LNCS 6742

Franz Winkler (Ed.)

Algebrai
Informat

4th International Conferen
Linz, Austria, June 2011
Proceedings

Franz Winkler (Ed.)

Algebraic Informatics

4th International Conference, CAI 2011
Linz, Austria, June 21-24, 2011
Proceedings

Springer

Volume Editor

Franz Winkler
Johannes Kepler University
Research Institute for Symbolic Computation (RISC)
Altenberger Straße 69, 4040, Linz, Austria
E-mail: franz.winkler@risc.jku.at

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-21492-9 e-ISBN 978-3-642-21493-6
DOI 10.1007/978-3-642-21493-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011928784

CR Subject Classification (1998): F.4, I.1.3, F.1.1, F.4.1, F.4.3, F.4.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

CAI 2011 was the 4th International Conference on Algebraic Informatics. The three previous conferences, CAI 2005, CAI 2007, and CAI 2009, were all held at the Department of Mathematics of the Aristotle University of Thessaloniki in Greece. At CAI 2009, the Steering Committee asked me to organize CAI 2011 at RISC in Linz/Hagenberg, Austria; a proposal which I gladly accepted. In fact the focus of CAI is in very close correlation to the research topics at RISC, seeing mathematics and computer science as one academic field with close internal interactions and mutual dependencies. Thus, CAI 2011 continued the tradition established by previous CAIs, namely, to bring together researchers from theoretical computer science and constructive algebra. The goal of this endeavor was to enhance the understanding of syntactic and semantic problems by algebraic models; and also to propagate the application of modern techniques from informatics in algebraic computation. CAI 2011 tried to achieve these goals via invited lectures, tutorials, and contributed research talks.

As stated in the call for papers and on the homepage of CAI 2011, the topics of interest included algebraic semantics, formal power series, syntactic objects, algebraic picture processing, finite and infinite computations, acceptors and transducers for discrete structures, decision problems, algebraic characterization of logical theories, process algebra, algebraic algorithms, algebraic coding theory, algebraic aspects of cryptography, term rewriting, algebraic aspects of number theory.

The program of CAI 2011 consisted of 4 invited lectures and 13 contributed talks. With 1 exception, all these presentations are reflected in the proceedings. Additionally 2 tutorials were given: by A. Middeldorp and F. Neurauter on "Termination and Complexity of Rewrite Systems," and by P. Padawitz on "Co/Algebraic Modelling and Verification at Work." Unfortunatey, during the preparation phase of CAI 2011 one of our leading colleagues and designated member of the Program Committee, Stephen L. Bloom, passed away. In the first lecture at the opening of CAI 2011, Z. Ésik, member of the Steering Committee, recalled the life and academic achievements of Stephen L. Bloom.

I am grateful to a great number of colleagues for making CAI 2011 a successful event: the members of the Steering Committee, the colleagues in the Program Committee, my co-workers in the Local Committee, and also Alfred Hofmann and his team at Springer LNCS.

June 2011 Franz Winkler

Organization

CAI 2011 was organized by the Research Institute for Symbolic Computation, Johannes Kepler University Linz.

Steering Committee

Jean Berstel, Marne-la-Vallée
Symeon Bozapalidis, Thessaloniki
Zoltán Ésik, Szeged
Werner Kuich, Vienna
Arto Salomaa, Turku

Program Committee

Erhard Aichinger, Linz
Armin Biere, Linz
Symeon Bozapalidis, Thessaloniki
Bruno Courcelle, Bordeaux
Wan Fokkink, Amsterdam
Zoltán Fülöp, Szeged
Ulrike Golas, Berlin
Kevin Hammond, St. Andrews
Štěpăn Holub, Prague
Jan Willem Klop, Amsterdam
Barbara König, Duisburg
Laura Kovács, Vienna
Salvador Lucas, Valencia
Damian Niwinski, Warsaw
Attila Pethő, Debrecen
George Rahonis, Thessaloniki
Simona Ronchi Della Rocca, Turin
Davide Sangiorgi, Bologna
Wolfgang Schreiner, Linz
Mikhail Volkov, Ekaterinburg
Franz Winkler, Linz (Chair)

Referees

Organizing Committee

Sponsors

Bundesministerium für Wissenschaft und Forschung (BMWF)
Johannes Kepler Universität Linz (JKU)
Linzer Hochschulfonds (LHF)
Land Oberösterreich

Table of Contents

Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems*

Aart Middeldorp[1], Georg Moser[1], Friedrich Neurauter[1], Johannes Waldmann[2], and Harald Zankl[1]

[1] Institute of Computer Science, University of Innsbruck, Austria
[2] Fakultät Informatik, Mathematik und Naturwissenschaften, Hochschule für Technik, Wirtschaft und Kultur Leipzig, Germany

Abstract. Matrix interpretations can be used to bound the derivational complexity of term rewrite systems. In particular, triangular matrix interpretations over the natural numbers are known to induce polynomial upper bounds on the derivational complexity of (compatible) rewrite systems. Recently two different improvements were proposed, based on the theory of weighted automata and linear algebra. In this paper we strengthen and unify these improvements by using joint spectral radius theory.

Keywords: derivational complexity, matrix interpretations, weighted automata, joint spectral radius.

1 Introduction

This paper is concerned with automated complexity analysis of term rewrite systems. Given a terminating rewrite system, the aim is to obtain information about the maximal length of rewrite sequences in terms of the size of the initial term. This is known as derivational complexity. Developing methods for bounding the derivational complexity of rewrite systems has become an active and competitive[1] research area in the past few years (e.g. [6, 11–15, 19, 21]).

Matrix interpretations [4] are a popular method for automatically proving termination of rewrite systems. They can readily be used to establish upper bounds on the derivational complexity of compatible rewrite systems. However, in general, matrix interpretations induce exponential (rather than polynomial) upper bounds. In order to obtain polynomial upper bounds, the matrices used in a matrix interpretation must satisfy certain (additional) restrictions, the study of which is the central concern of [14, 15, 19].

So what are the conditions for polynomial boundedness of a matrix interpretation? In the literature, two different approaches have emerged. On the one hand, there is the automata-based approach of [19], where matrices are viewed as

* This research is supported by FWF (Austrian Science Fund) project P20133. Friedrich Neurauter is supported by a grant of the University of Innsbruck.

[1] `http://www.termination-portal.org/wiki/Complexity`

weighted (word) automata computing a weight function, which is required to be polynomially bounded. The result is a complete characterization (i.e., necessary and sufficient conditions) of polynomially bounded matrix interpretations over $\mathbb{N}$. On the other hand, there is the algebraic approach pursued in [15] (originating from [14]) that can handle matrix interpretations over $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$ but only provides sufficient conditions for polynomial boundedness. In what follows, we shall see, however, that these two seemingly different approaches can be unified and strengthened with the help of joint spectral radius theory [9, 10], a branch of mathematics dedicated to studying the growth rate of products of matrices taken from a set.

The remainder of this paper is organized as follows. In the next section we recall preliminaries from linear algebra and term rewriting. We give a brief account of the matrix method for proving termination of rewrite systems. In Section 3 we introduce the algebraic approach for characterizing the polynomial growth of matrix interpretations. We improve upon the results of [15] by considering the minimal polynomial associated with the component-wise maximum matrix of the interpretation. We further show that the joint spectral radius of the matrices in the interpretation provides a better characterization of polynomial growth and provide conditions for the decidability of the latter. Section 4 is devoted to automata-based methods for characterizing the polynomial growth of matrix interpretations. We revisit the characterization results of [19] and provide precise complexity statements. In Section 5 we unify the two approaches and show that, in theory at least, the joint spectral radius theory approach subsumes the automata-based approach. Automation of the results presented in earlier sections is the topic of Section 6. To this end we extend the results from [15, 19]. We also provide experimental results. We conclude with suggestions for future research in Section 7.

2 Preliminaries

As usual, we denote by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ the sets of natural, integer, rational and real numbers. Given $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ and $m \in D$, $>_D$ denotes the standard order of the respective domain and D_m abbreviates $\{x \in D \mid x \geqslant m\}$.

Linear Algebra: Let R be a ring (e.g., $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$). The ring of all *n-dimensional square matrices* over R is denoted by $R^{n \times n}$ and the *polynomial ring* in *n indeterminates* $x_1, \ldots, x_n$ by $R[x_1, \ldots, x_n]$. In the special case $n = 1$, any polynomial $p \in R[x]$ can be written as $p(x) = \sum_{k=0}^{d} a_k x^k$ for some $d \in \mathbb{N}$. For the largest k such that $a_k \neq 0$, we call $a_k x^k$ the *leading term* of p, a_k its *leading coefficient* and k its *degree*. The polynomial p is said to be *monic* if its leading coefficient is one. It is said to be *linear*, *quadratic*, *cubic* if its degree is one, two, three.

In case R is equipped with a partial order $\geqslant$, the component-wise extension of this order to $R^{n \times n}$ is also denoted as $\geqslant$. We say that a matrix A is *non-negative* if $A \geqslant 0$ and denote the set of all non-negative n-dimensional square matrices of $\mathbb{Z}^{n \times n}$ by $\mathbb{N}^{n \times n}$. The $n \times n$ *identity matrix* is denoted by I_n and the $n \times n$ *zero matrix* is denoted by 0_n. We simply write I and 0 if n is clear from the context.

The *characteristic polynomial* $\chi_A(\lambda)$ of a square matrix $A \in R^{n \times n}$ is defined as $\det(\lambda I_n - A)$, where det denotes the determinant. It is monic and its degree is n. The equation $\chi_A(\lambda) = 0$ is called the *characteristic equation* of A. The solutions of this equation, i.e., the *roots* of $\chi_A(\lambda)$, are precisely the *eigenvalues* of A, and the *spectral radius* $\rho(A)$ of A is the maximum of the absolute values of all eigenvalues. A non-zero vector x is an *eigenvector* of A if $Ax = \lambda x$ for some eigenvalue λ of A. We say that a polynomial $p \in R[x]$ *annihilates* A if $p(A) = 0$. The Cayley-Hamilton theorem [16] states that A satisfies its own characteristic equation, i.e., χ_A annihilates A. The unique monic polynomial of minimum degree that annihilates A is called the *minimal polynomial* $m_A(x)$ of A. The *multiplicity* of a root λ of $p \in R[x]$ is denoted by $\#p(\lambda)$.

With any matrix $A \in R^{n \times n}$ we associate a directed (weighted) graph $G(A)$ on n vertices numbered from 1 to n such that there is a directed edge (of weight A_{ij}) in $G(A)$ from i to j if and only if $A_{ij} \neq 0$. In this situation, A is said to be the *adjacency matrix* of the graph $G(A)$. The *weight of a path* in $G(A)$ is the product of the weights of its edges.

With a finite set of matrices $S \subseteq R^{n \times n}$ we associate the directed (weighted) graph $G(S) := G(M)$, where M denotes the component-wise maximum of the matrices in S, i.e., $M_{ij} = \max\{A_{ij} \mid A \in S \text{ and } 1 \leqslant i, j \leqslant n\}$. Following [10], we define a directed graph $G^k(S)$ for $k \geqslant 2$ on n^k vertices representing ordered tuples of vertices of $G(S)$, such that there is an edge from vertex $(i_1, \ldots, i_k)$ to $(j_1, \ldots, j_k)$ if and only if there is a matrix $A \in S$ with $A_{i_\ell j_\ell} > 0$ for all $\ell = 1, \ldots, k$. This is akin to the k-fold Kronecker product of the matrix A.

For functions $f, g \colon \mathbb{N} \to \mathbb{N}$ we write $f(n) = O(g(n))$ if there are constants $M, N \in \mathbb{N}$ such that $f(n) \leqslant M \cdot g(n) + N$ for all $n \in \mathbb{N}$. Furthermore, $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

Rewriting: We assume familiarity with the basics of term rewriting [1, 18]. Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a fixed-arity signature. The set of *terms* over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The *size* $|t|$ of a term t is defined as the number of function symbols and variables occurring in it. The set of *positions* $\mathcal{P}\mathsf{os}(t)$ of a term t is defined as usual. Positions are denoted as sequences of natural numbers. A *term rewrite system* (*TRS* for short) $\mathcal{R}$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is a *finite* set of rewrite rules $\ell \to r$ such that $\ell \notin \mathcal{V}$ and $\mathcal{V}\mathsf{ar}(\ell) \supseteq \mathcal{V}\mathsf{ar}(r)$. The smallest rewrite relation that contains $\mathcal{R}$ is denoted by $\to_{\mathcal{R}}$. The transitive (and reflexive) closure of $\to_{\mathcal{R}}$ is denoted by $\to_{\mathcal{R}}^{+}$ ($\to_{\mathcal{R}}^{*}$). Let s and t be terms. If exactly n steps are performed to rewrite s to t, we write $s \to^n t$. The *derivation height* of a term s with respect to a well-founded and finitely branching relation $\to$ is defined as $\mathsf{dh}(s, \to) = \max\{n \mid s \to^n t \text{ for some term } t\}$. The *derivational complexity function* of $\mathcal{R}$ is defined as: $\mathsf{dc}_{\mathcal{R}}(k) = \max\{\mathsf{dh}(t, \to_{\mathcal{R}}) \mid |t| \leqslant k\}$.

Matrix Interpretations: An $\mathcal{F}$-*algebra* $\mathcal{A}$ consists of a carrier set A and a collection of interpretations $f_{\mathcal{A}} \colon A^k \to A$ for each k-ary function symbol in $\mathcal{F}$. By $[\alpha]_{\mathcal{A}}(\cdot)$ we denote the usual evaluation function of $\mathcal{A}$ according to an assignment α which maps variables to values in A. An $\mathcal{F}$-algebra together with a well-founded order $>$ on A is called a *monotone algebra* if every function symbol

$f \in \mathcal{F}$ is monotone with respect to $>$ in all arguments. Any monotone algebra $(\mathcal{A}, >)$ (or just $\mathcal{A}$ if $>$ is clear from the context) induces a well-founded order on terms: $s >_{\mathcal{A}} t$ if and only if $[\alpha]_{\mathcal{A}}(s) > [\alpha]_{\mathcal{A}}(t)$ for all assignments α. A TRS $\mathcal{R}$ and a monotone algebra $\mathcal{A}$ are compatible if $\ell >_{\mathcal{A}} r$ for all $\ell \to r \in \mathcal{R}$.

For matrix interpretations, we fix a dimension $n \in \mathbb{N} \setminus \{0\}$ and use the set $\mathbb{R}_0^n$ as the carrier of an algebra $\mathcal{M}$, together with the order $>_\delta$ on $\mathbb{R}_0^n$ defined as $(x_1, x_2, \ldots, x_n)^{\mathrm{T}} >_\delta (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ if $x_1 >_{\mathbb{R},\delta} y_1$ and $x_i \geqslant_{\mathbb{R}} y_i$ for $2 \leqslant i \leqslant n$. Here $x >_{\mathbb{R},\delta} y$ if and only if $x \geqslant_{\mathbb{R}} y + \delta$. Each k-ary function symbol f is interpreted as a linear function of the following shape: $f_{\mathcal{M}}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k) = F_1\boldsymbol{v}_1 + \cdots + F_k\boldsymbol{v}_k + \boldsymbol{f}$ where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are (column) vectors of variables, $F_1, \ldots, F_k \in \mathbb{R}_0^{n \times n}$ and $\boldsymbol{f}$ is a vector in $\mathbb{R}_0^n$. The $F_1, \ldots, F_k$ are called the *matrices* of the interpretation $f_{\mathcal{M}}$, while $\boldsymbol{f}$ is called the *absolute vector* of $f_{\mathcal{M}}$. We write $\mathsf{abs}(f)$ for the $\boldsymbol{f}$. To ensure monotonicity, it suffices that the top left entry $(F_i)_{11}$ of every matrix F_i is at least one. Then it is easy to see that $(\mathcal{M}, >_\delta)$ forms a monotone algebra for any $\delta > 0$. We obtain matrix interpretations over $\mathbb{Q}$ by restricting to the carrier $\mathbb{Q}_0^n$. Similarly, matrix interpretations over $\mathbb{N}$ operate on the carrier $\mathbb{N}^n$ and use $\delta = 1$.

Let α_0 denote the assignment that maps every variable to $\mathbf{0}$. Let t be a term. In the following we abbreviate $[\alpha_0]_{\mathcal{M}}(t)$ to $[t]_{\mathcal{M}}$ (or $[t]$ if $\mathcal{M}$ can be inferred from the context) and we write $[t]_j$ $(1 \leqslant j \leqslant n)$ for the j-th element of $[t]$.

Let $\mathcal{M}$ be a matrix interpretation of dimension n. Let S be the set of matrices occurring in $\mathcal{M}$ and let $S^k = \{A_1 \cdots A_k \mid A_i \in S, 1 \leqslant i \leqslant k\}$ be the set of all products of length k of matrices taken from S. Here S^0 consists of the identity matrix. Further, S^* denotes the (matrix) monoid generated by S, i.e., $S^* = \bigcup_{k=0}^{\infty} S^k$.

3 Algebraic Methods for Bounding Polynomial Growth

In this section we study an algebraic approach to characterize polynomial growth of matrix interpretations (over $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$). We employ the following definition implicitly used in [14, 15].

Definition 1. *Let $\mathcal{M}$ be a matrix interpretation. We say that $\mathcal{M}$ is* polynomially bounded (with degree d) *if the growth of the entries of all matrix products in S^* is polynomial (with degree d) in the length of such products.*

The relationship between polynomially bounded matrix interpretations and the derivational complexity of compatible TRSs is as follows (cf. [15]).

Lemma 2. *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible matrix interpretation. If $\mathcal{M}$ is polynomially bounded with degree d then $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$.* □

3.1 Spectral Radius

In this subsection we over-approximate the growth of entries of matrix products of the form $A_1 \cdots A_k \in S^k$ by M^k where $M_{ij} = \max\{A_{ij} \mid A \in S\}$ for all $1 \leqslant i, j \leqslant n$. Then, by non-negativity of the matrices in S, we have

$$(A_1 \cdots A_k)_{ij} \leqslant (M^k)_{ij} \quad \text{for all } 1 \leqslant i, j \leqslant n \tag{1}$$

Thus, polynomial boundedness of the entries of $A_1 \cdots A_k$ follows from polynomial boundedness of the entries of M^k. In [15], the latter is completely characterized by the spectral radius $\rho(M)$ of M being at most one. Here we build on this result (and its proof) but establish a lemma that allows to obtain a tight bound for the degree of polynomial growth of the entries of M^k.

Lemma 3. *Let $M \in \mathbb{R}_0^{n\times n}$ and let $p \in \mathbb{R}[x]$ be a monic polynomial that annihilates M. Then $\rho(M) \leqslant 1$ if and only if all entries of M^k ($k \in \mathbb{N}$) are asymptotically bounded by a polynomial in k of degree d, where $d := \max_\lambda(0, \#p(\lambda) - 1)$ and λ are the roots of p with absolute value exactly one and multiplicity $\#p(\lambda)$.*

Proof. Straightforward adaptation of the proof of [15, Lemma 4]. □

Based on Lemma 3 (with $p(x) = \chi_M(x)$), one obtains the following theorem concerning complexity analysis via matrix interpretations.

Theorem 4 ([15, Theorem 6]). *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible matrix interpretation of dimension n. Further, let M denote the component-wise maximum of all matrices occurring in $\mathcal{M}$. If the spectral radius of M is at most one, then $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$, where $d := \max_\lambda(0, \#\chi_M(\lambda) - 1)$ and λ are the eigenvalues of M with absolute value exactly one.* □

Obviously, the set of (monic) polynomials that annihilate a matrix M is infinite. However, from linear algebra we know that this set is generated by a unique monic polynomial of minimum degree that annihilates M, namely, the *minimal polynomial* $m_M(x)$ of M. That is, if $p(x)$ is any polynomial such that $p(M) = 0$ then $m_M(x)$ divides $p(x)$. In particular, $m_M(x)$ divides the characteristic polynomial $\chi_M(x)$. Moreover, $m_M(\lambda) = 0$ if and only if λ is an eigenvalue of M, so every root of $m_M(x)$ is a root of $\chi_M(x)$. However, in case $m_M(x) \neq \chi_M(x)$, the multiplicity of a root in $m_M(x)$ may be lower than its multiplicity in $\chi_M(x)$ (cf. [8]). In light of these facts, we conclude that in Lemma 3 one should use the minimal polynomial $m_M(x)$ rather than the characteristic polynomial $\chi_M(x)$. Then the corresponding analogon of Theorem 4 is as follows.

Theorem 5. *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible matrix interpretation of dimension n. Further, let M denote the component-wise maximum of all matrices occurring in $\mathcal{M}$. If the spectral radius of M is at most one then $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$, where $d := \max_\lambda(0, \#m_M(\lambda) - 1)$ and λ are the eigenvalues of M with absolute value exactly one.* □

Next we illustrate the usefulness of Theorem 5 on an example.

Example 6. Consider the TRS $\mathcal{R}$ consisting of the following two rewrite rules:[2]

$$\begin{aligned} \mathsf{h}(x, \mathsf{c}(y, z)) &\to \mathsf{h}(\mathsf{c}(\mathsf{s}(y), x), z) \\ \mathsf{h}(\mathsf{c}(\mathsf{s}(x), \mathsf{c}(\mathsf{s}(0), y)), z) &\to \mathsf{h}(y, \mathsf{c}(\mathsf{s}(0), \mathsf{c}(x, z))) \end{aligned} \qquad M = \begin{pmatrix} 1&1&1&1 \\ 0&0&0&0 \\ 0&1&1&0 \\ 0&1&0&1 \end{pmatrix}$$

[2] TPDB problem TRS/Endrullis_06/direct.xml

There is a compatible matrix interpretation such that the component-wise maximum matrix M (given above) has characteristic polynomial $\chi_M(x) = x(x-1)^3$ and minimal polynomial $m_M(x) = x(x-1)^2$. Thus, the upper bound for the derivational complexity of $\mathcal{R}$ derived from Theorem 4 is cubic, whereas the bound obtained by Theorem 5 is quadratic.

