR11, Thm. 2] states an Eilenberg-like theorem, associating natural sets of typed monoids with varieties of languages. Note that $\mathscr{L}_{\text{CA}}$ is a *positive* variety of languages, that is, it is closed under union, intersection, inverse morphisms, and quotient by a word. As an Eilenberg-like theorem exists associating positive varieties of *regular* languages with natural classes of *finite* ordered monoids (see [Pin11] for a survey), it would be interesting to investigate such a correspondence between positive varieties of languages and typed ordered monoids.

Another problem left open in this work is whether $\mathscr{L}_{\text{CA}} \subseteq \text{NC}^1$. We hope that the final version of this paper will include a more precise result in this direction.

We also note that we did not derive from those results an additional characterization for $\mathscr{L}_{\text{DetCA}}$. We may however be able to show a typed equivalent of the *wreath product principle* of Straubing [Str79] (see [Pin03] for a modern account). This principle states that a language $L \subseteq \Sigma^*$ is recognizable by $M \wr N$ iff it is a finite union of languages of the form $U \cap \sigma^{-1}(V)$, for $U \subseteq \Sigma^*$ a language recognized by N, $V \subseteq \mathrm{T}^*$ a language recognized by M, and $\sigma\colon \Sigma^* \to \mathrm{T}^*$ a rational transduction.

The other main problem left open is whether $\mathscr{L}_{\text{APA}} = \mathscr{L}_{\text{DetAPA}}$, in particular over unary languages. We note that the characterization of $\mathscr{L}_{\text{DetAPA}}$ by means of rational series appears to be be a good venue to show this separation; indeed, it seems unlikely that, for instance, a sequence specified by a linear recurrence relation can be positive only at positions which are powers of two. We note that this question seems related to the theorem of Skolem, Mahler, and Lech, as presented in [BR10, Th. 6.4.1]. This theorem states that for any rational series r over a unary alphabet, $\{a^n \mid (r, a^n) = 0\}$ is regular. However, we expect a characterization of the unary languages of $\mathscr{L}_{\text{DetAPA}}$ to rely on a study of the sign of trigonometric polynomials (in the sense of [Cor89]) rather than p-adic analysis, as in the proof of this theorem.

The algebraic framework has several advantages, one of them being conciseness. It is interesting to compare, for instance, the proof of Corollary 96 of the closure of $\mathscr{L}_{\text{UnAPA}}$ under reversal with the following proof of closure of $\mathscr{L}_{\text{APA}}$. This relies on Lemma 89, which already simplifies the presentation of APA, hence the gain provided by algebra as compared to the following is the abstraction from the automaton.

▶ **Proposition 107.** *The class $\mathscr{L}_{\text{APA}}$ is closed under reversal.*

Proof. Let $L \in \mathscr{L}_{\text{APA}}$. Lemma 89 asserts that there is an automaton A with transition set δ, a morphism $h\colon \delta^* \to \mathscr{M}_{\mathbb{Z}}(k)$, for some k, and a type $\mathscr{X}$ of $(\mathbb{Z}, \mathbb{Z}^+)^{k^2}$, such that $L = \mu_A(h^{-1}(\mathscr{X}) \cap \mathsf{Run}(A))$.

Suppose that A has only one final state q_{f}. Let B be the automaton A where every transition is reversed (i.e., $q \bullet\!\!-a\!\!\to q'$ is a transition of A iff $q' \bullet\!\!-a\!\!\to q$ is a transition of B), the initial state of B is q_{f} and the only accepting state of B is the initial state of A.

Define g to be the morphism from the transitions of B to $\mathscr{M}_{\mathbb{Z}}(k)$ defined as $g(q \bullet\!\!-a\!\!\to q') = (h(q' \bullet\!\!-a\!\!\to q))^{\mathrm{T}}$. Let π be an accepting path in A and π' the accepting path in B obtained from π^{R} by reversing each transition. Then clearly $\mu_A(\pi) = (\mu_A(\pi'))^{\mathrm{R}}$ and $h(\pi) = g(\pi')^{\mathrm{T}}$. As $\mathsf{Run}(B)$ is the set of paths obtained from $\mathsf{Run}(A)$ in this fashion:

$$(\mu_A(h^{-1}(\mathscr{X}) \cap \mathsf{Run}(A)))^{\mathrm{R}} = \mu_B(g^{-1}(\mathscr{X}^{\mathrm{T}}) \cap \mathsf{Run}(B)) \ .$$

Now such a language is that of an APA — this is shown similarly as the inclusion $H^{-1}(\mathbf{ZMat}^{+}) \subseteq \mathscr{L}_{\text{DetAPA}}$ in Theorem 95.