Next we present an example that demonstrates the conceptual limitations arising from the over-approximation of matrix products by the powers of the corresponding maximum matrix.

Example 7. Consider the TRS $\mathcal{R}$ consisting of the rules

$$\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{g}(\mathsf{f}(x))) \qquad \mathsf{g}(\mathsf{g}(x)) \to x \qquad \mathsf{b}(x) \to x$$

Lemma 8. *There is a matrix interpretation compatible with $\mathcal{R}$ that is polynomially bounded (in the sense of Definition 1), but there is no matrix interpretation compatible with $\mathcal{R}$ where all entries in the component-wise maximum matrix are polynomially bounded.*

Proof. For the first item consider the following matrix interpretation $\mathcal{M}$ establishing linear derivational complexity of $\mathcal{R}$ (cf. Theorem 16) from Example 7:

$$\mathsf{f}_{\mathcal{M}}(\boldsymbol{x}) = \begin{pmatrix}1&1&0\\0&0&0\\0&0&0\end{pmatrix}\boldsymbol{x} + \begin{pmatrix}0\\4\\0\end{pmatrix} \quad \mathsf{g}_{\mathcal{M}}(\boldsymbol{x}) = \begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}\boldsymbol{x} + \begin{pmatrix}1\\0\\3\end{pmatrix} \quad \mathsf{b}_{\mathcal{M}}(\boldsymbol{x}) = \boldsymbol{x} + \begin{pmatrix}4\\4\\0\end{pmatrix}$$

For the second item assume a compatible matrix interpretation $\mathcal{M}$ of dimension n with $\mathsf{b}_{\mathcal{M}}(\boldsymbol{x}) = B\boldsymbol{x} + b$, $\mathsf{f}_{\mathcal{M}}(\boldsymbol{x}) = F\boldsymbol{x} + f$ and $\mathsf{g}_{\mathcal{M}}(\boldsymbol{x}) = G\boldsymbol{x} + g$. Assume $G \geqslant I_n$. To orient the first rule, the constraint $Ff > FGf$ must be satisfied, but by (weak) monotonicity of matrix multiplication we obtain the contradiction $Ff > FGf \geqslant FI_nf = Ff$ and hence $G \ngeqslant I_n$, i.e., there exists an index l such that $G_{ll} < 1$. Since $GG \geqslant I_n$ is needed to orient the second rule, we have $\sum_j G_{lj}G_{jl} \geqslant 1$ and consequently $\sum_{j\neq l} G_{lj}G_{jl} > 0$. The third rule demands $B \geqslant I_n$ and for the maximum matrix M we have $M \geqslant \max(I_n, G)$. From $(I_n)_{ll} \geqslant 1$ and $\sum_{j\neq l} G_{lj}G_{jl} > 0$ we conclude $(M^2)_{ll} > 1$, which gives rise to exponential growth of $(M^k)_{ll}$ since all entries in M are non-negative. □

3.2 Joint Spectral Radius

Instead of using a single maximum matrix to over-approximate the growth of finite matrix products taken from a set of matrices S, in this subsection we provide a concise analysis using joint spectral radius theory [9, 10]. In particular, we shall see that the joint spectral radius of S completely characterizes polynomial growth of all such products, just like the spectral radius of a single matrix characterizes polynomial growth of the powers of this matrix (cf. Lemma 3).

Definition 9. *Let $S \subseteq \mathbb{R}^{n \times n}$ be a finite set of real square matrices, and let $\|\cdot\|$ denote a matrix norm. The growth function* growth *associated with S is defined as follows:*

$$\mathsf{growth}_S(k, \|\cdot\|) := \max\{\, \|A_1 \cdots A_k\| \mid A_i \in S, 1 \leqslant i \leqslant k \,\}$$

The asymptotic behaviour of $\mathsf{growth}_S(k, \|\cdot\|)$ can be characterized by the joint spectral radius of S.

Definition 10. *Let $S \subseteq \mathbb{R}^{n \times n}$ be finite, and let $\|\cdot\|$ denote a matrix norm. The* joint spectral radius $\rho(S)$ *of S is defined by the limit*

$$\rho(S) := \lim_{k \to \infty} \max\{\, \|A_1 \cdots A_k\|^{1/k} \mid A_i \in S, 1 \leqslant i \leqslant k \,\}$$

It is well-known that this limit always exists and that it does not depend on the chosen norm, which follows from the equivalence of all norms in $\mathbb{R}^n$; e.g., one could take the norm given by the sum of the absolute values of all matrix entries. Further, if $S = \{A\}$ is a singleton set, the joint spectral radius $\rho(S)$ of S and the spectral radius $\rho(A)$ of A coincide:

$$\rho(S) = \lim_{k \to \infty} \|A^k\|^{1/k} = \max\{\, |\lambda| \mid \lambda \text{ is an eigenvalue of } A \,\} = \rho(A)$$

Since Definition 10 is independent of the actual norm, from now on we simply write $\mathsf{growth}_S(k)$. The following theorem (due to [2]) provides a characterization of polynomial boundedness of $\mathsf{growth}_S(k)$ by the joint spectral radius of S.

Theorem 11 **([2, Theorem 1.2]).** *Let $S \subseteq \mathbb{R}^{n \times n}$ be a finite set of matrices. Then $\mathsf{growth}_S(k) = O(k^d)$ for some $d \in \mathbb{N}$ if and only if $\rho(S) \leqslant 1$. In particular, $d \leqslant n - 1$.* □

Hence, polynomial boundedness of $\mathsf{growth}_S(k)$ is decidable if $\rho(S) \leqslant 1$ is decidable. But it is well-known that in general the latter is undecidable for arbitrary, but finite sets $S \subseteq \mathbb{R}^{n \times n}$, even if S contains only non-negative rational (real) matrices (cf. [10, Theorem 2.6]). However, if S contains only non-negative integer matrices then $\rho(S) \leqslant 1$ is decidable. In particular, there exists a polynomial-time algorithm that decides it (cf. [10, Theorem 3.1]). This algorithm is based on the following lemma.

Lemma 12 **([10, Lemma 3.3]).** *Let $S \subseteq \mathbb{R}_0^{n \times n}$ be a finite set of non-negative, real square matrices. Then there is a product $A \in S^*$ such that $A_{ii} > 1$ for some $i \in \{1, \ldots, n\}$ if and only if $\rho(S) > 1$.* □

According to [10], for $S \subseteq \mathbb{N}^{n \times n}$, the existence of such a product can be characterized in terms of the graphs $G(S)$ and $G^2(S)$ one can associate with S. More precisely, there is a product $A \in S^*$ with $A_{ii} > 1$ if and only if

1. there is a cycle in $G(S)$ containing at least one edge of weight $w > 1$, or
2. there is a cycle in $G^2(S)$ containing at least one vertex (i, i) and at least one vertex (p, q) with $p \neq q$.

Hence, we have $\rho(S) \leqslant 1$ if and only if neither of the two conditions holds, which can be checked in polynomial time according to [10]. Furthermore, as already mentioned in [10, Chapter 3], this graph-theoretic characterization does not only hold for non-negative integer matrices, but for any set of matrices such that all matrix entries are either zero or at least one (because then all paths in $G(S)$ have weight at least one).

Lemma 13. *Let $S \subseteq \mathbb{R}_0^{n \times n}$ be a finite set of matrices such that all matrix entries are either zero or at least one. Then $\rho(S) \leqslant 1$ is decidable in polynomial time.* □

So, in the situation of Lemma 13, polynomial boundedness of $\mathsf{growth}_S(k)$ is decidable in polynomial time. In addition, the exact degree of growth can be computed in polynomial time (cf. [10, Theorem 3.3]).

Theorem 14. *Let $S \subseteq \mathbb{R}_0^{n \times n}$ be a finite set of matrices such that $\rho(S) \leqslant 1$ and all matrix entries are either zero or at least one. Then*

$$\mathsf{growth}_S(k) = \Theta(k^d)$$

where the growth rate *d is the largest integer possessing the following property: there exist d different pairs of indices $(i_1, j_1), \ldots, (i_d, j_d)$ such that for every pair (i_s, j_s) the indices i_s, j_s are different and there is a product $A \in S^*$ for which $A_{i_s i_s}, A_{i_s j_s}, A_{j_s j_s} \geqslant 1$, and for each $1 \leqslant s \leqslant d-1$, there exists $B \in S^*$ with $B_{j_s i_{s+1}} \geqslant 1$. Moreover, d is computable in polynomial time.* □

Next we elaborate on the ramifications of joint spectral radius theory on complexity analysis of TRSs via polynomially bounded matrix interpretations. To begin with, we observe that the characterization of polynomially bounded matrix interpretations given in Definition 1 can be rephrased as follows: A matrix interpretation $\mathcal{M}$ is polynomially bounded if the joint spectral radius of the set of matrices occurring in $\mathcal{M}$ is at most one. This follows directly from Theorem 11 (using as $\|\cdot\|$ in $\mathsf{growth}_S(k, \|\cdot\|)$ the matrix norm given by the sum of the absolute values of all matrix entries). Due to the relationship between polynomially bounded matrix interpretations and the derivational complexity of compatible TRSs expressed in Lemma 2, we immediately obtain the following theorem, which holds for matrix interpretations over $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$.

Theorem 15. *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible matrix interpretation of dimension n. Further, let $S \subseteq \mathbb{R}_0^{n \times n}$ denote the set of all matrices occurring in $\mathcal{M}$. If the joint spectral radius of S is at most one then $\mathsf{dc}_{\mathcal{R}}(k) = O(k^n)$.* □

As this theorem assumes the worst-case growth rate for $\mathsf{growth}_S(k)$, the inferred degree of the polynomial bound may generally be too high (and unnecessarily so). Yet with the help of Theorem 14, from which we obtain the exact growth rate of $\mathsf{growth}_S(k)$, Theorem 15 can be strengthened, at the expense of having to restrict the set of permissible matrices.

Theorem 16. *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible matrix interpretation of dimension n. Further, let $S \subseteq \mathbb{R}_0^{n \times n}$ denote the set of all matrices occurring in $\mathcal{M}$ and assume that all matrix entries are either zero or at least one. If the joint spectral radius of S is at most one then $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$, where d refers to the growth rate obtained from Theorem 14.* □

4 Automata Methods for Bounding Polynomial Growth

In this section we study automata-based methods to classify polynomial growth of matrix interpretations. Complementing Definition 1, we employ the following definition of polynomial growth [19].

Definition 17. *The growth function of a matrix interpretation $\mathcal{M}$ is defined as $\mathsf{growth}_{\mathcal{M}}(k) := \max\{[t]_1 \mid t \text{ is a term and } |t| \leqslant k\}$. We say that $\mathcal{M}$ is* polynomially bounded with degree *d if $\mathsf{growth}_{\mathcal{M}}(k) = O(k^d)$ for some $d \in \mathbb{N}$.*

We recall basic notions of automata theory (cf. [3, 17]). A *weighted automaton (over $\mathbb{R}$)* is a quintuple $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ where Q is a finite set of states, Σ a finite alphabet, and the mappings $\lambda \colon Q \to \mathbb{R}$, $\gamma \colon Q \to \mathbb{R}$ are weight functions for entering and leaving a state. The transition function $\mu \colon \Sigma \to \mathbb{R}^{|Q| \times |Q|}$ associates with any letter in Σ a $(|Q| \times |Q|)$-matrix over $\mathbb{R}$. For $a \in \Sigma$, $\mu(a)_{pq}$ denotes the *weight* of the transition $p \xrightarrow{a} q$. We often view λ and γ as row and column vectors, respectively. We also write $\mathsf{weight}(p, a, q)$ for $\mu(a)_{pq}$ and extend the transition function μ homomorphically to words. The *weight of* $x \in \Sigma^*$, denoted by $\mathsf{weight}_{\mathcal{A}}(x)$, is the sum of the weights of paths in $\mathcal{A}$ labeled with x. We define

$$\mathsf{weight}_{\mathcal{A}}(x) = \sum_{p,q \in Q} \lambda(p) \cdot \mu(x)_{pq} \cdot \gamma(q) = \lambda \cdot \mu(x) \cdot \gamma$$

where the weight of the empty word is just $\lambda \cdot \gamma$.

A weighted automaton $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ is called *normal* if the row vector λ is $(1, \overline{0})$ and the column vector γ contains only entries with weight 0 or 1. Here $\overline{0}$ denotes a sequence of $|Q| - 1$ zeros. Let $\mathcal{A}$ be a normal weighted automaton. The unique state q_0 such that $\lambda(q_0) \neq 0$ is called *initial* and the states in $F := \{q \mid \gamma(q) \neq 0\}$ are called *final.* We call a state $p \in Q$ *useful* if there exists a path in $\mathcal{A}$ from q_0 to $q \in F$ that contains p. An automaton $\mathcal{A}$ is *trim* if all states are useful. In the sequel all considered automata will be normal.

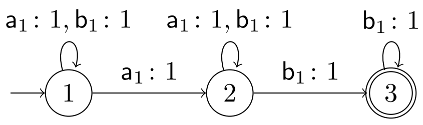
The main result of this section is a complete (and polytime decidable) characterization of polynomial growth of matrix interpretations over $\mathbb{N}$, thus re-stating the main result in [19]. We extend upon [19] by clarifying the polytime decidability of the properties involved. Given a matrix interpretation $\mathcal{M}$, we denote by $C^{\mathcal{M}}$ the component-wise maximum of all absolute vectors in $\mathcal{M}$.

Definition 18. *With every n-dimensional matrix interpretation $\mathcal{M}$ for a signature $\mathcal{F}$ we associate a weighted automaton $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ as follows: $Q = \{1, \ldots, n\}$, $\Sigma = \{f_i \mid f \in \mathcal{F} \text{ has arity } k \text{ and } 1 \leqslant i \leqslant k\}$, $\lambda = (1, \overline{0})$, $\gamma \in \{0,1\}^n$ such that $\gamma(i) = 1$ if and only if $C_i^{\mathcal{M}} > 0$, and $\mu(f_i) = F_i$ where F_i denotes the i-th matrix of $f_{\mathcal{M}}$.*

Example 19. Consider the matrix interpretation $\mathcal{M}$ of dimension 3 with

$$\mathsf{a}_{\mathcal{M}}(\boldsymbol{x}) = \begin{pmatrix}1&1&0\\0&1&0\\0&0&0\end{pmatrix}\boldsymbol{x} \qquad \mathsf{b}_{\mathcal{M}}(\boldsymbol{x}) = \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\boldsymbol{x} + \begin{pmatrix}0\\0\\1\end{pmatrix}$$

The following automaton $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ corresponds to $\mathcal{M}$:


$\mathsf{a}_1 : 1, \mathsf{b}_1 : 1$
$\mathsf{a}_1 : 1, \mathsf{b}_1 : 1$
$\mathsf{b}_1 : 1$
$\mathsf{a}_1 : 1$
$\mathsf{b}_1 : 1$
1
2
3

As $\mathcal{A}$ is normal, we represent λ and γ by indicating the input and output states as usual for finite automata.

In analogy to Definition 17 we define the growth function of a weighted automaton.

Definition 20. *Let* $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ *be a weighted automaton. The growth function of* $\mathcal{A}$ *is defined as* $\mathsf{growth}_{\mathcal{A}}(k) := \max\{\mathsf{weight}(x) \mid x \in \Sigma^k\}$.

The following theorem restates [19, Theorem 3.3] in connection with the remark immediately following the theorem.

Theorem 21. *Let* $\mathcal{M}$ *be a matrix interpretation and* $\mathcal{A}$ *the corresponding automaton. Then* $\mathsf{growth}_{\mathcal{A}}(k) = O(k^d)$ *if and only if* $\mathsf{growth}_{\mathcal{M}}(k) = O(k^{d+1})$. □

As a consequence of Theorem 21 we observe that the growth of matrix interpretations and weighted automata are polynomially related. Hence if we can decide the polynomial growth rate of automata, we can decide the polynomial growth rate of matrix interpretations. In the remainder of this section we restrict to matrix interpretations over $\mathbb{N}$ (and therefore to weighted automata over $\mathbb{N}$).

The growth rate of automata has been studied in various contexts. Here we only mention two independent developments. Weber and Seidl provide in [20] a complete characterization of the degree of growth of the ambiguity of nondeterministic finite automata (NFAs). If we restrict to weights over $\{0, 1\}$, weighted automata simplify to NFAs. Furthermore, if the degree of growth of the ambiguity of an NFA $\mathcal{A}$ is d then $\mathsf{growth}_{\mathcal{A}}(k) = O(k^d)$ and vice versa. Jungers [10] completely characterizes polynomial growth rates of matrix products in the context of joint spectral radius theory (cf. Theorem 14). Due to the closeness of weighted automata to finite automata we follow the account of Weber and Seidl here. To make full use of the results in [20] it suffices to observe that weighted automata over $\mathbb{N}$ can be represented as finite automata with parallel edges. This will also allow to place later developments (see Section 5) into context.

Consider the following criterion for a weighted automaton $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \delta)$:

$$\exists\, q \in Q\ \exists\, x \in \Sigma^* \text{ such that } \mathsf{weight}(q, x, q) \geqslant 2 \text{ and } q \text{ is useful} \qquad (\mathsf{EDA})$$

The criterion was introduced in [20] for NFAs. A similar criterion can be distilled from the decision procedures given in [10, Chapter 3]. Note that the conditions given after Lemma 12 are equivalent to EDA if we restrict our attention to trim automata. Without loss of generality let $Q = \{1, \dots, n\}$ and let S be the set of transition matrices in $\mathcal{A}$. By definition, there exists a product $A \in S^*$ with $A_{ii} \geqslant 2$ if and only if there exists $x \in \Sigma^*$ such that $\mathsf{weight}(i, x, i) \geqslant 2$. Hence $\mathcal{A}$ fulfills EDA if and only if condition 1 or condition 2 is fulfilled. In the following we sometimes refer to EDA as the collection of these conditions.

Condition EDA is sufficient to decide the polynomial growth of $\mathbb{N}$-weighted automata. The following theorem embodies Theorem 5.2 in [19]. The polytime decision procedure stems from [20].

Theorem 22. *Let $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ be a weighted automaton over $\mathbb{N}$. Then there exists $d \in \mathbb{N}$ such that $\mathsf{growth}_{\mathcal{A}}(k) = O(k^d)$ if and only if $\mathcal{A}$ does not admit EDA. Furthermore this property is decidable in time $O(|Q|^4 \cdot |\Sigma|)$.* □

To determine the degree of polynomial growth, the following criterion introduced in [20] is used:

$$\begin{gathered}\exists\, p_1, q_1, \dots, p_d, q_d \in Q \;\exists\, v_1, u_2, v_2, \dots, u_d, v_d \in \Sigma^* \text{ such that } \forall\, i \geqslant 1 \;\forall\, j \geqslant 2 \\ p_i \neq q_i,\ p_i \xrightarrow{v_i} p_i,\ p_i \xrightarrow{v_i} q_i,\ q_i \xrightarrow{v_i} q_i,\ q_{j-1} \xrightarrow{u_j} p_j,\ p_i \text{ and } q_i \text{ are useful}\end{gathered} \tag{IDA_d}$$

The criterion IDA_d can be visualized as follows:

$$\begin{tikzcd} {} \arrow[r] & p_1 \arrow[loop above, "v_1"] \arrow[r, "v_1"] & q_1 \arrow[loop above, "v_1"] \arrow[r, "u_2"] & {} & \cdots & {} \arrow[r, "u_d"] & p_d \arrow[loop above, "v_d"] \arrow[r, "v_d"] & q_d \arrow[loop above, "v_d"] \arrow[r] & {} \end{tikzcd}$$

In the same vein as for EDA, one easily sees that the condition IDA_d is closely linked (on trim automata) to the conditions stated in Theorem 14. This will be detailed in Section 5.

Example 23 (continued from Example 19). It is easy to see that $\mathcal{A}$ complies with IDA_2, where $p_1 = 1$, $q_1 = p_2 = 2$, and $q_2 = 3$. Moreover, an easy calculation gives $\mathsf{weight}_{\mathcal{A}}(\mathsf{a}^n\mathsf{b}^m) = \mathsf{weight}_{\mathcal{A}}(1, \mathsf{a}^n\mathsf{b}^m, 3) = nm$ and thus the growth rate of $\mathcal{A}$ is quadratic.

The next theorem essentially follows from a close inspection of the proof of [20, Theorem 4.2] in connection with Theorem 21.

Theorem 24. *Let $\mathcal{M}$ be a matrix interpretation of dimension n and let $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ be the corresponding weighted automaton. Then $\mathsf{growth}_{\mathcal{M}}(k) = \Theta(k^{d+1})$ if and only if $\mathcal{A}$ does not comply with EDA nor with IDA_{d+1}, but complies with IDA_d. Furthermore this property is decidable in time $O(|Q|^6 \cdot |\Sigma|)$.* □

As a direct consequence of the theorem together with Theorem 21 we obtain the following corollary.

Corollary 25. *Let $\mathcal{R}$ be a TRS and let $\mathcal{M}$ be a compatible matrix interpretation of dimension n. Further, let $\mathcal{A}$ be the corresponding weighted automaton such that $\mathcal{A}$ does not comply with* EDA *nor with* IDA_{d+1}*. Then* $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$*. Furthermore the conditions given are decidable in polynomial time.* $\square$

Example 26 (continued from Example 23). It is easy to see that $\mathcal{A}$ does not comply with EDA nor with IDA_3. Hence the growth of the matrix interpretation $\mathcal{M}$ is at most cubic, i.e., $\mathsf{growth}_{\mathcal{M}}(k) = O(k^3)$.

5 Unifying Algebraic and Automata-Based Methods

In this section we unify the algebraic and the automata-based approach presented in Sections 3 and 4. We start by relating the two different notions of polynomial growth of matrix interpretations that have been studied in the literature (cf. Definitions 1 and 17). The next example shows that these two definitions are not polynomially related.

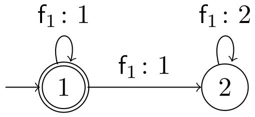
Example 27. Consider the TRS $\mathsf{f}(x) \to x$ together with the following compatible matrix interpretation $\mathcal{M}$:

$$\mathsf{f}_{\mathcal{M}}(\boldsymbol{x}) = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

It is easy to see that the growth of the entries in the second column of the matrix is exponential, so $\mathcal{M}$ is not of polynomial growth with respect to Definition 1. However, $[t]_1 < |t|$ for any term t and thus $\mathcal{M}$ is of polynomial growth with respect to Definition 17.

Still, *morally* both definitions are equal. Observe that the entries that grow exponentially in the matrix in Example 27 are irrelevant when computing the value $[t]_1$. In order to prove this we exploit the connection between matrix interpretations and weighted automata emphasized in Section 4.

Example 28 (continued from Example 27). The weighted automaton $\mathcal{A}$ corresponding to $\mathcal{M}$ can be pictured as follows:



Observe that $\mathcal{A}$ is not trim because state 2 is not useful.

Lemma 29. *Let $\mathcal{R}$ be a TRS and let $\mathcal{M}$ be a compatible matrix interpretation. There exists a matrix interpretation $\mathcal{N}$ compatible with $\mathcal{R}$ such that the corresponding automaton is trim. Furthermore,* $\mathsf{growth}_{\mathcal{M}}(k) = \mathsf{growth}_{\mathcal{N}}(k)$*.*

Proof. Following [19], with every position $p \in \mathcal{P}\mathsf{os}(t)$ we associate a word t_p over Σ:

$$t_p = \begin{cases} \epsilon & \text{if } p = \epsilon \\ f_i \cdot t_q & \text{if } p = i \cdot q \text{ and } t = f(t_1, \ldots, t_n) \end{cases}$$

The set $\mathsf{Path}(t)$ consists of all these words. We write $t(p)$ for the root symbol of the subterm of t at position p. Let $\mathcal{M}$ be a matrix interpretation of dimension n and let $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ be the corresponding normal weighted automaton. A key observation is the following identity, which holds for all assignments α of vectors in $\mathbb{N}^n$ to the variables in t:

$$[\alpha]_{\mathcal{M}}(t) = \sum_{p \in \mathcal{P}\mathsf{os}(t)} \mu(t_p) \cdot \mathsf{abs}(\alpha, t(p))$$

where $\mathsf{abs}(\alpha, t(p)) = \mathsf{abs}(t(p))$ if $t(p) \in \mathcal{F}$ and $\mathsf{abs}(\alpha, t(p)) = \alpha(t(p))$ otherwise. Without loss of generality we assume that $\mathcal{R}$ is non-empty. As $\mathcal{M}$ is compatible with $\mathcal{R}$, there must be a word in Σ^* of positive weight. This implies in particular that state 1 is useful. Now suppose that $\mathcal{A}$ is not trim. Let $\mathcal{B} = (Q', \Sigma, \lambda', \mu', \gamma')$ denote an equivalent trim automaton. We have $\lambda' = \lambda\restriction_{Q'}$, $\gamma' = \gamma\restriction_{Q'}$, and for all $a \in \Sigma$, $\mu'(a) = \mu(a)\restriction_{Q' \times Q'}$. To simplify notation we assume (without loss of generality; recall that state 1 is useful) that $Q' = \{1, \ldots, m\}$ for some $m < n$.

Let $\mathcal{N}$ be the matrix interpretation corresponding to $\mathcal{B}$, where the absolute vector of every $f_{\mathcal{N}}$ equals $\mathsf{abs}(f)\restriction_{Q'}$. We prove that $\mathcal{N}$ is compatible with $\mathcal{R}$. Let $\ell \to r$ be a rule in $\mathcal{R}$ and let $\beta \colon \mathcal{V} \to \mathbb{N}^m$ be an arbitrary assignment. Let $\alpha \colon \mathcal{V} \to \mathbb{N}^n$ be the assignment that is obtained from β by zero padding, i.e., $\beta(x)_i = \alpha(x)_i$ for $1 \leqslant i \leqslant m$ and $\beta(x)_i = 0$ for $m < i \leqslant n$. From the compatibility of $\mathcal{M}$ and $\mathcal{R}$ we obtain $[\alpha]_{\mathcal{M}}(\ell)_1 > [\alpha]_{\mathcal{M}}(r)_1$ and $[\alpha]_{\mathcal{M}}(\ell)_j \geqslant [\alpha]_{\mathcal{M}}(r)_j$ for all $1 < j \leqslant n$. We claim that $[\alpha]_{\mathcal{M}}(t)_i = [\beta]_{\mathcal{N}}(t)_i$ for all $1 \leqslant i \leqslant m$. From the claim we immediately obtain $[\beta]_{\mathcal{N}}(\ell)_1 > [\beta]_{\mathcal{N}}(r)_1$ and $[\beta]_{\mathcal{N}}(\ell)_j \geqslant [\beta]_{\mathcal{N}}(r)_j$ for all $1 < j \leqslant m$. It follows that $\mathcal{N}$ is compatible with $\mathcal{R}$ and by Definition 17, $\mathsf{growth}_{\mathcal{M}}(k) = \mathsf{growth}_{\mathcal{N}}(k)$.

To prove the claim, fix $i \in \{1, \ldots, m\}$ and let $\lambda_i = (\overline{0}, 1, \overline{0})$ be the row vector of dimension n where the 1 is at position i. Let $p \in \mathcal{P}\mathsf{os}(t)$. We have

$$\begin{aligned} \lambda_i \cdot \mu(t_p) \cdot \mathsf{abs}(\alpha, t(p)) &= (\mu(t_p) \cdot \mathsf{abs}(\alpha, t(p)))_i \\ &= (\mu'(t_p) \cdot \mathsf{abs}(\alpha, t(p))\restriction_{Q'})_i \\ &= (\mu'(t_p) \cdot \mathsf{abs}(\beta, t(p)))_i = \lambda'_i \cdot \mu'(t_p) \cdot \mathsf{abs}(\beta, t(p)) \end{aligned}$$

where $\lambda'_i = \lambda_i\restriction_{Q'}$. It follows that $[\alpha]_{\mathcal{M}}(t)_i = [\beta]_{\mathcal{N}}(t)_i$ for all $1 \leqslant i \leqslant m$. □

Example 30 (continued from Example 28). Because state 2 in the weighted automaton $\mathcal{A}$ is useless, it is removed to obtain an equivalent trim automaton $\mathcal{B}$. This one-state automaton gives rise to the 1-dimensional matrix interpretation (i.e., linear polynomial interpretation) $\mathcal{N}$ with $\mathsf{f}_{\mathcal{N}}(x) = x + 1$.

An immediate consequence of Theorem 21 and Lemma 29 is that Definitions 1 and 17 are equivalent in the following sense.

Corollary 31. *Let $\mathcal{R}$ be a TRS. The following statements are equivalent.*

1. *There exists a matrix interpretation $\mathcal{M}$ compatible with $\mathcal{R}$ that is polynomially bounded according to Definition 1.*
2. *There exists a matrix interpretation $\mathcal{N}$ compatible with $\mathcal{R}$ that is polynomially bounded according to Definition 17.*

Furthermore the entries of all products of matrices in $\mathcal{M}$ are polynomial with degree d in the length of such products if and only if $\mathsf{growth}_{\mathcal{N}}(k) = O(k^{d+1})$.

Proof. First, assume that $\mathcal{M}$ is a compatible matrix interpretation that is polynomially bounded according to Definition 1. Further assume the entries of all matrix products of matrices in $\mathcal{M}$ are polynomial with degree d in the length of such products. Since the matrices occurring in $\mathcal{M}$ form the transitions of the automaton $\mathcal{A}$ corresponding to $\mathcal{M}$, $\mathsf{growth}_{\mathcal{A}}(k) = O(k^d)$. Hence, by Theorem 21, $\mathsf{growth}_{\mathcal{M}}(k) = O(k^{d+1})$, which means that $\mathcal{M}$ is polynomially bounded according to Definition 17.

As to the converse statement, assume that $\mathcal{N}$ is a compatible matrix interpretation that is polynomially bounded according to Definition 17 and let $\mathsf{growth}_{\mathcal{N}}(k) = O(k^{d+1})$. By Lemma 29 there exists a compatible matrix interpretation $\mathcal{M}$ such that $\mathsf{growth}_{\mathcal{N}}(k) = \mathsf{growth}_{\mathcal{M}}(k)$ and the corresponding automaton $\mathcal{A}$ is trim. By Theorem 21, we have $\mathsf{growth}_{\mathcal{A}}(k) = O(k^d)$. This entails that all entries of all matrix products of matrices in $\mathcal{M}$ are polynomial with degree d, as $\mathcal{A}$ is trim, which means that $\mathcal{M}$ is polynomially bounded according to Definition 1. $\square$

In what follows we show that the algebraic approach presented in Section 3 readily subsumes the automata theory based approach of Section 4. To be precise, we show that the restriction of Theorem 16 to matrix interpretations over $\mathbb{N}$ applies in any situation where Corollary 25, the main result of Section 4, applies.

Theorem 32. *Let $\mathcal{R}$ be a TRS. If one can establish polynomial derivational complexity of some degree via Corollary 25, then one can also establish polynomial derivational complexity of the same degree via Theorem 16.*

Proof. Let $\mathcal{M}$ be a compatible matrix interpretation over $\mathbb{N}$ of dimension n and let $S \subseteq \mathbb{N}^{n \times n}$ denote the set of all matrices occurring in $\mathcal{M}$. By Lemma 29 we may assume that the automaton $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ corresponding to $\mathcal{M}$ is trim. By assumption $\mathcal{A}$ does not comply with EDA nor with IDA_{d+1} and $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$. As one obtains the tightest bound for $\mathsf{dc}_{\mathcal{R}}(k)$ if $d+1$ is the least integer such that $\neg\mathsf{IDA}_{d+1}$ holds in $\mathcal{A}$, or, equivalently, if d is the largest integer such that IDA_d holds in $\mathcal{A}$, this will be assumed in what follows.

First we show that $\neg\mathsf{EDA}$ implies $\rho(S) \leqslant 1$. By Lemma 12 we have $\rho(S) > 1$ if and only if there is a product $A \in S^*$ such that $A_{ii} > 1$ for some $i \in \{1, \ldots, n\}$. By construction of $\mathcal{A}$ the latter is equivalent to the existence of a word $x \in \Sigma^*$ (corresponding to the product $A \in S^*$) such that $\mu(x)_{ii} > 1$ for some state $i \in Q = \{1, \ldots, n\}$, or, equivalently, $\mu(x)_{ii} = \mathsf{weight}(i, x, i) \geqslant 2$ since $\mathcal{A}$ is $\mathbb{N}$-weighted. This means that $\mathcal{A}$ complies with EDA because state i is useful.

Next we show that d is exactly the growth rate mentioned in Theorem 16, that is, the growth rate inferred from Theorem 14. As $\mathcal{A}$ complies with IDA_d, there are useful states $p_1, q_1, \dots, p_d, q_d \in Q$ and words $v_1, u_2, v_2, \dots, u_d, v_d \in \Sigma^*$ such that $p_s \neq q_s$, $p_s \xrightarrow{v_s} p_s$, $p_s \xrightarrow{v_s} q_s$, and $q_s \xrightarrow{v_s} q_s$ for all $s = 1, \dots, d$ and $q_{s-1} \xrightarrow{u_s} p_s$ for all $s = 2, \dots, d$. Letting $A_s = \mu(v_s) \in S^*$ for $s = 1, \dots, d$ and $B_s = \mu(u_s) \in S^*$ for $s = 2, \dots, d$ (by the one-to-one correspondence between letters in Σ and matrices in S), this is equivalent to the existence of d pairs of indices $(p_1, q_1), \dots, (p_d, q_d)$ and matrix products $A_1, B_2, A_2, \dots, B_d, A_d \in S^*$ such that $p_s \neq q_s$ and $(A_s)_{p_s p_s}, (A_s)_{p_s q_s}, (A_s)_{q_s q_s} \geqslant 1$ for all $s = 1, \dots, d$ and $(B_s)_{q_{s-1} p_s} \geqslant 1$ for all $s = 2, \dots, d$. Note that all pairs of indices (states) must be different because otherwise $\mathcal{A}$ would comply with EDA, contradicting our assumption. Hence, all conditions concerning the growth rate in Theorem 14 are satisfied and Theorem 16 yields $\mathsf{dc}_{\mathcal{R}}(k) = O(k^{d+1})$. □

6 Automation and Experimental Results

6.1 Automation

In this section we describe certificates for polynomial boundedness of matrix interpretations. These certificates will be described by finite-domain constraint systems and are solved by machine. The constraint domain consists of relations on the state set of an automaton. In our implementations, the constraint system is solved via translation to a constraint system in propositional logic. This is motivated by the intended application in conjunction with a constraint system for the entries in the matrices that describes its compatibility with a given TRS [4]. So if there is a solution, it fulfills both properties (compatibility and polynomial growth).

First we focus on how to ensure $\rho(A) \leqslant 1$ for a single n-dimensional matrix with non-negative real entries, which is needed to implement Theorem 5. We base our encoding of the minimal polynomial m_A on the factorization approach **(C)** from [15] but instead of demanding certain properties of the characteristic polynomial we encode a monic polynomial p that annihilates A such that $|\lambda| \leqslant 1$ for every root λ of p. One candidate for p is the characteristic polynomial of A, so the degree of p can be chosen n. To cancel some factors we introduce variables $C, C_j \in \{0, 1\}$ such that

$$p(\lambda) = (C\lambda - Cr + 1 - C)^b \cdot \prod_j (C_j\lambda^2 + C_j p_j \lambda + C_j q_j + 1 - C_j)$$

Here $r, p_j, q_j \in \mathbb{R}$ and $b = 0$ if n is even and $b = 1$ otherwise. Note that if C is zero then this factor simplifies to one and hence does not affect the product, while if C is one, the factor contributes to p. The same property holds for the C_j. We add a constraint $p(A) = 0$ to the encoding which ensures that p annihilates A. By the shape of the factorization, p is automatically monic. The condition $|\lambda| \leqslant 1$ is encoded based on Cr, $C_j p_j$, $C_j q_j$ as in [15] if $C_j = 1$, while $C_j = 0$ does not require additional constraints. Finding the minimal polynomial m_A is then an optimization problem, i.e., it amounts to minimize the maximum multiplicity of roots with absolute value exactly one.

Next we sketch how to count the number of roots equal to one. We abbreviate $(C_jp_j)^2 - 4C_jq_j$ by D_j. Obviously $Cr = 1$ corresponds to an occurrence of root one and $Cr = -1$ to an occurrence of root minus one. For the other factors, inspecting the quadratic formula gives root one if $C_jp_j + C_jq_j = -1$. Then $D_j = 0$ corresponds to a double occurrence and $D_j > 0$ to a single occurrence. The reasoning for root minus one is similar and based on $C_jp_j = C_jq_j + 1$. To keep the encoding simple, we over-approximate the multiplicity of a complex root with absolute value one (i.e., we possibly identify different complex roots for counting). This counter is incremented by one if $D_j < 0$ and $C_jq_j = 1$.

The following example shows that computing the minimal polynomial after establishing a compatible matrix interpretation may not result in optimal bounds.

Example 33. Consider the TRS consisting of the single rule $\mathsf{f}(x) \to x$ and assume a compatible matrix interpretation of dimension n having maximum matrix A with $A_{ij} = 0$ if $i > j$ and $A_{ij} = 1$ otherwise. Then $\chi_A(x) = m_A(x)$ and hence Theorem 5 establishes a polynomial bound of degree n. However, if the maximum matrix equals I_n then Theorem 5 establishes a linear upper bound on the derivational complexity.

Next we aim to describe polynomial growth of arbitrary matrix products with entries from $\mathbb{N}$. Section 4 gives polynomial-time algorithms for checking polynomial boundedness and computing the degree of growth for weighted automata. We will express these algorithms as constraint systems. By Cook's theorem, we know that any P (even NP) computation can be simulated by a polynomially-sized SAT formula. The size of the formula is the product of space and time of the computation. Thus a straightforward translation of the decision procedure for polynomial boundedness ($\neg\mathsf{EDA}$) results in a formula size of $O(n^6)$, and for polynomial boundedness of a fixed degree ($\neg\mathsf{IDA}_{d+1}$), in a formula size of $O(n^9)$, where n is the dimension of the matrices.

Note that Cook's translation produces a formula where the number of variables is of the same order as the number of clauses. We will strive to reduce theses numbers, and in particular, the number of variables. E.g., we implement the criterion $\neg\mathsf{IDA}_{d+1}$ by $O(n^8)$ clauses but only $O(n^5)$ variables. The fewer variables, the less choice points there are for the SAT solver, and for a fixed set of variables, more constraints allow for more unit propagations, hopefully speeding up the solver even more.

The formula size is reduced by replacing parts of the original algorithms by approximations, but using them in such a way as to still ensure correctness and completeness. E.g., for reachability with respect to a set of matrices S, we specify a relation $R \subseteq Q^2$ such that R is reflexive and transitive and, for all $p, q \in Q$, $G(S)_{pq} > 0$ implies $(p, q) \in R$,

While the presence of patterns (EDA, IDA_d) is easily described by a constraint system in existential logic, the challenge is to specify the *absence* of those patterns.

$\neg\mathsf{EDA}$*:* By the algorithm given right after Lemma 12, we use the following constraints, for a given set S of matrices:

– Condition 1 follows from the constraint $G(S)_{ij} > 1 \Rightarrow (j,i) \notin R$, for all i and j, where R is the over-approximation of reachability discussed above.
– For condition 2, denote by G_2 the adjacency relation of $G^2(S)$. A cycle from (i,i) through some (p,q) for $p \neq q$ means that (p,q) is strongly G_2-connected to (i,i). Therefore, we set $D = \{(q,q) \mid q \in Q\}$ and specify an over-approximation D_1 of the index pairs reachable from the main diagonal by $D \subseteq D_1$ and $G_2(D_1) \subseteq D_1$, and an over-approximation D_2 of the index pairs that can reach the main diagonal by $D \subseteq D_2$ and $G_2^-(D_2) \subseteq D_2$, and then require $D_1 \cap D_2 \subseteq D$.

The resulting constraint system is satisfiable if and only if $\mathcal{A}$ does not comply with EDA. So we have a complete characterization of polynomial boundedness, equivalently, of $\rho(S) \leqslant 1$ for a set of non-negative integer matrices (cf. Lemma 13 and Theorem 22).

The constraint system has $O(n^2)$ variables and $O(n^4)$ constraints, which is small in comparison to the size of the constraint system that describes compatibility of the interpretation with the rewrite system [4].

$\neg\mathsf{IDA}_{d+1}$*:* We use the relation $I \subseteq Q^2$ given by

$$I = \{(p,q) \mid p \neq q \text{ and } p \xrightarrow{x} p,\, p \xrightarrow{x} q,\, q \xrightarrow{x} q \text{ for some } x \in \Sigma^+\}$$

and let $J = I \circ R$, where R is an over-approximation of reachability as defined previously. Then IDA_d implies that there is a J-chain of length d, and conversely, if there is no J-chain of length $d+1$ then $\neg\mathsf{IDA}_{d+1}$ holds. To specify I, we use the graph $G^3(S)$, and denote its adjacency relation by G_3. For each (p,q) with $p \neq q$ we specify a set $T \subseteq Q^3$ by $(p,p,q) \in T$ and $G_3(T) \subseteq T$. Then $(p,q,q) \in T$ implies $I(p,q)$. We bound the length of J-chains by encoding a height function $h\colon Q \to \{0,1,\ldots,d\}$ and a monotonicity property $J(p,q) \Rightarrow h(p) > h(q)$. Since only comparison but no arithmetic is needed here, it is convenient to represent numbers in unary notation.

The resulting constraint system is satisfiable if and only if $\neg\mathsf{IDA}_{d+1}$, so we obtain a characterization of growth $O(k^d)$. The system has $O(n^5)$ variables and $O(n^8)$ constraints. This is comparable in size to the compatibility constraints.

6.2 Experimental Results

The criteria proposed in this paper have been implemented in the complexity tools CaT [21] and matchbox [19]. All tests have been performed on a server equipped with 64 GB of main memory and eight dual-core AMD Opteron® 885 processors running at a clock rate of 2.6 GHz with a time limit of 60 seconds per system. For experiments[3] the 295 non-duplicating TRSs in TPDB 8.0 for which CaT could find a compatible matrix interpretation (not necessarily polynomially bounded) have been considered.

We searched for matrix interpretations of dimension $n \in \{1,\ldots,5\}$ by encoding the constraints as an SMT problem (quantifier-free non-linear arithmetic),

[3] For full details see http://colo6-c703.uibk.ac.at/hzankl/11cai

Table 1. Polynomial bounds for 295 systems

	$O(k)$	$O(k^2)$	$O(k^3)$	$O(k^n)$
triangular	92	194	208	210
Theorem 4	66	180	194	196
Theorem 5	72	187	193	194
$\sum$	92	201	215	216
Theorem 15 ($\neg$EDA)	73	173	188	213
Theorem 16 ($\neg$EDA, $\neg$IDA$_{d+1}$)	107	214	219	224
$\sum$	110	225	231	236
$\sum$	112	227	235	239

which is solved by "bit-blasting". That means the problem is transformed into a finite domain problem by prescribing discrete and finite domains for numerical unknowns. Then we encode these discrete domains by propositional values, to obtain a propositional satisfiability problem, for which efficient solvers are available. For finding matrix interpretations, this is a successful method, even with surprisingly small domains that can be represented by at most 5 bits. In the experiments we only considered matrix interpretations over the natural numbers.