If A has $f > 1$ final state, then we can decompose L into a finite union of f languages for which A has only one state, and use the $f = 1$ case above. Closure under union then completes the proof.

Conclusion

This thesis focused on several aspects of models of computation combining finite automata and semilinear constraints. It was presented through four research papers. The main results are:

- We investigated in greater details the expressive power of Parikh automata. To this aim, we introduced the constrained automaton which helped in gaining intuition and writing short proofs. Different pumping-style lemmata allowed us to give languages witnessing separations among flavors of Parikh automata. We showed that determinism does not affect the expressive power of Parikh automata over bounded languages.

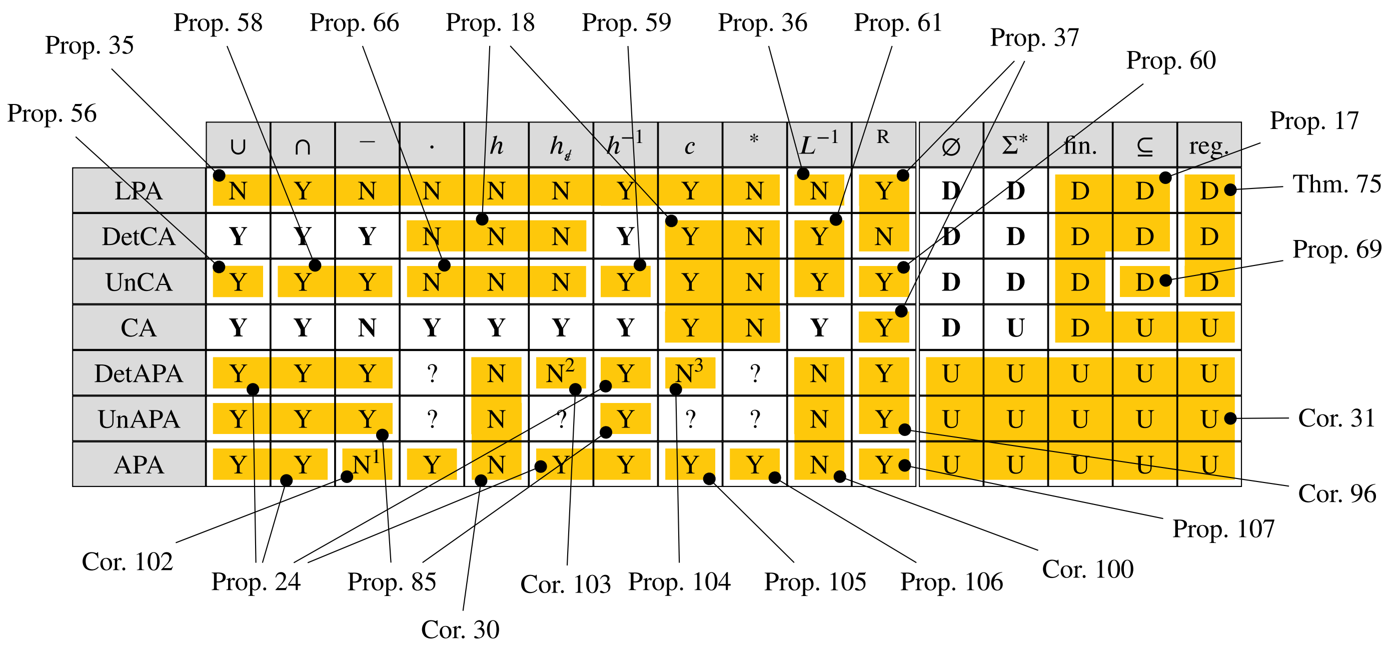
- We further studied the closure and decidability properties of Parikh automata, completing the picture started by Klaedtke and Rueß. These properties are summarized in Figure 3. Showing the decidability of regularity for deterministic Parikh automata proved to be challenging.

- We studied the *unambiguous* constrained automaton, resulting in a model with a better compromise between closure and decidability properties, on the one hand, and expressiveness, on the other hand, than deterministic or nondeterministic constrained automata.