Table 1 indicates the number of systems where polynomial upper bounds on the derivational complexity could be established. The rows in the table correspond to different approaches to polynomially bound the derivational complexity of rewrite systems and the columns give the degree of these polynomials. Triangular matrix interpretations [14, 15, 19] serve as a reference (here the degree of the polynomial is determined by the number of ones in the main diagonal).

The first three rows (and the accumulated results in row four) operate on the component-wise maximum matrix. While in theory Theorem 4 allows to establish strictly more polynomial bounds than triangular matrices, in practice the constraints are harder to solve, resulting in a worse overall score. A similar argument holds when comparing Theorems 4 and 5, i.e., explaining the better performance of the latter for low bounds but worse result for polynomial bounds. The accumulative score of the first three rows justifies each of the approaches.

The criteria in the next two rows are not restricted to the growth of the maximum matrix. In the former row no explicit bounds on the degree are added to the constraints (so the dimension of the matrices dominates the degree of polynomial growth) while in the latter row upper bounds are explicitly added to the search. Somehow surprisingly, the accumulated score of these two rows is noticeable larger than the score for each row on its own.

The final row in Table 1 shows that the methods from the first block do not significantly increase the power of the second block (in our experiments). However, the criteria from the first block are also suitable for matrix interpretations over $\mathbb{R}$, while the methods involved in the second block become undecidable over this domain.

7 Concluding Remarks

In this paper we employed joint spectral radius theory to unify as well as clarify the different approaches to obtain upper bounds on the derivational complexity of rewrite systems from compatible matrix interpretations. Our results are not limited to the study of *derivational* complexity, but can also be employed in the context of *runtime* complexity of rewrite systems [7].

It remains to be seen whether joint spectral radius theory will advance implementations (for matrix interpretations over $\mathbb{Q}$ and $\mathbb{R}$). For matrices with rational or real entries the joint spectral radius is not computable in general. In the future we will investigate whether good approximations can be obtained to improve the complexity bounds inferred from matrix interpretations over $\mathbb{Q}$ and $\mathbb{R}$ [5, 22].

We conclude the paper by observing that matrix interpretations are incomplete when it comes to establishing polynomial derivational complexity. This shows that new ideas are necessary to obtain a complete characterization of TRSs with polynomial derivational complexity (cf. RTA open problem #107).[4]

Lemma 34. *There exists a TRS with linear derivational complexity that is compatible with a matrix interpretation but not with a polynomially bounded one.*

Proof. To prove this result we extend the TRS $\mathcal{R}$ from Example 7 with the rule $\mathsf{b}(x) \to \mathsf{g}(x)$. The resulting TRS $\mathcal{S}$ has linear derivational complexity since it is match-bounded by 3 [6]. To obtain a matrix interpretation compatible with $\mathcal{S}$, we change the interpretation of b in the proof of Lemma 8 to

$$\mathsf{b}_{\mathcal{M}}(\boldsymbol{x}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 4 \\ 4 \\ 3 \end{pmatrix}$$

That there cannot exist a polynomially bounded matrix interpretation for $\mathcal{S}$ follows from Corollary 31 and the proof of Lemma 8 since $B \geqslant \max(I_n, G)$ and hence entries in B^k grow exponentially. □

Since the TRS in the proof of the above lemma is a string rewrite system and match-bounded, this lemma also answers a question by Waldmann, cf. [19, Section 10]: *Do all match-bounded string rewrite systems have a polynomially (even linearly) bounded matrix interpretation?* Here linearly bounded refers to Definition 17.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998)
2. Bell, J.: A gap result for the norms of semigroups of matrices. LAA 402, 101–110 (2005)

[4] `http://rtaloop.mancoosi.univ-paris-diderot.fr/problems/107.html`

3. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Springer, Heidelberg (2009)
4. Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. JAR 40(2-3), 195–220 (2008)
5. Gebhardt, A., Hofbauer, D., Waldmann, J.: Matrix evolutions. In: WST 2007, pp. 4–8 (2007)
6. Geser, A., Hofbauer, D., Waldmann, J., Zantema, H.: On tree automata that certify termination of left-linear term rewriting systems. I&C 205(4), 512–534 (2007)
7. Hirokawa, N., Moser, G.: Automated complexity analysis based on the dependency pair method. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 364–379. Springer, Heidelberg (2008)
8. Horn, R., Johnson, C.: Matrix Analysis. Cambridge University Press, Cambridge (1990)
9. Jungers, R.M., Protasov, V., Blondel, V.D.: Efficient algorithms for deciding the type of growth of products of integer matrices. LAA 428(10), 2296–2311 (2008)
10. Jungers, R.: The Joint Spectral Radius: Theory and Applications. Springer, Heidelberg (2009)
11. Koprowski, A., Waldmann, J.: Arctic termination...Below zero. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 202–216. Springer, Heidelberg (2008)
12. Moser, G.: Proof Theory at Work: Complexity Analysis of Term Rewrite Systems. Habilitation thesis, University of Innsbruck (2009)
13. Moser, G., Schnabl, A.: The derivational complexity induced by the dependency pair method. LMCS (2011) (to appear), `http://arxiv.org/abs/0904.0570`
14. Moser, G., Schnabl, A., Waldmann, J.: Complexity analysis of term rewriting based on matrix and context dependent interpretations. In: Hariharan, R., Mukund, M., Vinay, V. (eds.) FSTTCS 2008. LIPIcs, vol. 2, pp. 304–315. Schloss Dagstuhl, Dagstuhl (2008)
15. Neurauter, F., Zankl, H., Middeldorp, A.: Revisiting matrix interpretations for polynomial derivational complexity of term rewriting. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS(ARCoSS), vol. 6397, pp. 550–564. Springer, Heidelberg (2010)
16. Rose, H.E.: Linear Algebra: A Pure Mathematical Approach. Birkhäuser, Basel (2002)
17. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Cambridge (2009)
18. Terese: Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press, Cambridge (2003)
19. Waldmann, J.: olynomially bounded matrix interpretations. In: Lynch, C. (ed.) RTA 2010. LIPIcs, vol. 6, pp. 357–372. Schloss Dagstuhl, Dagstuhl (2010)
20. Weber, A., Seidl, H.: On the degree of ambiguity of finite automata. TCS 88(2), 325–349 (1991)
21. Zankl, H., Korp, M.: Modular complexity analysis via relative complexity. In: Lynch, C. (ed.) RTA 2010. LIPIcs, vol. 6, pp. 385–400. Schloss Dagstuhl, Dagstuhl (2010)
22. Zankl, H., Middeldorp, A.: Satisfiability of non-linear (ir)rational arithmetic. In: Clarke, E.M., Voronkov, A. (eds.) LPAR-16. LNCS(LNAI), vol. 6355, pp. 481–500. Springer, Heidelberg (2010)

From Grammars and Automata to Algebras and Coalgebras

Peter Padawitz

Technical University of Dortmund
Germany

Abstract. The increasing application of notions and results from category theory, especially from algebra and coalgebra, has revealed that any formal software or hardware model is constructor- or destructor-based, a *white-box* or a *black-box* model. A highly-structured system may involve both constructor- and destructor-based components. The two model classes and the respective ways of developing them and reasoning about them are dual to each other. Roughly said, algebras generalize the modeling with context-free grammars, word languages and structural induction, while coalgebras generalize the modeling with automata, Kripke structures, streams, process trees and all other state- or object-oriented formalisms. We summarize the basic concepts of co/algebra and illustrate them at a couple of signatures including those used in language or compiler construction like regular expressions or acceptors.

1 Introduction

More than forty years of research on formal system modeling led to the distinction between algebraic models on the one hand and coalgebraic ones on the other. The former describes a system in terms of the synthesis of its components by means of object-building operators (constructors). The latter models a system in terms of the analysis of its components by means of object-modifying, -decomposing or -measuring operators (destructors). The traditional presentation of a class of *algebraic* models is a context-free grammar that provides a concrete syntax of a set of constructors, whereas a class of *coalgebraic* models is traditionally given by all automata, transition systems or Kripke structures with the same *behavior*. Their respective state transition or labeling functions yield a set of the destructors. But not *any* member of such a class of models admits the application of powerful methods to operate on and reason about it. Among the members of an algebraic class it is the *initial* one, among those of a coalgebraic class it is the *final* one that the modeling should aim at. Initial algebras enable recursion and induction. Final coalgebras enable corecursion and coinduction.

Twenty years ago algebraic modeling was mainly algebraic specification and thus initial and free algebras were the main objects of interest [18,17,14,9], although *hidden algebra* and *final semantics* approaches [21,30,16,27,47,48] already tended to the coalgebraic view (mostly in terms of greatest quotients of initial models). But first the dual concepts of category and fixpoint theory paved the way to the principles and methods current algebraic system modeling is based upon.

Here we use a slim syntax of types and many-sorted signatures, expressive enough for describing most models one meets in practice, but avoiding new guises for well-established categorical concepts. For instance and in contrast to previous *hierarchical* approaches (including our own), we keep off the explicit distinction of primitive or base sorts and a fixed base algebra because such entities are already captured by the constant functors among the types of a signature. Section 2 presents the syntax and semantics of the domains for co/algebraic models of constructive resp. destructive signatures. Section 3 draws the connection from signatures to functor co/algebras and provides initial and final model constructions. Roughly said, the latter are least resp. greatest fixpoints of the respective functor. Section 4 presents fundamental concepts and rules dealing with the extension, abstraction or restriction of and the logical reasoning about co/algebraic models. Again we are faced with least and greatest fixpoints, here with the relational ones co/inductive proofs are based upon. Moreover, each element of a – usually coalgebraic – class of infinite structures can be modeled as the *unique* fixpoint of the function derived from a set of guarded equations.

We assume that the reader is somewhat familiar with the notions of a category, a functor, a diagram, a co/cone, a co/limit, a natural transformation and an adjunction. Given a category $\mathcal{K}$, the target object of a colimit resp. source object of a limit of the empty diagram $\emptyset \to \mathcal{K}$ is called **initial** resp. **final in** $\mathcal{K}$. We remind of the uniqueness of co/limits modulo or up to isomorphism. Hence initial or final objects are also unique up to isomorphism.

Set denotes the category of sets with functions as morphisms. Given an index set I, $\prod_{s\in I} A_i$ and $\coprod_{s\in I} A_i$ denote the product resp. coproduct (= disjoint union) of sets A_i, $i \in I$. For all $i \in I$, $\pi_i : \prod_{s\in I} A_i \to A_i$ and $\iota_i : A_i \to \coprod_{s\in I} A_i$ denote the i-th projection resp. injection: For all $a = (a_i)_{i\in I} \in \prod_{s\in I} A_i$, $i \in I$ and $b \in A_i$, $\pi_i(a) = a_i$ and $\iota_i(b) = (b, i)$. Given functions $f_i : A \to A_i$ and $g_i : A_i \to A$ for all $i \in I$, $\langle f_i\rangle_{i\in I} : A \to \prod_{s\in I} A_i$ and $[g_i]_{i\in I} : \coprod_{s\in I} A_i \to A$ denote the product resp. coproduct extension of $\{f_i\}_{i\in I}$: For all $a \in A$, $i \in I$ and $b \in A_i$, $\langle f_i\rangle_{i\in I}(a) = (f_i(a))_{i\in I}$, $[g_i](b, i) = g_i(b)$, $\prod_{s\in I} f_i = \langle f_i \circ \pi_i\rangle$ and $\coprod_{s\in I} g_i = [\iota_i \circ g_i]$.

1 denotes the singleton $\{*\}$. 2 denotes the two-element set $\{0, 1\}$. The elements of 2 are regarded as truth values. Let A be a set. $id_A : A \to A$ denotes the identity on A. A^*, $\mathcal{P}_{fin}(A)$ and $\mathcal{B}_{fin}(A) = \{f : A \to \mathbb{N} \mid supp(f) = f^{-1}(\mathbb{N} \setminus \{0\}) \text{ is finite}\}$ denote the sets of finite words, finite sets resp. finite multisets of elements of A.

2 Many-Sorted Signatures and Their Algebras

Let S be a set of **sorts**. An S**-sorted** or S**-indexed set** is a family $A = \{A_s \mid s \in S\}$ of sets. An S**-sorted subset** of A, written as $B \subseteq A$, is an S-sorted set with $A \subseteq B$ for all $s \in S$. Given S-sorted sets $A_1, \ldots, A_n$, an S**-sorted relation** $r \subseteq A_1 \times \ldots \times A_n$ is an S-sorted set with $r_s \subseteq A_{1,s} \times \ldots \times A_{n,s}$ for all $s \in S$. Given S-sorted sets A, B, an S**-sorted function** $f : A \to B$ is an S-sorted set such that for all $s \in S$, f_s is a function from A_s to B_s. Set^S denotes the category of S-sorted sets as objects and S-sorted functions as morphisms.

$\mathbb{BT}(S)$ denotes the inductively defined set of (bounded) **types over** S:

$$\begin{array}{lcl} S \subseteq \mathbb{BT}(S), & & \\ X \in Set & \Rightarrow & X \in \mathbb{BT}(S), \\ e_1,\dots,e_n \in \mathbb{BT}(S) & \Rightarrow & e_1 \times \ldots \times e_n, e_1 + \ldots + e_n \in \mathbb{BT}(S), \\ e \in \mathbb{BT}(S) & \Rightarrow & word(e), bag(e), set(e) \in \mathbb{BT}(S), \\ X \in Set \wedge e \in S & \Rightarrow & e^X \in \mathbb{BT}(S). \end{array}$$

We regard $e \in \mathbb{BT}(S)$ as a finite tree: Each inner node of e is labelled with a *type constructor* ($\times$, $+$, *list*, *bag*, *set* or $\_^X$ for some $X \in Set$) and each leaf is labelled with an element of S.

$e \in \mathbb{BT}(S)$ is **polynomial** if e does not contain *set*. $\mathbb{PT}(S)$ denotes the set of polynomial types over S.

The meaning of $e \in \mathbb{BT}(S)$ is a functor $F_e : Set^S \to Set$ that is inductively defined as follows (also called **predicate lifting**; see [24,25]): Let A, B be S-sorted sets, $h : A \to B$ be an S-sorted function, $s \in S$, $e, e_1, \dots, e_n \in \mathbb{BT}(S)$, $a_1, \dots, a_n \in F_e(A)$, $f \in \mathcal{B}_{fin}(F_e(A))$, $B \in \mathcal{P}_{fin}(F_e(A))$, $b \in B$, $X \in Set$ and $g : X \to F_e(A)$.

$$\begin{array}{l} F_s(A) = A_s,\ F_s(h) = h_s,\ F_X(A) = X,\ F_X(h) = id_X, \\ F_{e_1\times\ldots\times e_n}(A) = \prod_{i=1}^n F_{e_i}(A),\ F_{e_1\times\ldots\times e_n}(h) = \prod_{i=1}^n F_{e_i}(h), \\ F_{e_1+\ldots+e_n}(A) = \coprod_{i=1}^n F_{e_i}(A),\ F_{e_1+\ldots+e_n}(h) = \coprod_{i=1}^n F_{e_i}(h), \\ F_{word(e)}(A) = F_e(A)^*,\ F_{word(e)}(h)(a_1\dots a_n) = F_e(h)(a_1)\dots F_e(h)(a_n), \\ F_{bag(e)}(A) = \mathcal{B}_{fin}(F_e(A)),\ F_{bag(e)}(h)(f)(b) = \sum\{f(a) \mid F_e(h)(a) = b\}, \\ F_{set(e)}(A) = \mathcal{P}_{fin}(F_e(A)),\ F_{set(e)}(h)(B) = \{F_e(h)(b) \mid b \in B\}, \\ F_{e^X}(A) = (X \to F_e(A)),\ F_{e^X}(h)(g) = F_e(h) \circ g. \end{array}$$

Each function $E : S \to \mathbb{BT}(S)$ induces an endofunctor $F_E : Set^S \to Set^S$: For all $s \in S$, $F_E(A)(s) = F_{E(s)}(A)$ and $F_E(h)(s) = F_{E(s)}(h)$.

Given S-sorted sets A, B and an S-sorted relation $r \in A \times B$, the **relation lifting** $Rel_e(r) \subseteq F_e(A) \times F_e(B)$ of r is inductively defined as follows (analogously to [24,25]): Let $s \in S$, $e, e_1, \dots, e_n \in \mathbb{BT}(S)$ and $X \in Set$.

$$\begin{array}{l} Rel_s(r) = r_s,\ Rel_X(r) = \langle id_X, id_X\rangle(X), \\ Rel_{e_1\times\ldots\times e_n}(r) = \{((a_1,\dots,a_n),(b_1,\dots,b_n)) \mid \forall\, 1 \le i \le n : (a_i,b_i) \in Rel_{e_i}(r)\}, \\ Rel_{e_1+\ldots+e_n}(r) = \{((a,i),(b,i)) \mid (a,b) \in Rel_{e_i}(r),\ 1 \le i \le n\}, \\ Rel_{word(e)}(r) = \{(a_1\dots a_n, b_1\dots b_n) \mid \forall\, 1 \le i \le n : (a_i,b_i) \in Rel_e(r),\ n \in \mathbb{N}\}, \\ Rel_{bag(e)}(r) = \{(f,g) \mid \exists\, p : supp(f) \xrightarrow{\sim} supp(g) : \\ \qquad\qquad \forall\, a \in supp(f) : f(a) = g(p(a)) \wedge (a,p(a)) \in Rel_e(r)\}, \\ Rel_{set(e)}(r) = \{(C,D) \mid \forall\, c \in C\ \exists\, d \in D : (c,d) \in Rel_e(r)\ \wedge \\ \qquad\qquad \forall\, d \in D\ \exists\, c \in C : (c,d) \in Rel_e(r), \\ Rel_{e^X}(r) = \{(f,g) \mid \forall\, x \in X : \langle f,g\rangle(x) \in Rel_e(r)\}. \end{array}$$

A **signature** $\Sigma = (S, F, R)$ consists of a finite set S (of sorts), a finite $\mathbb{BT}(S)^2$-sorted set F of **function symbols** and a finite $\mathbb{BT}(S)$-sorted set R of **relation**

symbols. $f \in F_{(e,e')}$ is written as $f : e \to e' \in F$. $dom(f) = e$ is the **domain** of f, $ran(f) = e'$ is the **range** of f. $r \in R_e$ is written as $r : e \in R$. $f : e \to e'$ is an e'**-constructor** if $e' \in S$. f is an e**-destructor** if $e \in S$. Σ is **constructive** resp. **destructive** if F consists of constructors resp. destructors. Σ is **polynomial** if for all $f : e \to e' \in F$, e' is polynomial. A set X is a **base set** of Σ if it occurs in the domain or range of a function or relation symbol and thus induces the constant functor F_X.

Example 2.1. Here are some constructive signatures without relation symbols. Let X and Y be sets.

1. *Nat* (natural numbers) $S = \{nat\}$, $F = \{0 : 1 \to nat,\ succ : nat \to nat\}$.
2. *Reg*(X) (regular expressions over X) $S = \{reg\}$,

$$F = \{\ \emptyset,\ \epsilon : 1 \to reg,\ \_ : X \to reg,\\ \_|\_,\ \_\cdot\_ : reg \times reg \to reg,\ star : reg \to reg\ \}.$$

3. *List*(X) (finite sequences of elements of X) $S = \{list\}$, $F = \{nil : 1 \to list,\ cons : X \times list \to list\}$.
4. *Tree*(X,Y) (finitely branching trees of finite depth with node labels from X and edge labels from Y) $S = \{tree, trees\}$,

$$F = \{\ join : Y \times trees \to tree,\ nil : 1 \to trees,\\ cons : X \times tree \times trees \to trees\ \}.$$

5. *BagTree*(X,Y) (finitely branching unordered trees of finite depth) $S = \{tree\}$, $F = \{join : Y \times bag(X \times tree) \to tree\}$.
6. *FDTree*(X,Y) (finitely or infinitely branching trees of finite depth) $S = \{tree\}$, $F = \{join : Y \times ((X \times tree)^{\mathbb{N}} + word(X \times tree)) \to tree\}$. ❑

Example 2.2. Here are some destructive signatures without relation symbols. For each signature, we list its base sorts, sorts and function symbols. All other components are empty.

1. *CoNat* (natural numbers with infinity) $S = \{nat\}$, $F = \{pred : nat \to 1 + nat\}$.
2. *CoList*(X) (finite or infinite sequences of elements of X; *CoList*$(1) \simeq$ *CoNat*) $S = \{list\}$, $F = \{split : list \to 1 + (X \times list)\}$.
3. *DetAut*(X,Y) (deterministic Moore automata with input set X and output set Y) $S = \{state\}$, $F = \{\delta : state \to state^X,\ \beta : state \to Y\}$.
4. *NDAut*(X,Y) (non-deterministic Moore automata; image finite labelled transition systems) $S = \{state\}$, $F = \{\delta : state \to set(state)^X,\ \beta : state \to Y\}$.
5. *CoTree*(X,Y) (finitely or infinitely branching trees of finite or infinite depth with node labels from X and edge labels from Y) $S = \{tree, trees\}$,

$$F = \{\ root : tree \to Y,\ subtrees : tree \to trees,\\ split : trees \to 1 + (X \times tree \times trees)\ \}.$$

6. $FBTree(X,Y)$ (finitely branching trees of finite or infinite depth) $S = \{tree\}$, $F = \{root : tree \to Y,\ subtrees : tree \to word(X \times tree)\}$. ❏

Let $\Sigma = (S, F, R)$ be a signature. A **Σ-algebra** A consists of an S-sorted set, the **carrier** of A, also denoted by A, for each $f : e \to e' \in F$, a function $f^A : F_e(A) \to F_{e'}(A)$, and for each $r : e \in R$, a relation $r^A \subseteq F_e(A)$.