- We introduced the affine Parikh automaton over $\mathbb{Q}$ and $\mathbb{N}$. We studied its expressiveness, decidability, and closure properties (see Figure 3). We investigated natural restrictions of this model, in particular to make the emptiness problem decidable. We showed in this vein that deterministic affine Parikh automata with a condition expressing the finiteness of its behavior is equivalent to unambiguous constrained automata.

- We used computational complexity assumptions to complete our pictures of the world of Parikh automata (Figures 3 and 4).
- We proposed algebraic equivalents to the classes of languages defined by deterministic and unambiguous contrained automata and affine Parikh automata. We deduced circuit complexity upper bounds and logical characterizations for unambiguous constrained automata.

Figures 3 and 4 summarize expressiveness, decidability, closure results, and the hierarchy of the classes of languages we studied. Open questions appear in these figures.



	$\cup$	$\cap$	$-$	$\cdot$	h	$h_{\not\varepsilon}$	h^{-1}	c	$*$	L^{-1}	R	$\varnothing$	Σ^*	fin.	$\subseteq$	reg.
LPA	N	Y	N	N	N	N	Y	Y	N	N	Y	**D**	**D**	D	D	D
DetCA	**Y**	**Y**	**Y**	N	N	N	**Y**	Y	N	Y	N	**D**	**D**	D	D	D
UnCA	Y	Y	Y	N	N	N	Y	Y	N	Y	Y	**D**	**D**	D	D	D
CA	**Y**	**Y**	**N**	**Y**	**Y**	**Y**	**Y**	Y	N	**Y**	Y	**D**	**U**	D	U	U
DetAPA	Y	Y	Y	?	N	N[2]	Y	N[3]	?	N	Y	U	U	U	U	U
UnAPA	Y	Y	Y	?	N	?	Y	?	?	N	Y	U	U	U	U	U
APA	Y	Y	N[1]	Y	N	Y	Y	Y	Y	N	Y	U	U	U	U	U

Prop. 35
Prop. 58
Prop. 66
Prop. 18
Prop. 59
Prop. 36
Prop. 61
Prop. 37
Prop. 60
Prop. 17
Prop. 56
Thm. 75
Prop. 69
Cor. 31
Cor. 96
Prop. 107
Cor. 100
Cor. 102
Prop. 24
Prop. 85
Cor. 30
Cor. 103
Prop. 104
Prop. 105
Prop. 106

Figure 3: Closure in the effective sense (Y) or nonclosure (N) of language classes defined by PA variants, under set operations, concatenation, morphisms, nonerasing morphisms, inverse morphisms, commutation, iteration, quotient, and reversal; decidability (D) or undecidability (U) of emptiness, universality, finiteness, inclusion, and regularity; boldface denotes results known *prior* to this paper. [1]: if CoNP $\neq$ NP (see Cor. 102). [2]: if P $\neq$ NP (see Cor. 103). [3]: if EXP $\neq$ NEXP (see Prop. 104).

We propose several avenues of research:

- Is model-checking tractable using deterministic or unambiguous constrained automata? Or using moving deterministic affine Parikh automata, which are equivalent to unambiguous constrained automata? We observed that deterministic

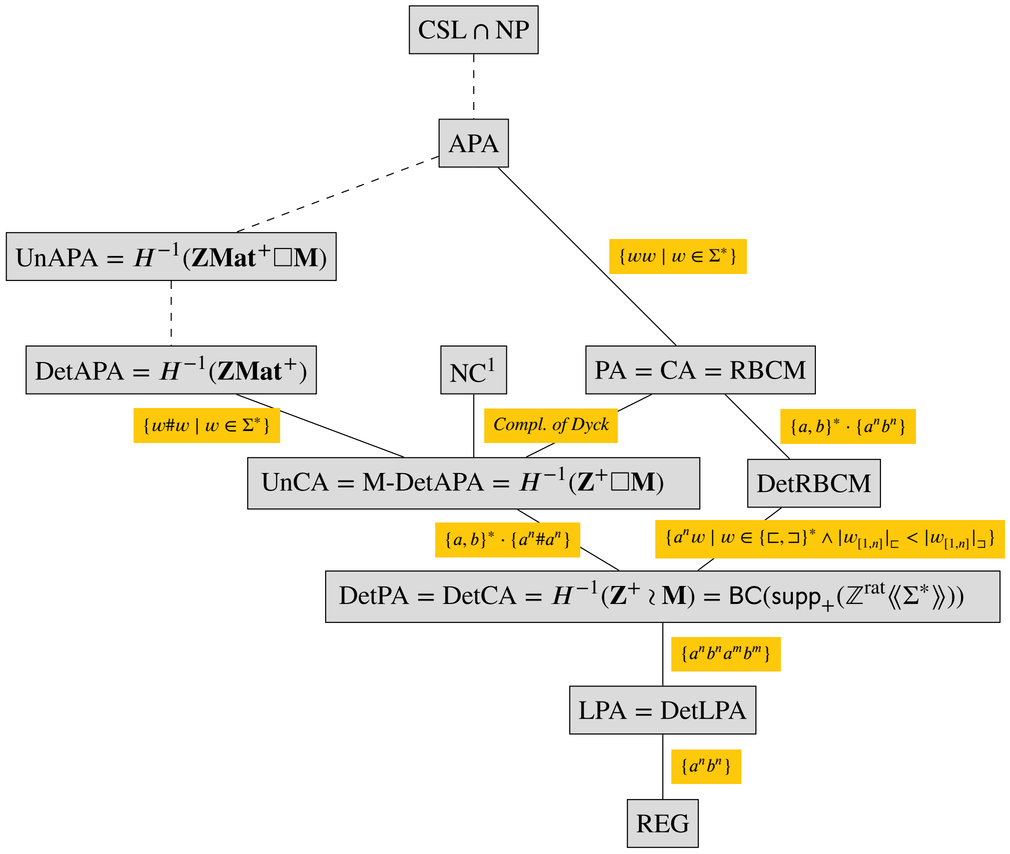

$\mathrm{CSL} \cap \mathrm{NP}$
APA
$\mathrm{UnAPA} = H^{-1}(\mathbf{ZMat}^{+} \square \mathbf{M})$
$\{ww \mid w \in \Sigma^*\}$
$\mathrm{DetAPA} = H^{-1}(\mathbf{ZMat}^{+})$
NC^1
$\mathrm{PA} = \mathrm{CA} = \mathrm{RBCM}$
$\{w\#w \mid w \in \Sigma^*\}$
Compl. of Dyck
$\{a,b\}^* \cdot \{a^n b^n\}$
$\mathrm{UnCA} = \text{M-DetAPA} = H^{-1}(\mathbf{Z}^{+} \square \mathbf{M})$
DetRBCM
$\{a,b\}^* \cdot \{a^n \# a^n\}$
$\{a^n w \mid w \in \{\sqsubset, \sqsupset\}^* \wedge |w_{[1,n]}|_{\sqsubset} < |w_{[1,n]}|_{\sqsupset}\}$
$\mathrm{DetPA} = \mathrm{DetCA} = H^{-1}(\mathbf{Z}^{+} \wr \mathbf{M}) = \mathsf{BC}(\mathsf{supp}_{+}(\mathbb{Z}^{\mathrm{rat}}\langle\langle \Sigma^* \rangle\rangle))$
$\{a^n b^n a^m b^m\}$
$\mathrm{LPA} = \mathrm{DetLPA}$
$\{a^n b^n\}$
REG

Figure 4: Relationships between language classes. When two classes are linked, the lower one is included in the upper one. The inclusion is strict and witnessed by the given language if the line is solid; it is not known to be strict otherwise.

constrained automata over bounded languages were already used in this context by Bouajjani and Habermehl [BH99]; can our characterization of the class of bounded languages recognized by deterministic constrained automata as the bounded semilinear languages (Paper II) motivate further use of this computing model?

- Are one-success constrained automata different from constrained automata?

- Can we prove an unconditional separation between the classes of languages recognized by deterministic, unambiguous, and nondeterministic affine Parikh automata? Is this question so tightly linked to computational complexity that showing the separation would solve a major open problem?

- Using a notion of typed ordered monoids, can the classes of languages recognized by constrained automata and affine Parikh automata be characterized algebraically?

- Can the algebraic characterizations provide us with fine upper bounds on the circuit complexity of the classes of languages we study? Can they help us in showing that the Dyck languages are not recognized by affine Parikh automata? Or that $\{a, b\}^* \cdot \{a^n b^n\} \cdot \{a, b\}^*$ is not in $\mathscr{L}_{\text{UnCA}}$?