Let A and B be Σ-algebras, $h : A \to B$ be an S-sorted function and $f : e \to e' \in F$. h is **compatible with** f if $F_{e'}(h) \circ f^A = f^B \circ F_e(h)$. h is a **Σ-homomorphism** if for all $f \in F$, h is compatible with f and for all $r : e \in R$, $F_e(h)(r^A) \subseteq r^B$. h is **relation preserving** if the converse holds true as well, i.e., for all $r : e \in R$, $r^B \subseteq F_e(h)(r^A)$. A **$\Sigma$-isomorphism** is a bijective and relation preserving Σ-homomorphism. Alg_Σ denotes the category of Σ-algebras and Σ-homomorphisms.

A signature $\Sigma' = (S', F', R')$ is a **subsignature** of Σ if $S' \subseteq S$, $F' \subseteq F$ and $R' \subseteq R$. Let A be a Σ-algebra and $h : A \to B$ be a Σ-homomorphism. The **Σ'-reducts** $A|_{\Sigma'}$ and $h|_\Sigma$ of A resp. h are the Σ'-algebra resp. Σ'-homomorphism defined as follows:

- For all $s \in S'$, $(A|_{\Sigma'})_s = A_s$ and $(h|_\Sigma)_s = h_s$.
- For all $f \in F' \cup R'$, $f^{A|_{\Sigma'}} = f^A$.

Σ'-reducts yield the **reduct functor** or **forgetful functor** $\_|_{\Sigma'}$ from Alg_Σ to $Alg_{\Sigma'}$.

A constructive signature $\Sigma = (S, F, R)$ **admits terms** if for all $f \in F$ there are $e_1, \dots, e_n \in S \cup Set$ with $dom(f) = e_1 \times \dots \times e_n$. If Σ admits terms, then the Σ-algebra T_Σ of (ground) **Σ-terms** is defined inductively as follows:

- For all $s \in S$, $f : e \to s \in F$ and $t \in F_e(T_\Sigma)$, $f^{T_\Sigma}(t) = ft \in T_{\Sigma,s}$.

If a Σ-term is regarded as a tree, each inner node is labelled with some $f \in F$, while each leaf is labelled with an element of a base set of Σ. The interpretation of R in T_Σ is not fixed. Any such interpretation would be an S-sorted set of term relations, usually called a *Herbrand structure*. Constructive signatures that admit terms can be presented as context-free grammars:

A **context-free grammar** $G = (S, Z, BS, P)$ consists of finite sets S of **sorts** (also called nonterminals), Z of **terminals**, BS of **base sets** and $P \subseteq S \times (S \cup Z \cup BS)^*$ of **rules** (also called productions). The constructive signature $\Sigma(G) = (S, F, \emptyset)$ with

$$F \ = \ \{f_p : e_1 \times \dots \times e_n \to s \mid \begin{matrix} p = (s, w_0 e_1 w_1 \dots e_n w_n) \in P, \\ w_0, \dots, w_n \in Z^*,\ e_1, \dots, e_n \in S \cup BS \end{matrix}\}$$

is called the **abstract syntax** of G (see [18], Section 3.1; [45], Section 3). $\Sigma(G)$-terms are usually called **syntax trees** of G.

Example 2.3. The regular expressions over X form the reg-carrier of the $Reg(X)$-algebra $T_{Reg(X)}$ of $Reg(X)$-terms.

The usual interpretation of regular expressions over X as languages (= sets of words) over X yields the $Reg(X)$-algebra $Lang$: $Lang_{reg} = \mathcal{P}(X^*)$. For all

$x \in X$ and $L, L' \in \mathcal{P}(X^*)$,

$$\begin{array}{l} \emptyset^{Lang} = \emptyset,\ \epsilon^{Lang} = \{\epsilon\},\ \_^{Lang}(x) = \{x\}, \\ L|^{Lang}L' = L \cup L',\ L \cdot^{Lang} L' = \{vw \mid v \in L,\ w \in L'\}, \\ star^{Lang}(L) = \{w_1 \dots w_n \mid n \in \mathbb{N},\ \forall\ 1 \le i \le n : w_i \in L\}. \end{array}$$

The $Reg(X)$-Algebra *Bool* interprets the regular operators as Boolean functions: $Bool_{reg} = 2$. For all $x \in X$ and $b, b' \in 2$,

$$\begin{array}{l} \emptyset^{Bool} = 0,\ \epsilon^{Bool} = 1,\ \_^{Bool}(x) = 0, \\ b|^{Bool}b' = b \vee b',\ b \cdot^{Bool} b' = b \wedge b',\ star^{Bool}(b) = 1. \end{array}$$ ❑

Let $\Sigma = (S, F, R)$ be a signature and A be a Σ-algebra. An S-sorted subset inv of A is a **Σ-invariant** or **-subalgebra** of A if inv is **compatible** with all $f : e \to e' \in F$, i.e. $f^A(F_e(inv)) \subseteq F_{e'}(inv)$. $inc : inv \to A$ denotes the injective S-sorted **inclusion function** that maps a to a. inv can be extended to a Σ-algebra: For all $f : e \to e' \in F$, $f^{inv} = f^A \circ F_e(inc)$, and for all $r : e \in R$, $r^{inv} r^A \cap F_e(inv)$. Given an S-sorted subset B of A, the least Σ-invariant including B is denoted by $\langle B \rangle$.

An S-sorted relation $\sim \subseteq A^2$ is a **Σ-congruence** if $\sim$ is **compatible** with all $f : e \to e' \in F$, i.e. $(f^A \times f^A)(Rel_e(\sim)) \subseteq Rel_{e'}(\sim)$. $\sim^{eq}$ denotes the equivalence closure of $\sim$. $A_\sim$ denotes the Σ-algebra that agrees with A except for the interpretation of all $r : e \in R$: $r^{A_\sim} = \{b \in F_e(A) \mid \exists\ a \in r^A : a \sim^{eq} b\}$. $A/\sim$ denotes the S-sorted quotient set $\{[a]_\sim \mid a \in A\}$ where $[a]_\sim = \{b \in A \mid a \sim^{eq} b\}$. $nat_\sim : A \to A/\sim$ denotes the surjective S-sorted **natural function** that maps $a \in A$ to $[a]_\sim$. $A/\sim$ can be extended to a Σ-algebra: For all $f : e \to e' \in F$, $f^{A/\sim} \circ F_e(nat_\sim) = F_{e'}(nat_\sim) \circ f^A$. For all $r : e \in R$, $r^{A/\sim} = \{F_e(nat_\sim)(a) \mid a \in r^{A_\sim}\}$.

Let $h : A \to B$ be an S-sorted function. The S-sorted subset $img(h) = \{h(a) \mid a \in A\}$ of B is called the **image** of h. The S-sorted relation $ker(h) = \{(a, b \in A^2) \mid h(a) = h(b)\}$ is called the **kernel** of h.

Proposition 2.4. (1) inc and $nat_\sim$ are Σ-homomorphisms. $h : A \to B$ is surjective iff $img(h) = B$. h is injective iff $ker(h) = \langle id_A, id_A \rangle(A)$.

(2) A is a Σ-algebra and h is a Σ-homomorphism iff $ker(h)$ is a Σ-congruence. B is a Σ-algebra and h is a Σ-homomorphism iff $img(h)$ is a Σ-invariant. ❑

Homomorphism Theorem 2.5. h is a Σ-homomorphism iff there is a unique surjective Σ-homomorphism $h' : A \to img(h)$ with $inc \circ h' = h$ iff there is a unique injective Σ-homomorphism $h' : A/ker(h) \to B$ with $h' \circ nat_{ker(h)} = h$.❑

Example 2.6. Given a *behavior* function $f : X^* \to Y$, the *minimal realization* of f coincides with the invariant $\langle f \rangle$ of the following $DetAut(X, Y)$-algebra *MinAut*: $MinAut_{state} = (X^* \to Y)$ and for all $f : X^* \to Y$ and $x \in X$, $\delta^{MinAut}(f)(x) = \lambda w.f(xw)$ and $\beta^{MinAut}(f) = f(\epsilon)$.

Let $Y = 2$. Then behaviors $f : X^* \to Y$ coincide with languages over X, i.e. subsets L of X^*, and $\langle L \rangle$ is $DetAut(X, 2)$-isomorphic to the minimal acceptor of L with $\{L \subseteq X^* \mid \epsilon \in L\}$ as the set of final states. Hence the *state*-carrier of $MinAut$ agrees with the *reg*-carrier of $Lang$ (see Ex. 2.3). $T_{Reg(X)}$ also provides acceptors of regular languages, i.e., $T = T_{Reg(X)}$ is a $DetAut(X, 2)$-algebra. Its transition function $\delta^T : T \to T^X$ is called a *derivative* function. It has been shown that for all regular expressions R, $\langle R \rangle \subseteq T_{Reg(X)}$ has only finitely many states ([13], Thm. 4.3 (a); [41], Section 5; [26], Lemma 8). If combined with coinductive proofs of state equivalence (see Section 4), the stepwise construction of the least invariant $\langle R \rangle$ of $T_{Reg(X)}$ can be transformed into a stepwise construction of the least invariant $\langle L(R) \rangle$ of $MinAut = Lang$, thus leading to an efficient construction of the minimal acceptor that avoids the usual detour via powerset construction and minimization (see [44], Section 4).

Let $X = 1$. Then $MinAut$ is $DetAut(1, Y)$-isomorphic to the algebra of **streams over** Y: $MinAut_{state} = Y^{1^*} \cong Y^{\mathbb{N}}$. For all $s \in Y^{\mathbb{N}}$, $\beta(s) = s(0)$ and $\delta(s)(*) = \lambda n.s(n + 1)$.

Let $X = 2$. Then $MinAut$ represents the set of **infinite binary trees** with node labels from Y: $MinAut_{state} = X^{2^*}$. For all $t \in X^{2^*}$ and $b \in 2$, $\beta(t) = s(\epsilon)$, $\delta(t)(b) = \lambda w.t(bw)$. ❏

3 Σ-Algebras and F-Algebras

Let $\mathcal{K}$ be a category and F be an endofunctor on $\mathcal{K}$.

An F**-algebra** or F**-dynamics** is a $\mathcal{K}$-morphism $\alpha : F(A) \to A$. Alg_F denotes the category whose objects are the F-algebras and whose morphisms from $\alpha : F(A) \to A$ to $\beta : F(B) \to B$ are the $\mathcal{K}$-morphisms $h : A \to B$ with $h \circ \alpha = \beta \circ F(h)$. Hence α is initial in Alg_F if for all F-algebras β there is unique Alg_F-morphism h from α to β. h is called a **catamorphism** and to be **defined by recursion**.

An F**-coalgebra** or F**-codynamics** is a $\mathcal{K}$-morphism $\alpha : A \to F(A)$. $coAlg_F$ denotes the category whose objects are the F-coalgebras and whose morphisms from $\alpha : A \to F(A)$ to $\beta : B \to F(B)$ are the $\mathcal{K}$-morphisms $h : A \to B$ with $F(h) \circ \alpha = \beta \circ h$. Hence α is final in $coAlg_F$ if for all F-coalgebras β there is unique $coAlg_F$-morphism h from β to α. h is called an **anamorphism** and to be **defined by corecursion**.

Theorem 3.1. ([28], Lemma 2.2; [10], Prop. 5.12; [7], Section 2; [40], Thm. 9.1) Initial F-algebras and final F-coalgebras are isomorphisms in $\mathcal{K}$. ❏

In other words, the object A of an initial F-algebra $\alpha : F(A) \to A$ or a final F-coalgebra $\alpha : A \to F(A)$ is a *fixpoint* of F, i.e., A *solves* the equation $F(A) = A$.

Let $\Sigma = (S, F, R)$ be a signature. Σ induces an endofunctor H_Σ on Set^S (notation follows [1]): For all S-sorted sets and functions A and $s \in S$,

$$H_\Sigma(A)_s = \begin{cases} \coprod_{f:e \to s \in F} F_e(A) & \text{if } \Sigma \text{ is constructive,} \\ \prod_{f:s \to e \in F} F_e(A) & \text{if } \Sigma \text{ is destructive.} \end{cases}$$

Example 3.2. (see Exs. 2.1 and 2.2) Let A be an S-sorted set.

$$\begin{array}{ll}
H_{Nat}(A)_{nat} & = H_{CoNat}(A)_{nat} = 1 + A_{nat}, \\
H_{List(X)}(A)_{list} & = H_{CoList(X)}(A)_{list} = 1 + (X \times A_{list}), \\
H_{Reg(X)}(A)_{reg} & = 1 + 1 + X + A_{reg}^2 + A_{reg}^2 + A_{reg}, \\
H_{DetAut(X,Y)}(A)_{state} & = A_{state}^X \times Y, \\
H_{NDAut(X,Y)}(A)_{state} & = \mathcal{P}_{fin}(A_{state})^X \times Y, \\
H_{Tree(X,Y)}(A)_{tree} & = H_{CoTree(X,Y)}(A)_{tree} = X \times A_{trees}, \\
H_{Tree(X,Y)}(A)_{trees} & = H_{CoTree(X,Y)}(A)_{trees} = 1 + (X \times A_{tree} \times A_{trees}), \\
H_{BagTree(X,Y)}(A)_{tree} & = Y \times \mathcal{B}_{fin}(X \times A_{tree}),
\end{array}$$

$$\begin{array}{l}
H_{FDTree(X,Y)}(A)_{tree} = Y \times ((X \times A_{tree})^{\mathbb{N}} + word(X \times A_{tree})), \\
H_{FBTree(X,Y)}(A)_{tree} = Y \times word(X \times A_{tree}).
\end{array}$$ ❏

Given a constructive signature Σ, an H_Σ-algebra $H_\Sigma(A) \xrightarrow{\alpha} A$ is an S-sorted function and uniquely corresponds to a Σ-algebra A: For all $s \in S$ and $f : e \to s \in F$,

$$\begin{tikzcd}
H_\Sigma(A)_s \arrow[r, "\alpha_s = {[f^A]}_{f:e\to s\in F}"] & A_s \\
F_e(A) \arrow[u, "\iota_f"] \arrow[ur, "f^A = \alpha_s \circ \iota_f"'] \arrow[ur, phantom, "=", near start, shift left=4ex] &
\end{tikzcd}$$

Given a destructive signature Σ, an H_Σ-coalgebra $A \xrightarrow{\alpha} H_\Sigma(A)$ is an S-sorted function and uniquely corresponds to a Σ-algebra A: For all $s \in S$ and $f : s \to e \in F$,

$$\begin{tikzcd}
A_s \arrow[r, "\alpha_s = \langle f^A \rangle_{f:s\to e\in F}"] \arrow[dr, "f^A = \pi_f \circ \alpha_s"'] & H_\Sigma(A)_s \arrow[d, "\pi_f"] \\
& F_e(A)
\end{tikzcd}$$

α_s combines all s-constructors resp. -destructors into a single one.

An **ascending $\mathcal{K}$-chain** is a diagram sending the category $\{n \to n+1 \mid n \in \mathbb{N}\}$ to $\mathcal{K}$. $\mathcal{K}$ is **ω-cocomplete** if the empty diagram and each ascending $\mathcal{K}$-chain has a colimit. A **descending $\mathcal{K}$-chain** is a diagram sending the category $\{n \leftarrow n+1 \mid n \in \mathbb{N}\}$ to $\mathcal{K}$. $\mathcal{K}$ is **ω-complete** if the empty diagram and each descending $\mathcal{K}$-chain has a limit.

Let $\mathcal{K}$ and $\mathcal{L}$ be ω-cocomplete. A functor $F : \mathcal{K} \to \mathcal{L}$ is **ω-cocontinuous** if for all ascending $\mathcal{K}$-chains $\mathcal{D}$ and colimits $\{\mu_n : \mathcal{D}(n) \to C \mid n \in \mathbb{N}\}$ of $\mathcal{D}$, $\{F(\mu_n) \mid n \in \mathbb{N}\}$ is a colimit of $F \circ \mathcal{D}$.

Let $\mathcal{K}$ and $\mathcal{L}$ be ω-complete. A functor $F : \mathcal{K} \to \mathcal{L}$ is **ω-continuous** if for all descending $\mathcal{K}$-chains $\mathcal{D}$ and limits $\{\nu_n : C \to \mathcal{D}(n) \to C \mid n \in \mathbb{N}\}$ of $\mathcal{D}$, $\{F(\nu_n) \mid n \in \mathbb{N}\}$ is a limit of $F \circ \mathcal{D}$.

Theorem 3.3. ([7], Section 2; [29], Thm. 2.1) Let $\mathcal{K}$ be ω-cocomplete, $F : \mathcal{K} \to \mathcal{K}$ be an ω-cocontinuous functor, I be initial in $\mathcal{K}$, ini be the unique $\mathcal{K}$-morphism from I to $F(I)$ and A be the target of the colimit of the ascending $\mathcal{K}$-chain $\mathcal{D}$ defined as follows:

$$n \to n+1 \quad \mapsto \quad F^n(I) \overset{F^n(ini)}{\to} F^{n+1}(I).$$

Since F is ω-cocontinuous, $F(A)$ is the target of the colimit of $F \circ \mathcal{D}$. Hence there is a unique $\mathcal{K}$-morphism $ini'(F) : F(A) \to A$, which can be shown to be an initial F-algebra.

Let $\mathcal{K}$ be ω-complete, $F : \mathcal{K} \to \mathcal{K}$ be an ω-continuous functor, T be final in $\mathcal{K}$, fin be the unique $\mathcal{K}$-morphism from $F(T)$ to T and A be the source of the limit of the descending $\mathcal{K}$-chain $\mathcal{D}$ defined as follows:

$$n \leftarrow n+1 \quad \mapsto \quad F^n(T) \overset{F^n(fin)}{\leftarrow} F^{n+1}(T).$$

Since F is ω-continuous, $F(A)$ is the source of the limit of $F \circ \mathcal{D}$. Hence there is a unique $\mathcal{K}$-morphism $fin' : A \to F(A)$, which can be shown to be a final F-coalgebra. ❑

Theorem 3.4. (folklore) *Set* and Set^S are ω-complete and ω-cocomplete. ❑

Theorem 3.5. *For all polynomial types e over S, $F_e : Set^S \to Set$ is ω-continuous. For all types e over S, F_e is ω-cocontinuous. Consequently, for all $E : S \to \mathbb{PT}(S)$, $F_E : Set^S \to Set^S$ is ω-continuous, and for all $E : S \to \mathbb{BT}(S)$, F_E is ω-cocontinuous (see Section 2).*

Proof. By [8], Thm. 1 and 4, or [11], Prop. 2.2 (1) and (2), ω-continuous or -cocontinuous functors are closed under all finite products and all coproducts. Moreover, by [11], Prop. 2.2 (3), they are also closed under quotients of ω-continuous or -cocontinuous functors modulo a finite equivalence relation. Since for all $e \in \mathbb{BT}(S)$ and $A \in Set^S$, $F_{word(e)}(A) \cong \coprod_{n\in\mathbb{N}} F_e(A)^n$ and $F_{bag(e)}(A) \cong \coprod_{n\in\mathbb{N}} F_e(A)^n/{\sim_n}$ where $a \sim_n b$ iff a is a permutation of b, we conclude that for all $e \in \mathbb{PT}(S)$, F_e is ω-continuous and -cocontinuous. Since $\mathcal{P}_{fin}$ is ω-cocontinuous (see [4], Ex. 2.2.13) and the composition of ω-co/continuous functors is ω-co/continuous, we conclude that for all $e \in \mathbb{BT}(S)$, F_e is ω-cocontinuous. A proof for the fact that $\mathcal{P}_{fin}$ is not ω-continuous is given by [4], Ex. 2.3.11. ❑

Define $E(\Sigma) : S \to \mathbb{BT}(S)$ as follows: For all $s \in S$,

$$E(\Sigma)(s) = \begin{cases} dom(f_1) + \ldots + dom(f_n) & \text{if } \{f_1,\ldots,f_n\} = \{f \in F \mid ran(f) = s\} \\ & \quad \text{and } \Sigma \text{ is constructive,} \\ ran(f_1) \times \ldots \times ran(f_n) & \text{if } \{f_1,\ldots,f_n\} = \{f \in F \mid dom(f) = s\} \\ & \quad \text{and } \Sigma \text{ is destructive.} \end{cases}$$

Obviously, the endofunctor $F_{E(\Sigma)}$ agrees with H_Σ. Hence by Thm. 3.5, if Σ is constructive, then there is an initial Σ-algebra, if Σ is destructive and polynomial, then there is a final Σ-algebra, and both algebras are represented by co/limits of ascending resp. descending Set^S-chains:

Theorem 3.6. Let Σ be a constructive signature, I be the S-sorted set with $I_s = \emptyset$ for all $s \in S$, ini be the unique S-sorted function from I to $H_\Sigma(I)$ and $\sim_s$ is the equivalence closure of $\{(a, H_\Sigma^n(ini)(a)) \mid a \in H_\Sigma^n(I)_s,\ n \in \mathbb{N}\}$. By Thm. 3.3, the following Σ-algebra A is initial: For all $s \in S$ and $f : e \to s \in F$,

$$A_s = (\coprod_{n\in\mathbb{N}} H_\Sigma^n(I)_s)/{\sim_s} \quad \text{and} \quad f^A = ini'(H_\Sigma) \circ \iota_f.$$

Let B be a Σ-algebra, β_0 be the unique S-sorted function from I to B and for all $n \in \mathbb{N}$ and $s \in S$, $\beta_{n+1,s} = [f^B \circ F_e(\beta_{n,s})]_{f:e\to s\in F} : H_\Sigma^{n+1}(I)_s \to B_s$. The unique Σ-homomorphism $fold^B : A \to B$ is the unique S-sorted function satisfying $fold^B \circ nat_\sim = [\beta_n]_{n\in\mathbb{N}}$. ❏

Theorem 3.7. If Σ admits terms, then T_Σ is an initial Σ-algebra and for all Σ-algebras A, $fold^A : T_\Sigma \to A$ agrees with **term evaluation in** A, $eval^A$: For all $f : e \to s \in F$ and $t \in F_e(T_\Sigma)$, $eval^A(ft) = f^A(F_e(eval^A)(t))$. ❏

Let $G = (S, Z, BS, P)$ be a context-free grammar (see Section 2) and $Y = \cup_{X\in BS} X$. The following $\Sigma(G)$-algebra is called the **word algebra of** G: For all $s \in S$, $Word(G)_s = Z^*$. For all $w_0, \dots, w_n \in Z^*$, $e_1, \dots, e_n \in S \cup BS$, $p = (s, w_0 s_1 w_1 \dots s_n w_n) \in P$ and $v \in F_{e_1\times\dots\times e_n}(Word(G)) \subseteq (Z \cup Y)^n$, $f_p^{Word(G)}(v) = w_0 v_1 w_1 \dots v_n w_n$. The **language** $L(G)$ **of** G is the S-sorted image of $T_{\Sigma(G)}$ under term evaluation in $Word(G)$: For all $s \in S$, $L(G)_s = \{eval^{Word(G)}(t) \mid t \in T_{\Sigma(G),s}\}$. $L(G)$ is also characterized as the least solution of the set $E(G)$ of equations between the left- and right-hand sides of the rules of G (with the non-terminals regarded as variables). If G is not left-recursive, then the solution is unique [39]. This provides a simple method of proving that a given language L agrees with $L(G)$: Just show that L solves $E(G)$.