Bibliography

[AM09] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56:16:1–16:43, 2009.

[BB74] Brenda S. Baker and Ronald V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8(3):315–332, 1974.

[BBF$^+$01] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, Philippe Schnoebelen, and Pierre McKenzie. *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer, 2001.

[BFLS05] Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Philippe Schnoebelen. Flat acceleration in symbolic model checking. In *Automated Technology for Verification and Analysis*, volume 3707 of *Lecture Notes in Computer Science*, pages 474–488. Springer, 2005.

[BH99] Ahmed Bouajjani and Peter Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. *Theoretical Computer Science*, 221(1–2):211–250, 1999.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci*, 41(3):274–306, 1990.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BKM07] Christoph Behle, Andreas Krebs, and Mark Mercer. Linear circuits, two-variable logic and weakly blocked monoids. In *Mathematical Foundations of Computer Science*, volume 4708 of *Lecture Notes in Computer Science*, pages 147–158. Springer-Verlag, 2007.

[BKR11] Christoph Behle, Andreas Krebs, and Stephanie Reifferscheid. Typed monoids - an Eilenberg-like theorem for non regular languages. In *Algebraic Informatics*, volume 6742 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2011.

[BL81] Meera Blattner and Michel Latteux. Parikh-bounded languages. In *Automata, Languages and Programming*, volume 115 of *Lecture Notes in Computer Science*, pages 316–323. Springer Berlin / Heidelberg, 1981.

[BNP74] Ronald Book, Maurice Nivat, and Michael Paterson. Reversal-bounded acceptors and intersections of linear languages. *SIAM Journal on Computing*, 3(4):283, 1974.

[Boj09] Mikolaj Bojanczyk. Weak MSO with the unbounding quantifier. In *Theoretical Aspects of Computer Science*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 159–170. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009.

[BR10] Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2010.

[Bra81] Franz-Josef Brandenburg. Analogies of PAL and COPY. In *Fundamentals of Computation Theory*, volume 117 of *Lecture Notes in Computer Science*, pages 61–70. Springer Berlin / Heidelberg, 1981.

[Büc60] J. Richard Büchi. Weak second order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

[CDI$^+$11] Ehsan Chiniforooshan, Mark Daley, Oscar H. Ibarra, Lila Kari, and Shinnosuke Seki. One-reversal counter machines and multihead automata: Revisited. In *Current Trends in Theory and Practice of Computer Science*, volume 6543 of *Lecture Notes in Computer Science*, pages 166–177. Springer, 2011.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Trans. Info. Theory*, 1:113–124, 1956.

[CMMP10] Christian Choffrut, Andreas Malcher, Carlo Mereghetti, and Beatrice Palano. On the expressive power of FO[+]. In Adrian-Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications*, volume 6031 of *Lecture Notes in Computer Science*, pages 190–201. Springer Berlin / Heidelberg, 2010.

[Col12] Thomas Colcombet. Forms of determinism for automata (invited talk). In *Theoretical Aspects of Computer Science*, volume 14 of *Leibniz International Proceedings in Informatics*, pages 1–23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Theory of Computing*, pages 151–158. ACM, 1971.

[Cor89] Constantin Corduneanu. *Almost Periodic Functions*. Chelsea Publishing Series. Chelsea Publishing Company, 1989.

[DFGvD10] Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Model-checking (CTL*) over flat Presburger counter systems. *Journal of Applied Non-Classical Logics*, 20(4):313–344, 2010.

[DIB$^+$00] Zhe Dang, Oscar H. Ibarra, Tevfik Bultan, Richard A. Kemmerer, and Jianwen Su. Binary reachability analysis of discrete pushdown timed automata. In *Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2000.

[DV08] Flavio D'Alessandro and Stefano Varricchio. On the growth of context-free languages. *J. Autom. Lang. Comb.*, 13:95–104, 2008.

[DZL03] Silvano Dal-Zilio and Denis Lugiez. XML schema, tree logic and sheaves automata. In *Rewriting Techniques and Applications*, volume 2706 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2003.

[Eil76] Samuel Eilenberg. *Automata, Languages, and Machines, Volume B*. Pure and Applied Mathematics. Academic Press, 1976.

[End72] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[ES69] Samuel Eilenberg and Marcel-Paul Schützenberger. Rational sets in commutative monoids. *Journal of Algebra*, 13:173–191, 1969.

[Fis65] Patrick C. Fischer. Multi-tape and infinite-state automata—a survey. *Commun. ACM*, 8(12):799–805, 1965.

[FIS03] Alain Finkel, S. Purushothaman Iyer, and Grégoire Sutre. Well-abstracted transition systems: application to FIFO automata. *Information and Computation*, 181(1):1–31, 2003.

[FL02] Alain Finkel and Jérôme Leroux. How to compose Presburger-accelerations: applications to broadcast protocols. In *Proc. 22nd Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'2002), Kanpur*, pages 145–156. Springer, 2002.

[FL08] Alain Finkel and Jérôme Leroux. Presburger functions are piecewise linear. Research Report LSV-08-08, Laboratoire Spécification et Vérification, ENS Cachan, France, 2008.

[Gin66] Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., New York, NY, USA, 1966.

[Gin68] Abraham Ginzburg. *Algebraic Theory of Automata*. Academic Press, New York, 1968.

[GMM10] Pierre Ganty, Rupak Majumdar, and Benjamin Monmege. Bounded underapproximations. In *Computer Aided Verification*, pages 600–614, 2010.

[Gre68] Sheila A. Greibach. A note on undecidable properties of formal languages. *Math Systems Theory*, 2(1):1–6, 1968.

[Gre81] Sheila A. Greibach. Formal languages: Origins and directions. *Annals of the History of Computing*, 3(1):14–41, 1981.

[GS64] Seymour Ginsburg and Edwin H. Spanier. Bounded ALGOL-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

[GS66a] Seymour Ginsburg and Edwin H. Spanier. Bounded regular sets. *Proceedings of the American Mathematical Society*, 17(5):1043–1049, 1966.

[GS66b] Seymour Ginsburg and Edwin H. Spanier. Finite-turn pushdown automata. *SIAM Journal on Control and Optimization*, 4(3):429, 1966.

[GS66c] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[HU79] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

[Iba78] Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

[IL81] Oscar H. Ibarra and Brian S. Leininger. Characterizations of Presburger functions. *SIAM Journal on Computing*, 10(1):22–39, 1981.

[IS99] Oscar H. Ibarra and Jianwen Su. A technique for proving decidability of containment and equivalence of linear constraint queries. *J. Comput. Syst. Sci.*, 59(1):1–28, 1999.

[IS11] Oscar H. Ibarra and Shinnosuke Seki. Characterizations of bounded semilinear languages by one-way and two-way deterministic machines. In *Automata and Formal Languages*, pages 211–224. Institute of Mathematics and Computer Science of Nyíregyháza College, 2011.

[ISD$^+$02] Oscar H. Ibarra, Jianwen Su, Zhe Dang, Tevfik Bultan, and Richard A. Kemmerer. Counter machines and verification problems. *Theor. Comput. Sci.*, 289(1):165–189, 2002.

[Jac09] Nathan Jacobson. *Basic algebra*. Dover Publications, Mineola, N.Y, 2009.

[Kar04] Wong Karianto. Parikh automata with pushdown stack. Diploma thesis, RWTH Aachen, 2004.

[Kar05] Wong Karianto. Adding monotonic counters to automata and transition graphs. In *Developments in Language Theory*, volume 3572 of *Lecture Notes in Computer Science*, pages 308–319. Springer, 2005.

[KF94] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[KKT06] Wong Karianto, Aloys Krieg, and Wolfgang Thomas. On intersection problems for polynomially generated sets. In *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 516–527. Springer, 2006.

[KLR07] Andreas Krebs, Klaus-Jörn Lange, and Stephanie Reifferscheid. Characterizing TC^0 in terms of infinite groups. *Theory of Computing Systems*, 40(4):303–325, 2007.

[KR02] Felix Klaedtke and Harald Rueß. Parikh automata and monadic second-order logics with linear cardinality constraints. Tech. rep. 177, Universität Freiburg, 2002.

[KR03] Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *International Colloquium on Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 681–696. Springer-Verlag, 2003.

[Kre08] Andreas Krebs. *Typed semigroups, majority logic, and threshold circuits*. PhD thesis, Eberhard Karls University of Tübingen, 2008.

[KS89] Nils Klarlund and Fred B. Schneider. Verifying safety properties using non-deterministic infinite-state automata. Technical report, Cornell University, Ithaca, NY, USA, 1989.