Each parser for G can be presented as a function $parse_G : (Z \cup Y)^* \to M(T_{\Sigma(G)})$ where M is a *monadic functor* that embeds $T_{\Sigma(G)}$ into a larger set of possible results, including syntax errors and/or sets of syntax trees for realizing non-deterministic parsing [39]. The parser is correct if $parse_G \circ eval^{Word(G)} = \eta_{T_{\Sigma(G)}}$ (where η is the *unit* of M) and if all words of $(Z \cup Y)^* \setminus L(G)$ are mapped to error messages.

The most fascinating advantage of algebraic compiler construction is the fact that the same *generic* compiler can be used for translating $L(G)$ into an arbitrary target language formulated as a $\Sigma(G)$-algebra A. The respective instance $compile_G^A : (Z \cup Y)^* \to M(A)$ agrees with the composition $M(eval^A) \circ parse_G$. More efficiently than by first constructing a syntax tree and then evaluating it in A, $compile_G^A$ can be implemented as a slight modification of $parse_G$. Whenever the parser performs a reduction step w.r.t. a rule p of G by building the syntax tree $f_p(t_1, \dots, t_n)$ from already constructed trees $t_1, \dots, t_n$, the compiler derived from $parse_G$ applies the interpretation f_p in A to already computed elements $a_1, \dots, a_n$ of A and thus returns the target object $f_p^A(a_1, \dots, a_n)$ instead of the tree $f_p(t_1, \dots, t_n)$. Syntax trees need not be constructed at all!

Expressing the target language of a compiler for G as a $\Sigma(G)$-algebra *Target* also provides a method for proving that the compiler is correct w.r.t. the semantics $Sem(G)$ and $Sem(Target)$ of G resp. *Target*. The correctness amounts to the commutativity of the following diagram:

$$\begin{tikzcd}
T_{\Sigma(G)} \arrow[r, "eval^{Target}"] \arrow[d, "eval^{Sem(G)}"'] \arrow[dr, phantom, "(1)"] & Target \arrow[d, "execute"] \\
Sem(G) \arrow[r, "encode"] & Sem(Target)
\end{tikzcd}$$

Of course, $Sem(G)$ has to be a $\Sigma(G)$-algebra. $Sem(Target)$, however, usually refers to a signature different from $\Sigma(G)$. But the interpretations of the constructors of $\Sigma(G)$ in *Target* can often be transferred easily to $Sem(Target)$ such that the interpreter *execute* becomes a $\Sigma(G)$-homomorphism: For all $p \in P$, $f_p^{Sem(Target)} \circ execute = execute \circ f_p^{Target}$ is a definition of $f_p^{Sem(Target)}$ iff the kernel of *execute* is compatible with f_p^{Target}. Finally, we need a homomorphism *encode* that mirrors the (term) compiler $eval^{Target}$ on the semantical level. Finally, if all four functions of (1) are $\Sigma(G)$-homomorphisms, then the initiality of $T_{\Sigma(G)}$ in $Alg_{\Sigma(G)}$ implies that the diagram commutes!

Algebraic approaches to formal languages and compiler design are not new. They have been applied sucessfully to various programming languages (see, e.g., [18,45,32,12,46,31,36]). Hence it is quite surprising that they are more or less ignored in the currently hot area of document definition and query languages (XML and all that) – although structured data play a prominent rôle in such languages. Instead of associating these data with adequate co/algebraic types, XML theoreticians boil everything down to regular expressions, words and word recognizing automata.

Example 3.8. (cf. Exs. 2.1 and 2.4) $\mathbb{N}$ is an initial *Nat*-algebra: $0^{\mathbb{N}} = 0$ and for all $n \in \mathbb{N}$, $succ^{\mathbb{N}}(n) = n + 1$.

$T_{Reg(X)}$ is an initial $Reg(X)$-algebra. Hence $T_{Reg(X),reg}$ is the set of regular expressions over X. For all such expressions R, $fold^{Lang}(R) = eval^{Lang}(R)$ is the language of R and $eval^{Bool}(R)$ checks it for inclusion of the empty word.

For $\Sigma \in \{List(X), Tree(X,Y), BagTree(X,Y), FDTree(X,Y)\}$, the elements of the *list*- resp. *tree*-carrier of an initial Σ-algebra can be represented by the sequences resp. trees that Ex. 2.1 associates with Σ. ❑

We proceed with to the destructor analogue of Thm. 3.6:

Theorem 3.9. Let Σ be a polynomial destructive signature, T be the S-sorted set with $T_s = 1$ for all $s \in S$ and *fin* be the unique S-sorted function from $H_\Sigma(T)$ to T. By Thm. 3.3, the following Σ-algebra A is final: For all $s \in S$ and $f : s \to e \in F$,

$$A_s = \{a \in \prod_{n\in\mathbb{N}} H_\Sigma^n(T)_s \mid \forall\, n \in \mathbb{N} : a_n = H_\Sigma^n(fin)(a_{n+1})\} \quad \text{and} \quad f^A = fin'(H_\Sigma)\circ\pi_f.$$

Let B be a Σ-algebra, β_0 be the unique S-sorted function from B to T and for all $n \in \mathbb{N}$ and $s \in S$, $\beta_{n+1,s} = \langle F_e(\beta_{n,s}) \circ f^A \rangle_{f:s\to e\in F} : A_s \to H_\Sigma^{n+1}(T)_s$. The unique Σ-homomorphism $unfold^B : B \to A$ is the unique S-sorted function satisfying $inc \circ unfold^B = \langle \beta_n \rangle_{n\in\mathbb{N}}$. ❏

Example 3.10. (cf. Exs. 2.2 and 2.6) $A = \mathbb{N} \cup \{\infty\}$ is a final $CoNat$-algebra: For all $n \in A$,

$$pred^A(n) = \begin{cases} * & \text{if } n = 0, \\ n-1 & \textit{if } n > 0, \\ \infty & \textit{if } n = \infty. \end{cases}$$

$MinAut$ is a final $DetAut(X,Y)$-algebra, in particular, the $DetAut(1,Y)$-algebra of streams over Y is a final $DetAut(1,Y)$-algebra.

Since $T = T_{Reg(X)}$ and $Lang$ are $DetAut(X,2)$-algebras, $fold^{Lang} = eval^{Lang} : T \to Lang$ is a $DetAut(X,2)$-homomorphism (see [39], Section 12) and $Lang$ is a final $DetAut(X,2)$-algebra, $fold^{Lang}$ coincides with $unfold^T$. This allows us to extend T to a generic parser, not only for regular languages, but also for context-free ones (see [39], Sections 12 and 14).

For $\Sigma \in \{CoList(X), CoTree(X,Y), FBTree(X,Y)\}$, the elements of the *list*- resp. *tree*-carrier of a final Σ-algebra can be represented by the sequences resp. trees that Ex. 2.2 associates with Σ. ❏

The construction of $CoNat$, $CoList$ and $CoTree$ from Nat, $List$ resp. $Tree$ is not accidental. Let $\Sigma = (S,F,R)$ be a constructive signature and A be a Σ-algebra. Σ induces a destructive signature $Co\Sigma$: Since for all $s \in S$, $H_\Sigma(A)_s = \coprod_{f:e\to s\in F} F_e(A)$, and since by Thm. 3.1, the initial H_Σ-algebra $[f^A]_{f:e\to s\in F}$ is an isomorphism, ini^{-1} is both a H_Σ-algebra and a $H_{Co\Sigma}$-coalgebra where

$$Co\Sigma = (S, \{d_s : s \to \coprod_{f:e\to s\in F} e \mid s \in S\}, R).$$

The final $Co\Sigma$-algebra is a *completion* of the initial Σ-algebra in the sense of [2]. Its carriers consist of finitely branching trees such that each node is labelled with a base set or a constructor of Σ (cf. [18], Section 4; [6], Section II.2):

Let BS be the set of base sets of Σ and $Y = \cup_{X\in BS} X$. The $(BS \cup S)$-sorted set CT_Σ of **Σ-trees** consists of all partial functions $t : \mathbb{N}^* \to Y \cup F$ such that for all $s \in BS$, $CT_{\Sigma,s} = Y_s$ and for all $s \in S$, $t \in CT_{\Sigma,s}$ iff for all $w \in \mathbb{N}^*$,

- $t(\epsilon) \in F \wedge ran(t(\epsilon)) = s$,
- if $t(w) : e_1 \times \ldots \times e_n \to s' \in F$, then for all $0 \leq i \leq n$, $t(wi) \in Y_{e_i}$ or $t(wi) \in F$ and $ran(t(wi)) = s'$.

CT_Σ is both a Σ- and a $Co\Sigma$-algebra: For all $f : e \to s \in F$, $t \in F_e(CT_\Sigma)$ and $w \in \mathbb{N}^*$,

$$f^{CT_\Sigma}(t)(w) = \begin{cases} f & \text{if } w = \epsilon, \\ \pi_i(t)(v) & \text{if } \exists\, i \in \mathbb{N}, v \in \mathbb{N}^* : w = iv. \end{cases}$$

For all $s \in S$ and $t \in CT_{\Sigma,s}$,

$$d_s^{CT_\Sigma}(t) = ((\lambda w.t(0w), \ldots, \lambda w.t((|dom(t(\epsilon))| - 1)w)), t(\epsilon)) \in \coprod_{f:e\to s\in F} F_e(CT_\Sigma).$$

Moreover, CT_Σ is a(n S-sorted) complete partial order (ω-complete $\sqcup$-semilattice) – provided that Σ is **inhabited** or grounded, i.e., for all $s \in S$, $T_{\Sigma,s} \neq \emptyset$ ($T_{\Sigma,s}$ needs a least element!). A Σ-algebra A is **ω-continuous** if its carriers are complete partial orders and if for all $f \in F$, f^A is ω-continuous. ωAlg_Σ denotes the subcategory of Alg_Σ that consists of all ω-continuous Σ-algebras as objects and all ω-continuous Σ-homomorphisms between them.

Theorem 3.11. If Σ is inhabited, then CT_Σ is initial in ωAlg_Σ. In any case, CT_Σ is a final $Co\Sigma$-algebra.

Proof. The first part follows from [18], Thm. 4.15, [11], Thm. 3.2, or [2], Prop. IV.2. Here is a proof of the second part: Let A be a $Co\Sigma$-algebra. An S-sorted function $h = unfold^A : A \to CT_\Sigma$ is defined as follows: For all $s \in S$, $a \in A_s$, $i \in \mathbb{N}$ and $w \in \mathbb{N}^*$, $d_s^A(a) = ((a_1,\dots,a_n),f)$ implies

$$\begin{aligned} h(a)(\epsilon) &= f,\\ h(a)(iw) &= \begin{cases} h(a_i)(w) & \text{if } 0 \le i < |dom(f)|,\\ \text{undefined otherwise,} \end{cases} \end{aligned}$$

in short: $h(a) = f(h(a_1),\dots,h(a_n))$. Let $s \in S$, $a \in A_s$ and $d_s^A(a) = ((a_1,\dots,a_n),f)$. Then

$$\begin{aligned} &d_s^{CT_\Sigma}(h(a)) = d_s^{CT_\Sigma}(f(h(a_1),\dots,h(a_n)))\\ &= ((h(a_1),\dots,h(a_n)),f) = h((a_1,\dots,a_n),f) = h(d_s^A(a)). \end{aligned}$$

Hence h is a $co\Sigma$-homomorphism. Conversely, let $h' : A \to CT_\Sigma$ be a $co\Sigma$-homomorphism. Then

$$\begin{aligned} &d_s^{CT_\Sigma}(h'(a)) = h'(d_s^A(a)) = h'((a_1,\dots,a_n),f) = ((h'(a_1),\dots,h'(a_n)),f)\\ &= d_s^{CT_\Sigma}(f(h'(a_1),\dots,h'(a_n))) \end{aligned}$$

and thus $h'(a) = f(h'(a_1),\dots,h'(a_n))$ because $d_s^{CT_\Sigma}$ is injective. We conclude that h' agrees with h. ❏

Another class of polynomial destructive signatures is obtained by dualizing constructive signatures that admit terms. A destructive signature $\Sigma = (S,F,R)$ **admits coterms** if for all $f \in F$ there are $e_1,\dots,e_n \in S \cup Set$ with $ran(f) = e_1 + \ldots + e_n$. If Σ admits terms, then the Σ-algebra coT_Σ of **Σ-coterms** is defined as follows:

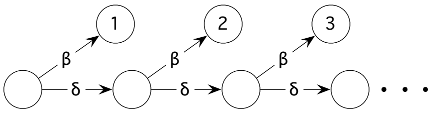
- For all $s \in S$, $coT_{\Sigma,s}$ is the greatest set of finitely branching trees t of infinite depth such that for all $f : s \to e_1 + \ldots + e_n \in F$, $n \in \mathbb{N}$, a unique arc a labelled with a pair (f,i), $1 \le i \le n$, emanates from the (unlabelled) root of t and either $e_i \in S$ and the target of a is in coT_{Σ,e_i} or e_i is a base set and the target of a is leaf labelled with an element of e_i.
- For all $f : s \to e_1 + \ldots + e_n \in F$ and $t \in coT_{\Sigma,s}$, $f^{coT_\Sigma}(t)$ is the tree where the edge emanating from the root of t and labelled with (f,i) for some i points to.

Again, the interpretation of R in T_Σ is not fixed.

Theorem 3.12. If Σ admits coterms, then coT_Σ is an final Σ-algebra and for all Σ-algebras A, $unfold^A : A \to coT_\Sigma$ agrees with **coterm evaluation**

in A, *coeval*^{A}: Let $s \in S$, $a \in A_s$, $f : s \to e \in F$ and $f^A(a) = (b_f, i_f)$. $\{(f, i_f) \mid f : s \to e \in F\}$ is the set of labels of the arcs emanating from the root of $coeval^A(a)$ and for all $f : s \to e \in F$, the outarc labelled with (f, i) points to $coeval^A(b_f)$. ❏

Example 3.13. (cf. Exs. 2.2 and 3.10) Since $DetAut(1, \mathbb{N})$ admits coterms (if $state^X$ is replaced by *state*), $coT_{DetAut(1,\mathbb{N})}$ is a final $DetAut(1, \mathbb{N})$-algebra. For instance, the stream $[1, 2, 3, \ldots]$ is represented in $coT_{DetAut(1,\mathbb{N})}$ as the following infinite tree:



We omitted the number component of the edge labels because it is always 1. ❏

Of course, the construction of a destructive signature from a constructive one can be reversed: Let $\Sigma = (S, F, OP)$ be a destructive signature. Then

$$Co\Sigma = (S, \{c_s : \prod_{f:s\to e\in F} e \to s \mid s \in S\}, R)$$

is a constructive signature.

It remains to supply a construction for final Σ-algebras for *non-polynomial* destructive signatures where the range of some destructor involves the finite-set constructor *set*.

Given an S-sorted set M, a signature Σ is M**-bounded** if for all Σ-algebras A, $s \in S$ and $a \in A_s$, $|\langle a\rangle_s| \le |M_s|$.

By [5], Thm. 4.1, boundedness is equivalent to a kind of cocontinuity, namely accessibility, i.e., preservation of colimits of κ-chains where κ is a regular cardinal.

Example 3.14. (cf. Ex. 2.2) By [40], Ex. 6.8.2, or [19], Lemma 4.2, $H_{DetAut(X,Y)}$ is X^*-bounded: For all $DetAut(X, Y)$-algebras A and $a \in A_{state}$,

$$\langle st\rangle = \{\delta^{A*}(a)(w),\ w \in X^*\}$$

where $\delta^{A*}(a)(\epsilon) = st$ and $\delta^{A*}(a)(xw) = \delta^{A*}(\delta^A(a)(x))(w)$ for all $x \in X$ and $w \in X^*$. Hence $|\langle st\rangle| \le |X^*|$. ❏

Example 3.15. (cf. Ex. 2.2) $H_{NDAut(X,Y)}$ is $(X^* \times \mathbb{N})$-bounded: For all *NDAut*-algebras A and $a \in A_{state}$, $\langle st\rangle = \cup\{\delta^{A*}(a)(w),\ w \in X^*\}$ where $a \in A_{state}$, $\delta^{A*}(a)(\epsilon) = \{st\}$ and $\delta^{A*}(a)(xw) = \cup\{\delta^{A*}(st')(w) \mid st' \in \delta^A(a)(x)\}$ for all $x \in X$ and $w \in X^*$. Since for all $a \in A_{state}$ and $x \in X$, $|\delta^A(a)(x)| \in \mathbb{N}$, $|\langle st\rangle| \le |X^* \times \mathbb{N}|$. If $X = 1$, then $X^* \times \mathbb{N} \cong \mathbb{N}$ and thus $H_{NDAut(1,Y)}$ is $\mathbb{N}$-bounded (see [40], Ex. 6.8.1; [19], Section 5.1). ❏

Theorem 3.16. ([40], Thm. 10.6; [19], Cor. 4.9 and Section 5.1) All signatures are bounded. ❏

Lemma 3.17. Let X be an S-sorted set and $\Sigma = (S, F, R)$ be a destructive signature such that for all $f : s \to e \in F$, $e = s^{X_s}$ or e is a base set. Σ is polynomial and thus by Thm. 3.8, Alg_Σ has a final object A. If $|S| = 1$, then Σ agrees with $DetAut(X,Y)$ and thus A is $DetAut(X,Y)$-isomorphic to $MinAut$ (see Exs. 2.6 and 3.9). Otherwise A can be constructed as a straightforward extension of $MinAut$ to several sorts: For all $s \in S$, $A_s = (X_s^* \to \prod_{g:s\to Z\in F} Z)$, and for all $f : s \to s^{X_s}, g : s \to Z \in F$ and $h \in A_s$, $f^A(h) = \lambda x.\lambda w.h(xw)$ and $g^A(h) = \pi_g(h(\epsilon))$.

A can be visualized as the S-sorted set of trees of infinite depth such that for all $s \in S$ and $h \in A_s$, h is $|X_s|$-branching and for all $f : s \to s^{X_s}, g : s \to Z \in F$ and $x \in X_s$, each node of h is labelled with an element of Z, $f^A(h)(x) = \lambda w.h(xw)$ is the subtree of h where the x-th outarc of the root r of h points to and $g^A(h)$ is the Z-label of r. ❑

Theorem 3.18. Let M be an S-sorted set, $\Sigma = (S,F,R)$ be an M-bounded destructive signature, $F' = \{f_s : s \to s^{M_s} \mid s \in S\} \cup \{f' : s \to F_e(M) \mid f : s \to e \in F\}$, $\Sigma' = (S,F',R)$ and $\eta : H_{\Sigma'} \to H_\Sigma$ be the function defined as follows: For all S-sorted sets A, $f : s \to e \in F$ and $a \in H_{\Sigma'}(M)_s$, $\pi_f(\eta_{A,s}(a)) = F_e(\pi_{f_s}(a))(\pi_{f'}(a))$. η is a surjective natural transformation.

Proof. The theorem is an adaption of [19], Thm. 4.7 (i)⇒(iv), and the definitions in its proof to our many-sorted syntax. ❑

Lemma 3.19. Let $\Sigma = (S,F,R)$ and $\Sigma' = (S,F',R)$ be destructive signatures and $\eta : H_{\Sigma'} \to H_\Sigma$ be a surjective natural transformation. The following Σ-algebra B is **weakly final** (i.e., the Σ-homomorphisms emanating from B are not unique): For all $s \in S$, $B_s = A_s$, and for all $f : s \to e \in F$, $f^B = \pi_f \circ \eta_{A,s} \circ \langle g_1, \dots, g_n\rangle$ where $\{g_1, \dots, g_n\} = \{g^A \mid g : s \to e \in F'\}$. Moreover, $B/\!\sim$ is a final Σ-algebra where $\sim$ is the greatest Σ-congruence on B, i.e. the union of all Σ-congruence on B. ❑

Proof. The lemma is an adaption of [19], Lemma 2.3 (iv), and the definitions in its proof to our many-sorted syntax. ❑

Given an arbitrary destructive signature Σ, the previous results lead to a construction of the final Σ-algebra – provided that the bound is known:

Theorem 3.20. Let M be an S-sorted set, $\Sigma = (S,F,R)$ be an M-bounded destructive signature and $F' = \{f' : s \to F_e(M) \mid f : s \to e \in F\}$. The following Σ-algebra C is weakly final: For all $s \in S$, $C_s = (M_s^* \to \prod_{f':s\to Z\in F'} Z)$, and for all $f : s \to e \in F$ and $h \in C_s$, $f^C(h) = F_e(\lambda x.\lambda w.h(xw))(\pi_{f'}(h(\epsilon)))$. $C/\!\sim$ is a final Σ-algebra where $\sim$ is the greatest Σ-congruence on C, i.e. the greatest S-sorted binary relation on C such that for all $f : s \to e \in F$ and $h, h' \in C_s$,

$$h \sim h' \quad\Rightarrow\quad f^C(h)\ Rel_e(\sim)\ f^C(h').$$

Proof. Let $\Sigma' = (S, F' \cup \{f_s : s \to s^{M_s} \mid s \in S\}, R)$. By Thm. 3.18, $\eta : H_{\Sigma'} \to H_\Sigma$ with $\pi_f(\eta_{A,s}(a)) = F_e(\pi_{f_s}(a))(\pi_{f'}(a))$ for all $A \in Set^S$, $f : s \to e \in F$ and $a \in H_{\Sigma'}(M)_s$, is a surjective natural transformation. By Lemma 3.17, the following Σ'-algebra is final: For all $s \in S$, $A_s = (M_s^* \to \prod_{f':s\to Z\in F'} Z)$, and for all $h \in A_s$ and $f' : s \to Z \in F'$, $f_s^A(h) = \lambda m.\lambda w.h(mw)$ and $f'^A(h) = \pi_{f'}(h(\epsilon))$.