[Kur64] Sige-Yuki Kuroda. Classes of languages and linear bounded automata. *Information and Control*, 7(2):207–223, 1964.

[Lat78] Michel Latteux. Mots infinis et langages commutatifs. *R.A.I.R.O. Informatique Théorique*, 12(3):185–192, 1978.

[Ler05] Jérôme Leroux. A polynomial time Presburger criterion and synthesis for number decision diagrams. In *Logic in Computer Science*, pages 147–156. IEEE Computer Society, 2005.

[LK05] L. P Lisovik and D. A Koval'. Language recognition by two-way deterministic pushdown automata. *Cybernetics and Systems Analysis*, 40:939–942, 2005.

[Lot97] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, second edition, 1997.

[Maz10] David R. Mazur. *Combinatorics: A Guided Tour*. Mathematical Association of Mathematics, 2010.

[Min61] Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):pp. 437–455, 1961.

[MP71] Robert McNaughton and Seymour Papert. *Counter-Free Automata*. The MIT Press, Cambridge, Mass., 1971.

[MS01] Victor Mitrana and Ralf Stiebe. Extended finite automata over groups. *Discrete Appl. Math.*, 108(3):287–300, 2001.

[MTV10] Pierre McKenzie, Michael Thomas, and Heribert Vollmer. Extensional uniformity for boolean circuits. *SIAM J. Comput.*, 39(7):3186–3206, 2010.

[Niv56] Ivan Niven. *Irrational Numbers*. Carus Mathematical Monographs. Mathematical Association of America, 1956.

[Niv63] Ivan Niven. *Diophantine Approximations*. Dover Publications, 1963.

[Par66] Rohit J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.

[Per95] Dominique Perrin. Les débuts de la théorie des automates. *Technique et science informatiques*, 14(4):409–433, 1995.

[Per03] Dominique Perrin. Automi e linguaggi formali. In Sandro Petruccioli, editor, *Storia della Scienza*, volume IX, pages 197–205. Istituto della Enciclopedia Italiana, 2003.

[Pin03] Jean-Éric Pin. Algebraic tools for the concatenation product. *Theoretical Computer Science*, 292:317–342, 2003.

[Pin11] Jean-Éric Pin. Theme and variations on the concatenation product. In *Algebraic Informatics*, volume 6742 of *Lecture Notes in Computer Science*, pages 44–64. Springer, 2011.

[Pre27] Mojzesz Presburger. Über de vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchen, die addition als einzige operation hervortritt. In *Comptes Rendus du Premier Congrès des Mathématiciens des Pays Slaves*, pages 92–101, Warsaw, 1927.

[Res75] Antonio Restivo. A characterization of bounded regular sets. In *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 239–244. Springer, 1975.

[Sak76] Jacques Sakarovitch. An algebraic framework for the study of the syntactic monoids application to the group languages. In *Mathematical Foundations of Computer Science*, volume 45 of *Lecture Notes in Computer Science*, pages 510–516. Springer, 1976.

[Sak03] Jacques Sakarovitch. *Élements de théorie des automates*. Vuibert, 2003.

[Sch65] Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[Sim93] Imre Simon. The product of rational languages. In *Automata, Languages and Programming*, volume 700 of *Lecture Notes in Computer Science*, pages 430–444. Springer-Verlag, 1993.

[Sip97] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Co., Boston, Massachusetts, 1997.

[SSM03] Helmut Seidl, Thomas Schwentick, and Anca Muscholl. Numerical document queries. In *Principles of Database Systems*, pages 155–166, San Diego, CA, USA, 2003. ACM Press.

[Str79] Howard Straubing. Families of recognizable sets corresponding to certain varieties of finite monoids. *J. Pure Appl. Algebra*, 15(3):305–318, 1979.

[Str94] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, 1994.

[TT07] Pascal Tesson and Denis Thérien. Logic meets algebra: the case of regular languages. *Logical Methods in Computer Science*, 3(1), 2007.

[vdD98] Lou van den Dries. *Tame Topology and O-minimal Structures*. Cambridge Univ. Press, 1998.

[WB95] Pierre Wolper and Bernard Boigelot. An automata-theoretic approach to Presburger arithmetic constraints. In *Static Analysis*, volume 983 of *Lecture Notes in Computer Science*, pages 21–32. Springer Berlin / Heidelberg, 1995.

Index

Bold page numbers indicate the definition of the term or its main reference.