Let $\{g_1,\dots,g_n\}=\{f^A\mid f:s\to e\in F'\}$, By Lemma 3.19, the following Σ-algebra B is weakly final: For all $s\in S$, $B_s=A_s$, and for all $f:s\to e\in F$, $f^B=\pi_f\circ\eta_{A,s}\circ\langle g_1,\dots,g_n\rangle$. Hence for all $f:s\to e\in F$ and $h\in B_s$,

$$\begin{aligned}&f^B(h)=\pi_f(\eta_{A,s}(\langle g_1,\dots,g_n\rangle(h)))=\pi_f(\eta_{A,s}(g_1(h),\dots,g_n(h)))\\&=F_e(\pi_{f_s}(g_1(h),\dots,g_n(h)))(\pi_{f'}(g_1(h),\dots,g_n(h)))=F_e(f_s^A(h))(f'^A(h))\\&=F_e(\lambda x.\lambda w.h(xw))(\pi_{f'}(h(\epsilon)))=f^C(h).\end{aligned}$$

We conclude that $C=B$ is weakly final and $C/{\sim}$ is final in Alg_Σ. ❑

Example 3.21. Let $M=M_{state}=X^*\times\mathbb{N}$, $Z_1=F_{set(state)^X}(M)=\mathcal{P}_{fin}(M)^X$, $Z_2=F_Y(M)=Y$ and $F'=\{\delta':state\to Z_1,\beta':state\to Z_2\}$. By Ex. 3.11, $H_{NDAut(X,Y)}$ is M-bounded. Hence by Thm. 3.17, the following $NDAut(X,Y)$-algebra C is weakly final: $C_{state}=(M^*\to Z_1\times Z_2)$ and for all $h\in C_{state}$ and $x\in X$, $h(\epsilon)=(g,y)$ implies

$$\begin{aligned}\delta^C(h)(x)&=F_{set(state)^X}(\lambda m.\lambda w.h(mw))(\pi_{\delta'}(h(\epsilon)))(x)\\&=F_{set(state)^X}(\lambda m.\lambda w.h(mw))(g)(x)=F_{set(state)}(\lambda m.\lambda w.h(mw))(g(x))\\&=\{F_{state}(\lambda m.\lambda w.h(mw))(m)\mid m\in g(x)\}\\&=\{\lambda m.\lambda w.h(mw))(m)\mid m\in g(x)\}=\{\lambda w.h(mw)\mid m\in g(x)\},\\\beta^C(h)&=F_Y(\lambda x.\lambda w.h(xw))(\pi_{\beta'}(h(\epsilon)))=F_Y(\lambda x.\lambda w.h(xw))(y)=id_Y(y)=y.\end{aligned}$$

Moroever, $C/{\sim}$ is a final Σ-algebra where $\sim$ is the greatest binary relation on C such that for all $h,h'\in C_{state}$,

$$h\sim h'\quad\Rightarrow\quad\delta^C(h)\ Rel_{set(state)^X}(\sim)\ \delta^C(h')\ \wedge\ \beta^C(h)\ Rel_Y(\sim)\ \beta^C(h'),$$

i.e., for all $x\in X$, $h\sim h'$, $h(\epsilon)=(g,y)$ and $h'(\epsilon)=(g',y')$ imply

$$\begin{aligned}&\forall\ m\in g(x)\ \exists n\in g'(x):\lambda w.h(mw)\sim\lambda w.h'(nw)\ \wedge\\&\forall\ n\in g'(x)\ \exists m\in g(x):\lambda w.h(mw)\sim\lambda w.h'(nw)\ \wedge\ y=y'.\end{aligned}$$

Let $NDAut(X,Y)'=(\{state\},F'\cup\{f_{state}:state\to state^M\},\emptyset)$. By the proof of Thm. 3.20, C is constructed from the $NDAut(X,Y)'$-algebra A with $A_{state}=C_{state}$ and for all $h\in A_{state}$, $f^A_{state}(h)=\lambda m.\lambda w.h(mw)$ and $\langle\delta'^A,\beta'^A\rangle(h)=h(\epsilon)$. Hence by Lemma 3.17, C_{state} can be visualized as the set of $|M|$-branching trees h of infinite depth such that for all $m\in M$, each node of h is labelled with an element of $\mathcal{P}_{fin}(M)^X\times Y$, $\lambda w.h(mw)$ is the subtree of h where the m-th outarc of the root r of h points to and $h(\epsilon)$ is the pair of labels of r. See [20], Section 5, for a description of $C/{\sim}$ in the case $X=Y=1$. ❑

4 Co/Induction, Abstraction, Restriction, Extension and Co/Recursion

After having shown in the previous sections how to build the domains of many-sorted initial or final models, let us turn to their analysis (by co/induction), the definition of functions on their domains (by co/recursion), their extension by further constructors resp. destructors, the factoring (abstraction) of initial models and the restriction of final one.

The dual operations of the last two, i.e., restriction of an initial model or abstraction of a final model, are impossible because an initial Σ-algebra has no Σ-invariants besides itself and a final Σ-algebra has no congruences besides the diagonal ([43], Thm. 4.3):

Lemma 4.1. (see Section 2) Let Σ be a constructive signature. (1) For all Σ-algebras A, $img(fold^A)$ is the least Σ-invariant of A. (2) If A is initial, then A is the only Σ-invariant of A.

Let Σ be a destructive signature. (3) For all Σ-algebras A, $ker(unfold^A)$ is the greatest Σ-congruence on A. (4) If A is final, then $\langle id_A, id_A\rangle(A)$ is the only Σ-congruence on A.

Proof of (2) and (4). Let Σ be constructive, A be initial and inv be a Σ-invariant of A. Then $inc \circ fold^{inv} = id_A$. Hence $inc \circ fold^{inv}$ and thus inc are surjective. We conclude that inv and A are Σ-isomorphic.

Let Σ be destructive, A be final and $\sim$ be a Σ-congruence on A. Then $unfold^{A/\sim} \circ nat_\sim = id_A$. Hence $unfold^{A/\sim} \circ nat_\sim$ and thus $nat_\sim$ are injective. We conclude that A and $A/\sim$ are Σ-isomorphic. ❏

By Lemma 4.1 (2) and (4), algebraic co/induction is sound:

Algebraic Induction. Let Σ be a constructive signature with sort set S, A be an initial Σ-algebra and p be an S-sorted subset of A. $p = A$ iff $inv \subseteq p$ for some Σ-invariant inv of A. ❏

Algebraic Coinduction. Let Σ be a destructive signature with sort set S, A be a final Σ-algebra and r be an S-sorted relation $r \subseteq A$. $r = \langle id_A, id_A\rangle(A)$ iff $r \subseteq\sim$ for some Σ-congruence $\sim$ on A. ❏

In practice, an inductive proof of $p = A$ starts with $inv := p$ and stepwise decreases inv as long as inv is not an invariant. In terms of the formula φ that represents inv, each modification of inv is a conjunctive extension – usually called a *generalization* – of φ. The goal $p = A$ means that A satisfies φ.

Dually, a coinductive proof of $r = \langle id_A, id_A\rangle(A)$ starts with $\sim := r$ and stepwise increases $\sim$ as long as $\sim$ is not a congruence. In terms of the formula φ that represents $\sim$, each modification of $\sim$ is a disjunctive extension of φ. The goal $r = \langle id_A, id_A\rangle(A)$ means that A satisfies the equations given by φ.

Example 4.2. (see Exs. 2.6 and 3.10) Let A be a $DetAut(X,2)$-algebra. $\sim\subseteq A^2$ is a $DetAut(X,2)$-congruence iff for all $a, b \in A_{state}$ and $x \in X$, $a \sim b$ implies $\delta^A(a)(x) \sim \delta^A(b)(x)$ and $\beta^A(a)(x) = \beta^A(b)(x)$. Since the algebra $T = T_{Reg(X)}$ of regular expressions and the algebra $Lang$ of languages over X is a final $DetAut(X,2)$-algebra, $Lang$ is final and $unfold^T$ agrees with $fold^{Lang} = eval^{Lang}$, two regular expressions R, R' have the same language ($=$ image under $eval^{Lang}$) iff for some $w \in X^*$, the regular expressions $\delta^{T*}(R)(w)$ and $\delta^{T*}(R')(w)$ (see Ex. 3.14) have the same language (since, e.g., they are rewritable into each other by applying basic properties of regular operators). It is easy to see how this way of proving language equality can also be used for constructing the minimal acceptor $\langle L\rangle$ of the language L of a regular expression. ❑

Algebraic co/induction is a special case of relational co/induction that applies to an arbitrary Σ-algebra A and least resp. greatest interpretations of relation symbols of Σ, which are axiomatized in terms of *co/Horn clauses* [33,34,37,38]:

Let $\Sigma = (S,F,R)$ be a signature and A be a Σ-algebra. A **Σ-formula** φ is a well-typed first-order formula built up from logical operators, symbols of $F \cup R$, liftings thereof (see Section 2) and elements of a fixed $\mathbb{BT}(S)$-sorted set Var of variables. The interpretation φ^A of φ in A is the set of $\mathbb{BT}(S)$-sorted *valuations* $f : Var \to A$ (where $A_e = F_e(A)$), which satisfy φ. The interpretation $t^A : A^{Var} \to A$ of a term t occurring in φ is the $\mathbb{BT}(S)$-sorted function that takes a valuation f and evaluates t in A under f. (For lack of space, we omit formal definitions here.) A Σ-formula $\varphi \Leftarrow \psi$ resp. $\varphi \Rightarrow \psi$ is called a **Σ-Horn clause** resp. **Σ-co-Horn clause** if φ is an atom(ic formula) and ψ is negation-free.

Let $\Sigma' = (S,F,\emptyset)$, C be a Σ'-algebra and $Alg_{\Sigma,C}$ be the category of all Σ-algebras A with $A|_{\Sigma'} = B$. $Alg_{\Sigma,C}$ is a *complete lattice*: For all $A,B \in Alg_{\Sigma,C}$, $A \le B \Leftrightarrow \forall\ r \in R : r^A \subseteq r^B$. For all $\mathcal{A} \subseteq Alg_{\Sigma,C}$ and $r : e \in R$, $r^\perp = \emptyset$, $r^\top = F_e(A)$, $r^{\sqcup(\mathcal{A})} = \bigcup_{A\in\mathcal{A}} r^A$ and $r^{\sqcap(\mathcal{A})} = \bigcap_{A\in\mathcal{A}} r^A$. Let $\Phi : Alg_{\Sigma,C} \to Alg_{\Sigma,C}$ be a monotone function. $A \in Alg_{\Sigma,C}$ is **Φ-closed** if $\Phi(A) \le A$. A is **Φ-dense** if $A \le \Phi(A)$. The well-known fixpoint theorem of Knaster and Tarski provides fixpoints of Φ:

Theorem 4.3. *$lfp(\Phi) = \sqcap\{A \in Alg_{\Sigma,C} \mid A$ is Φ-dense$\}$ is the least and $gfp(\Phi) = \sqcup\{A \in Alg_{\Sigma,C} \mid A$ is Φ-closed$\}$ is the greatest fixpoint of Φ.* ❑

Obviously, for all negation-free formulas φ and $A,B \in Alg_{\Sigma,C}$, $A \le B$ implies $\varphi^A \subseteq \varphi^B$. A set AX of Σ-formulas that consists of only Horn clauses or only co-Horn clauses induces a monotone function $\overline{AX,C} : Alg_{\Sigma,C} \to Alg_{\Sigma,C}$: For all $A \in Alg_{\Sigma,C}$ and $r : e \in R$, $r^{\overline{AX,C}}(A) = \{(t^A(f) \mid f \in \varphi^A,\ r(t) \Leftarrow \varphi \in AX\}$ if AX consists of Horn clauses and $r^{\overline{AX,C}}(A) = F_e(A) \setminus \{(t^A(f) \mid f \in A^{Var} \setminus \varphi^A,\ r(t) \Rightarrow \varphi \in AX\}$ if AX consists of co-Horn clauses. Hence by Thm. 4.3, $\overline{AX,C}$ has a least fixpoint $lfp = lfp(\overline{AX,C})$ and a greatest fixpoint $gfp = gfp(\overline{AX,C})$. In other words, lfp and gfp are the least resp. greatest $A \in Alg_{\Sigma,C}$ that satisfy AX, or, if we regard the relation symbols in AX as variables, then $\{r^{lfp} \mid r \in R\}$ is the least and $\{r^{gfp} \mid r \in R\}$ is the greatest solution of AX in R. This implies immediately that relational co/induction is sound:

Relational Induction. Let AX be set of Horn clauses. $lfp(\overline{AX,C})$ satisfies $r(x) \Rightarrow \psi(x)$ iff there is a formula $\psi'(x)$ such that for all $r(t) \Leftarrow \varphi \in AX$, $lfp(\overline{AX,C})$ satisfies $r(t) \Leftarrow \varphi'$ where φ' is obtained from φ by replacing all occurrences of atoms $r(u)$ with $\psi(u) \wedge \psi'(u)$. ❑

Relational Coinduction. Let AX be set of co-Horn clauses. $gfp(\overline{AX,C})$ satisfies $r(x) \Leftarrow \psi(x)$ iff there is a formula $\psi'(x)$ such that for all $r(t) \Rightarrow \varphi \in AX$, $gfp(\overline{AX,C})$ satisfies $r(t) \Rightarrow \varphi'$ where φ' is obtained from φ by replacing all occurrences of atoms $r(u)$ with $\psi(u) \vee \psi'(u)$. ❑

Co/Horn clause syntax admits four ways of axiomatizing invariants resp. congruences and thus restricting resp. factoring initial or final models:

Theorem 4.4. (restrictions) [35] For all $s \in S$, let $r : s \in R$. (1) Suppose that AX includes co-Horn clauses such that for all $A \in Alg_{\Sigma,C}$, r^A is a Σ-invariant. Let C be final and $gfp = gfp(\overline{AX,C})$. AX meets certain syntactical restrictions, then r^{gfp} is the greatest Σ-invariant of gfp satisfying AX and final in the category $Alg^{\bigtriangledown}_{\Sigma,C,AX}$ of all algebras of $Alg_{\Sigma,C}$ that satisfy AX and interpret $r : s$ as C_s.

(2) Suppose that AX includes Horn clauses such that for all $A \in Alg_{\Sigma,C}$, r^A is a Σ-invariant. If C is initial, then Lemma 4.1 implies $r^A = C$, and it is easy to see that algebraic induction is a special case of relation induction. Let C be final and $lfp = lfp(\overline{AX,C})$. If AX meets certain syntactical restrictions, then r^{lfp} is initial in the category of all F-*observable* (see below) algebras of$Alg^{\bigtriangledown}_{\Sigma,C,AX}$. ❑

Theorem 4.5. (abstractions) [35] For all $s \in S$, let $r : s \times s \in R$. (1) Suppose that AX consists of Horn clauses such that for all $A \in Alg_{\Sigma,C}$, r^A is a Σ-congruence. Let C be initial and $lfp = lfp(\overline{AX,C})$. If AX meets certain syntactical restrictions, then r^{lfp} is the least Σ-congruence on lfp satisfying AX and the quotient of lfp by r^{lfp} is initial in the category $Alg^{\bigtriangleup}_{\Sigma,C,AX}$ of all algebras of $Alg_{\Sigma,C}$ that satisfy AX and interpret $r : s \times s$ as $\langle id, id\rangle(C_s)$.

(2) Suppose that AX includes co-Horn clauses such that for all $A \in Alg_{\Sigma,C}$, r^A is a Σ-congruence. If C is final, then Lemma 4.1 implies $r^A = \langle id, id\rangle(C)$, and it is easy to see that algebraic coinduction is a special case of relation coinduction. Let C is initial and $gfp = gfp(\overline{AX,C})$. If AX meets certain syntactical restrictions, then the quotient of gfp by r^{gfp} is final in the category of all F-*reachable* (see below) algebras of $Alg^{\bigtriangleup}_{\Sigma,C,AX}$. r^{gfp} coincides with the final semantics [27,47,48] deal with. ❑

Let $F' \subseteq F$ and $\Sigma' = (S, F', R)$, A be a Σ-algebra and $B = A|_{\Sigma'}$.

Let Σ be constructive, AX be a set of Σ-Horn clauses, $AX' \subseteq AX$ and $\mu\Sigma'$ and $\mu\Sigma$ be initial in $Alg^{\bigtriangleup}_{\Sigma',C,AX'}$ resp. $Alg^{\bigtriangleup}_{\Sigma,C,AX}$. A is F'**-reachable** or -generated if $fold^B : \mu\Sigma' \to B$ is surjective. A is F'**-consistent** if $fold^B$ is injective. (Σ, AX) is a **conservative extension** of (Σ', AX') if $\mu\Sigma$ is F'-reachable and F'-consistent, i.e. if $\mu\Sigma|_{\Sigma'}$ and $\mu\Sigma'$ are isomorphic.

Let Σ be destructive, AX be a set of Σ-co-Horn clauses, $AX' \subseteq AX$ and $\nu\Sigma'$ and $\nu\Sigma$ be initial in $Alg^{\bigtriangledown}_{\Sigma',C,AX'}$ resp. $Alg^{\bigtriangledown}_{\Sigma,C,AX}$. A is F'**-observable** or -cogenerated if $unfold^B : B \to \nu\Sigma'$ is injective. A is F'**-complete** if $unfold^B$ is surjective. (Σ, AX) is a **conservative extension** of (Σ', AX') if $\nu\Sigma$ is F'-observable and F'-complete, i.e. $\nu\Sigma|_{\Sigma'}$ and $\nu\Sigma'$ are isomorphic.

Proposition 4.6. [35] Let Σ be constructive. If A is F-reachable, then A is F'-reachable iff $img(fold^A)$ is compatible with $F \setminus F'$. If $\mu\Sigma'$ can be extended to an algebra of $Alg^{\bigtriangleup}_{\Sigma,C,AX}$, then (Σ, AX) is a conservative extension of (Σ', AX') and, equivalently (!), $\mu\Sigma'$ satisfies the same Σ-formulas as $\mu\Sigma|_{\Sigma'}$ does.

Let Σ be destructive. If A is F-observable, then A is F'-observable iff $ker(unfold^A)$is compatible with $F \setminus F'$. If $\nu\Sigma'$ can be extended to an algebra of

$Alg^{\nabla}_{\Sigma,C,AX}$, then (Σ, AX) is a conservative extension of (Σ', AX') and, equivalently (!), $\nu\Sigma'$ satisfies the same Σ-formulas as $\nu\Sigma|_{\Sigma'}$ does. ❏

Conservative extensions add constructors or destructors to a signature without changing the carrier of the initial resp. final model. Each other functions can be axiomatized in terms of co/recursive equations, which means that there is a Σ-algebra A such that f agrees with $fold^A$ resp. $unfold^A$. By Prop. 2.4 (2), this holds true iff f is simply an S-sorted function whose kernel resp. image is compatible with F. (The use Prop. 2.4 (2) for co/recursive definitions on initial resp. final co/algebras was first suggested by [15], Thm. 4.2 resp. 5.2.) However, as constructors and destructors usually are not (components of) S-sorted functions, the domain or range of f is seldom a single sort $s \in S$, but a composed type $e \in \mathbb{BT}(S)$. Hence we follow [22] and start out from a category $\mathcal{K}$ and an adjunction between $\mathcal{K}$ and Set^S such that f can be described as a $\mathcal{K}$-morphism, while $fold^A$ resp. $unfold^A$ comes up as the unique Set^S-extension of f that the adjunction generates:

Let $\Sigma = (S, F, R)$ be a constructor signature and $(L : Set^S \to \mathcal{K}, G : \mathcal{K} \to Set^S, \eta, \epsilon)$ be an adjunction. A $\mathcal{K}$-morphism $f : L(\mu\Sigma) \to B$ is **Σ-recursive** if the kernel of the Set^S-extension $f^\# : \mu\Sigma \to G(B)$ of f is compatible with F.

Let $\Sigma = (S, F, R)$ be a destructor signature and $(L : \mathcal{K} \to Set^S, G : Set^S \to \mathcal{K}, \eta, \epsilon)$ be an adjunction. A $\mathcal{K}$-morphism $f : A \to G(\nu\Sigma)$ is **Σ-corecursive** if the image of the Set^S-extension $f^* : L(A) \to \nu\Sigma$ of f is compatible with F.

Example 4.7. The factorial function $fact : \mathbb{N} \to \mathbb{N}$ is usually axiomatized by the following equations involving the constructors $0 : 1 \to nat$ and $succ : nat \to nat$ of Nat (see Ex. 2.1):

$$fact(0) = 1, \qquad fact(n+1) = fact(n) * (n+1).$$

Since by Ex. 3.8, $\mathbb{N}$ is an initial Nat-algebra, we may show that $fact$ is Nat-recursive. This cannot be concluded from the above equations because the variable n occurs at a non-argument position. Hence we add the identity on $\mathbb{N}$ and show that the desired property for $fact$ and id simultaneously. The corresponding equations read as follows:

$$\langle fact, id\rangle(0) = (1, 0), \qquad \langle fact, id\rangle(n+1) = (fact(n) * (id(n)+1), id(n)+1).$$

We choose the product adjunction

$$((\_,\_) : Set \to Set^2, \times : Set^2 \to Set, \lambda A.\langle id_A, id_A\rangle, (\pi_1, \pi_2)).$$

The latter equations imply that the kernel of the Set-extension $(fact, id)^\# = \langle fact, id\rangle : \mathbb{N} \to \mathbb{N}^2$ of $(fact, id) : (\mathbb{N}, \mathbb{N}) \to (\mathbb{N}, \mathbb{N})$ is compatible with 0 and $succ$. Hence $(fact, id)$ is Nat-recursive and by Prop. 2.4 (2), $\langle fact, id\rangle$ is a Nat-homomorphism, in particular, $\mathbb{N}^2$ is a Nat-algebra: $0^{\mathbb{N}^2} = (1, 0)$ and $succ^{\mathbb{N}^2} = \lambda(m, n).(m * (n+1), n+1)$. ❏

Example 4.8. The streams $\overline{01} = [0, 1, 0, 1, \ldots]$ and $\overline{10} = [1, 0, 1, 0, \ldots]$ can be axiomatized by the following equations involving the destructors $\delta : state \to state$ and $\beta : state \to 2$ of $DetAut(1, 2)$ (see Ex. 2.2):

$$\langle\delta,\beta\rangle(\overline{01}) = (\overline{10},0), \qquad \langle\delta,\beta\rangle(\overline{10}) = (\overline{01},1) \tag{1}$$

Since by Ex. 3.10, $2^{\mathbb{N}}$ is a final $DetAut(1,2)$-algebra, we may show that $(\overline{01},\overline{10})$ is $DetAut(1,2)$-corecursive. We choose the coproduct adjunction

$$(+ : Set^2 \to Set, (\_,\_) : Set \to Set^2, (\iota_1,\iota_2), \lambda A.[id_A, id_A]).$$

The above equations imply that the image of the Set-extension $(\overline{01},\overline{10})^* = [\overline{01},\overline{10}] : 2 \to 2^{\mathbb{N}}$ of $(\overline{01},\overline{10}) : (1,1) \to (2^{\mathbb{N}},2^{\mathbb{N}})$ is compatible with δ and β. Hence $(\overline{01},\overline{10})$ is $DetAut(1,2)$-corecursive and by Prop. 2.4 (2), $[\overline{01},\overline{10}]$ is a $DetAut(1,2)$-homomorphism, in particular, 2 is a $DetAut(1,2)$-algebra: $\delta^2(0) = 1$, $\delta^2(1) = 0$ and $\beta^2 = id_2$.

Since for all sets 2, $DetAut(1,2)$ admits coterms, $DetAut(1,2)$ induces the constructive signature $CoDetAut(1,2) = (\{state\}, \{cons : 2 \times state \to state\}, \emptyset)$ that admits terms (see Section 3). It does not matter that *initial* $CoDetAut(1,2)$-algebras are empty. Here we only use the syntax of $CoDetAut(1,2)$: The streams $\overline{01}$ and $\overline{10}$ can be axiomatized by equations involving *cons*:

$$\overline{01} = cons(0,\overline{10}), \qquad \overline{10} = cons(1,\overline{01}). \tag{2}$$

The definition of $\overline{01}$ and $\overline{10}$ derived from (1) provides a *solution* of (2) where $\overline{01}$ and $\overline{10}$ are regarded as variables. Conversely, each solution (a,b) of (2) has the unique Set-extension $[a,b]$, which is a $DetAut(1,2)$-homomorphism into a final $DetAut(1,2)$-algebra and thus unique. Hence (2) has a unique solution! ❏

The last observation can be generalized to the following result obtained in several ways and on many levels of abstraction (see, e.g., [18], Thm. 5.2; [3], Thm. 3.3): Given a constructive signature Σ that admits terms, *ideal* or *guarded* Σ-equations like (2) have unique solutions in CT_Σ (see Section 3). Via this result, coalgebra has even found its way into functional programming (see, e.g. [42,23]).

References

1. Adámek, J.: Introduction to Coalgebra. Theory and Applications of Categories 14, 157–199 (2005)
2. Adámek, J.: Final coalgebras are ideal completions of initial algebras. Journal of Logic and Computation 12, 217–242 (2002)
3. Aczel, P., Adámek, J., Velebil, J.: A Coalgebraic View of Infinite Trees and Iteration. In: Proc. Coalgebraic Methods in Computer Science. ENTCS, vol. 44, pp. 1–26. Elsevier, Amsterdam (2001)
4. Adámek, J., Milius, S., Moss, L.S.: Initial algebras and terminal coalgebras: a survey, draft of February 7, TU Braunschweig (2011)
5. Adámek, J., Porst, H.-E.: From varieties of algebras to covarieties of coalgebras. In: Proc. Coalgebraic Methods in Computer Science. ENTCS, vol. 44, pp. 27–46. Elsevier, Amsterdam (2001)
6. Adámek, J., Porst, H.-E.: On Tree Coalgebras and Coalgebra Presentations. Theoretical Computer Science 311, 257–283 (2004)

7. Arbib, M.A.: Free dynamics and algebraic semantics. In: Karpinski, M. (ed.) FCT 1977. LNCS, vol. 56, pp. 212–227. Springer, Heidelberg (1977)
8. Arbib, M.A., Manes, E.G.: Parametrized Data Types Do Not Need Highly Constrained Parameters. Information and Control 52, 139–158 (1982)
9. Astesiano, E., Kreowski, H.-J., Krieg-Brückner, B. (eds.): Algebraic Foundations of Systems Specification. IFIP State-of-the-Art Report. Springer, Heidelberg (1999)
10. Barr, M.: Coequalizers and Free Triples. Math. Zeitschrift 116, 307–322 (1970)
11. Barr, M.: Terminal coalgebras in well-founded set theory. Theoretical Computer Science 114, 299–315 (1993)
12. van den Brand, M.G.J., Heering, J., Klint, P., Olivier, P.A.: Compiling Rewrite Systems: The ASF+SDF Compiler. ACM TOPLAS 24 (2002)
13. Brzozowski, J.A.: Derivatives of regular expressions. Journal ACM 11, 481–494 (1964)
14. Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specification, vol. 1. Springer, Heidelberg (1985)
15. Gibbons, J., Hutton, G., Altenkirch, T.: When is a function a fold or an unfold? In: Proc. Coalgebraic Methods in Computer Science. ENTCS, vol. 44, pp. 146–159. Elsevier, Amsterdam (2001)
16. Goguen, J., Malcolm, G.: A Hidden Agenda. Theoretical Computer Science 245, 55–101 (2000)
17. Goguen, J.A., Thatcher, J.W., Wagner, E.G.: An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types. In: Yeh, R. (ed.) Current Trends in Programming Methodology, vol. 4, pp. 80–149. Prentice-Hall, Englewood Cliffs (1978)
18. Goguen, J.A., Thatcher, J.W., Wagner, E.G., Wright, J.B.: Initial Algebra Semantics and Continuous Algebras. J. ACM 24, 68–95 (1977)
19. Gumm, H.P., Schröder, T.: Coalgebras of bounded type. Math. Structures in Computer Science 12, 565–578 (2002)
20. Gumm, H.P.: Universelle Coalgebra. In: Ihringer, T. (ed.) Allgemeine Algebra. Heldermann Verlag (2003)
21. Guttag, J., Horowitz, E., Musser, D.R.: Abstract Data Types and Software Validation. Communications of the ACM 21, 1048–1064 (1978)
22. Hinze, R.: Adjoint Folds and Unfolds. In: Bolduc, C., Desharnais, J., Ktari, B. (eds.) MPC 2010. LNCS, vol. 6120, pp. 195–228. Springer, Heidelberg (2010)
23. Hinze, R.: Reasoning about Codata. In: Horváth, Z., Plasmeijer, R., Zsók, V. (eds.) CEFP 2009. LNCS, vol. 6299, pp. 42–93. Springer, Heidelberg (2010)
24. Jacobs, B.: Invariants, Bisimulations and the Correctness of Coalgebraic Refinements. In: Johnson, M. (ed.) AMAST 1997. LNCS, vol. 1349, pp. 276–291. Springer, Heidelberg (1997)
25. Jacobs, B.: Introduction to Coalgebra. Radboud University, Nijmegen (2005)
26. Jacobs, B.: A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages. In: Futatsugi, K., Jouannaud, J.-P., Bevilacqua, V. (eds.) Algebra, Meaning, and Computation. LNCS, vol. 4060, pp. 375–404. Springer, Heidelberg (2006)
27. Kamin, S.: Final Data Type Specifications: A New Data Type Specification Method. ACM TOPLAS 5, 97–123 (1983)
28. Lambek, J.: A fixpoint theorem for complete categories. Math. Zeitschrift 103, 151–161 (1968)
29. Lehmann, D.J., Smyth, M.B.: Algebraic Specification of Data Types: A Synthetic Approach. Math. Systems Theory 14, 97–139 (1981)

30. Meseguer, J., Goguen, J.A.: Initiality, Induction and Computability. In: Nivat, M., Reynolds, J. (eds.) Algebraic Methods in Semantics, pp. 459–541. Cambridge University Press, Cambridge (1985)
31. Meseguer, J., Rosu, G.: The Rewriting Logic Semantics Project. Theoretical Computer Science 373 (2007)
32. van der Meulen, E.A.: Deriving incremental implementations from algebraic specifications. In: Proc. 2nd AMAST, pp. 277–286. Springer, Heidelberg (1992)
33. Padawitz, P.: Proof in Flat Specifications. In: Astesiano, E., Kreowski, H.-J., Krieg-Brückner, B. (eds.) Algebraic Foundations of Systems Specification. IFIP State-of-the-Art Report, pp. 321–384. Springer, Heidelberg (1999)
34. Padawitz, P.: Swinging Types = Functions + Relations + Transition Systems. Theoretical Computer Science 243, 93–165 (2000)
35. Padawitz, P.: Dialgebraic Specification and Modeling, slides, TU Dortmund (2011), `http://fldit-www.cs.tu-dortmund.de/~peter/DialgSlides.pdf`
36. Padawitz, P.: Algebraic compilers and their implementation in Haskell, Sierra Nevada IFIP WG 1.3 Meeting (January 14-18, 2008)
37. Padawitz, P.: Algebraic Model Checking. In: Drewes, F., Habel, A., Hoffmann, B., Plump, D. (eds.) Manipulation of Graphs, Algebras and Pictures. Electronic Communications of the EASST, vol. 26 (2010), extended slides `http://fldit-www.cs.tu-dortmund.de/~peter/CTL.pdf`
38. Padawitz, P.: Expander2 as a Prover and Rewriter, `http://fldit-www.cs.tu-dortmund.de/~peter/Prover.pdf`
39. Padawitz, P.: Übersetzerbau, course notes, TU Dortmund (2010), `http://fldit-www.cs.tu-dortmund.de/~peter/CbauFolien.pdf`
40. Rutten, J.J.M.M.: Universal Coalgebra: A Theory of Systems. Theoretical Computer Science 249, 3–80 (2000)
41. Rutten, J.J.M.M.: Automata and coinduction (an exercise in coalgebra). In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 194–218. Springer, Heidelberg (1998)
42. Rutten, J.J.M.M.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theoretical Computer Science 308, 1–53 (2003)
43. Rutten, J.J.M.M., Turi, D.: Initial Algebra and Final Coalgebra Semantics for Concurrency, Report CS-R9409, CWI, Amsterdam (1994)
44. Sen, K., Rosu, G.: Generating Optimal Monitors for Extended Regular Expressions. In: Proc. Runtime Verification 2003. ENTCS, vol. 89, pp. 226–245. Elsevier, Amsterdam (2003)
45. Thatcher, J.W., Wagner, E.G., Wright, J.B.: More on Advice on Structuring Compilers and Proving Them Correct. Theoretical Computer Science 15, 223–249 (1981)
46. Visser, E.: Program Transformation with Stratego/XT: Rules, Strategies, Tools, and Systems. In: Lengauer, C., Batory, D., Blum, A., Vetta, A. (eds.) Domain-Specific Program Generation. LNCS, vol. 3016, pp. 216–238. Springer, Heidelberg (2004)
47. Wand, M.: Final algebra semantics and data type extension. J. Comp. Syst. Sci. 19, 27–44 (1979)
48. Wirsing, M.: Algebraic Specification. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, pp. 675–788. Elsevier, Amsterdam (1990)

Theme and Variations on the Concatenation Product

Jean-Éric Pin[⋆]

LIAFA, University Paris-Diderot and CNRS, France

Abstract. The concatenation product is one of the most important operations on regular languages. Its study requires sophisticated tools from algebra, finite model theory and profinite topology. This paper surveys research advances on this topic over the last fifty years.

The concatenation product plays a key role in two of the most important results of automata theory: Kleene's theorem on regular languages [23] and Schützenberger's theorem on star-free languages [60].

This article surveys the most important results and tools related to the concatenation product, including connections with algebra, profinite topology and finite model theory. The paper is organised as follows: Section 1 presents some useful algebraic tools for the study of the concatenation product. Section 2 introduces the main definitions on the product and its variants. The classical results are summarized in Section 3. Sections 4 and 5 are devoted to the study of two algebraic tools: Schützenberger products and relational morphisms. Closure properties form the topic of Section 6. Hierarchies and their connection with finite model theory are presented in Sections 7 and 8. Finally, new directions are suggested in Section 9.

1 The Instruments

This section is a brief reminder on the algebraic notions needed to study the concatenation product: semigroups and semirings, syntactic ordered monoids, free profinite monoids, equations and identities, varieties and relational morphisms. More information can be found in [1,2,3,18,35,42,45].

1.1 Semigroups and Semirings

If S is a semigroup, the set $\mathcal{P}(S)$ of subsets of S is also a semiring, with union as addition and multiplication defined, for every $X, Y \in \mathcal{P}(S)$, by

$$XY = \{xy \mid x \in X, y \in Y\}$$

In this semiring, the empty set is the zero and for this reason, is denoted by 0. It is also convenient to denote simply by x a singleton $\{x\}$.

If k is a semiring, we denote by $M_n(k)$ be the semiring of square matrices of size n with entries in k.

⋆ The author acknowledges support from the project ANR 2010 BLAN 0202 02 FREC.

1.2 Syntactic Ordered Monoid

Let L be a language of A^*. The *syntactic preorder* of L is the relation $\leqslant_L$ defined on A^* by $u \leqslant_L v$ if and only if, for every $x, y \in A^*$,

$$xvy \in L \Rightarrow xuy \in L$$

The *syntactic congruence* of L is the relation $\sim_L$ defined by $u \sim_L v$ if and only if $u \leqslant_L v$ and $v \leqslant_L u$.

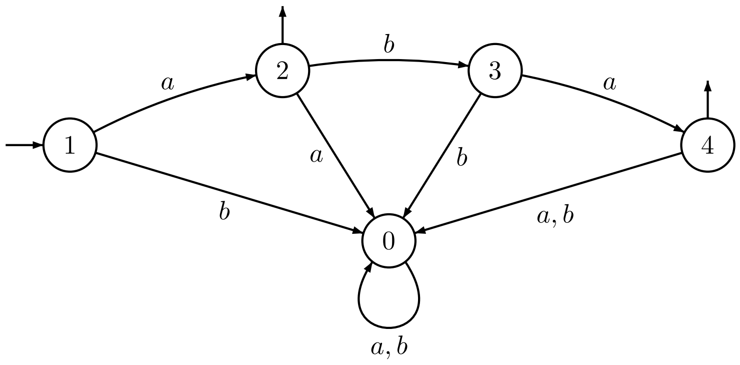
The *syntactic monoid* of L is the quotient $M(L)$ of A^* by $\sim_L$ and the natural morphism $\eta : A^* \to A^*/{\sim_L}$ is called the *syntactic morphism* of L. The syntactic preorder $\leqslant_L$ induces an order on the quotient monoid $M(L)$. The resulting ordered monoid is called the *syntactic ordered monoid* of L.

The syntactic ordered monoid can be computed from the minimal automaton as follows. First observe that if $\mathcal{A} = (Q, A, \cdot, q_-, F)$ is a minimal deterministic automaton, the relation $\leqslant$ defined on Q by $p \leqslant q$ if for all $u \in A^*$,

$$q\cdot u \in F \Rightarrow p\cdot u \in F$$

is an order relation, called the *syntactic order* of the automaton. Then the syntactic ordered monoid of a language is the transition monoid of its ordered minimal automaton. The order is defined by $u \leqslant v$ if and only if, for all $q \in Q$, $q\cdot u \leqslant q\cdot v$.

For instance, let L be the language $\{a, aba\}$. Its minimal deterministic automaton is represented below:



The order on the set of states is $2 \leqslant 4$, $1 \leqslant 3$ and $1, 2, 3, 4 \leqslant 0$. Indeed, one has $0\cdot u = 0$ for all $u \in A^*$ and thus, the formal implication

$$0\cdot u \in F \Rightarrow q\cdot u \in F$$

holds for any state q. Similarly, $1 \leqslant 3$ since a is the only word such that $3\cdot a \in F$ and one also has $1\cdot a \in F$.

The syntactic monoid of L is the monoid $M = \{1, a, b, ab, ba, aba, 0\}$ presented by the relations $a^2 = b^2 = bab = 0$. Its syntactic order is $1 < ab < 0$, $1 < ba < 0$, $a < aba < 0$, $b < 0$.

1.3 Free Profinite Monoids

We briefly recall the definition of a free profinite monoid. More details can be found in [1,45]. A finite monoid M *separates* two words u and v of A^* if there is a morphism $\varphi : A^* \to M$ such that $\varphi(u) \neq \varphi(v)$. We set

$$r(u,v) = \min\big\{|M| \mid M \text{ is a finite monoid that separates } u \text{ and } v\ \big\}$$

and $d(u,v) = 2^{-r(u,v)}$, with the usual conventions $\min \emptyset = +\infty$ and $2^{-\infty} = 0$. Then d is a *metric* on A^* and the completion of A^* for this metric is denoted by $\widehat{A^*}$. The product on A^* can be extended by continuity to $\widehat{A^*}$. This extended product makes $\widehat{A^*}$ a compact topological monoid, called the *free profinite monoid.* Its elements are called *profinite words.*

In a compact monoid, the smallest closed subsemigroup containing a given element s has a unique idempotent, denoted s^ω. This is true in particular in a finite monoid and in the free profinite monoid.

One can show that every morphism φ from A^* into a (discrete) finite monoid M extends uniquely to a a uniformly continuous morphism $\widehat{\varphi}$ from $\widehat{A^*}$ to M. It follows that if x is a profinite word, then $\widehat{\varphi}(x^\omega) = \widehat{\varphi}(x)^\omega$.

1.4 Equations and Identities

Let φ be a morphism from A^* into a finite [ordered] monoid M and let x, y be two profinite words of $\widehat{A^*}$. We say that φ *satisfies the profinite equation* $x = y$ $[x \leqslant y]$ if $\widehat{\varphi}(x) = \widehat{\varphi}(y)$ $[\widehat{\varphi}(x) \leqslant \widehat{\varphi}(y)]$.

A regular language of A^* *satisfies a profinite equation* if its syntactic morphism satisfies this equation. More generally, we say that a set of regular languages $\mathcal{L}$ *is defined a set of profinite equations* E if $\mathcal{L}$ is the set of all regular languages satisfying every equation of E.

A *lattice of languages* is a set $\mathcal{L}$ of languages of A^* containing $\emptyset$ and A^* and closed under finite union and finite intersection. It is closed under quotients if, for each $L \in \mathcal{L}$ and $u \in A^*$, the languages $u^{-1}L$ and Lu^{-1} are also in $\mathcal{L}$. It is proved in [19] that a set of regular languages is a lattice [Boolean algebra] closed under quotient if and only if it can be defined by a set of profinite equations of the form $u \leqslant v$ $[u = v]$.

A finite [ordered] monoid M *satisfies the identity* $x = y$ $[x \leqslant y]$ if every morphism from A^* into M satisfies this equation. These notions can be extended to semigroups by considering morphisms from the free semigroup A^+ to a finite semigroup.

1.5 Varieties of Monoids

In this paper, we will only consider varieties in Eilenberg's sense. Thus, for us, a *variety of semigroups* is a class of finite semigroups closed under taking subsemigroups, quotients and finite direct products [18]. Varieties of *ordered semigroups*, *monoids* and *ordered monoids* are defined analogously [39].

Given a set E of identities, we denote by $[\![E]\!]$ the class of all finite [ordered] monoids which satisfy all the identities of E. Reiterman's theorem [57] and its extension to ordered structures [53] states that every variety of [ordered] monoids [semigroups] can be defined by a set of identities. For instance, the variety of ordered semigroups $[\![x^\omega y x^\omega \leqslant x^\omega]\!]$ is the variety of ordered semigroups S such that, for each idempotent $e \in S$ and for each $s \in S$, $ese \leqslant e$.

The following varieties will be used in this paper: the variety $\mathbf{A}$ of *aperiodic* monoids, defined by the identity $x^{\omega+1} = x^\omega$, the variety $\mathbf{R}$ [$\mathbf{L}$] of $\mathcal{R}$-*trivial* [$\mathcal{L}$-*trivial*] monoids, defined by the identity $(xy)^\omega x = x^\omega$ [$y(xy)^\omega = x^\omega$] and the variety $\mathbf{DA}$, which consists of the aperiodic monoids whose regular $\mathcal{J}$-classes are idempotent semigroups. This variety is defined by the identities $x^\omega = x^{\omega+1}$ and $(xy)^\omega(yx)^\omega(xy)^\omega = (xy)^\omega$.

We will also consider two group varieties: the variety $\mathbf{G}_p$ of p-groups (for a prime p) and the variety $\mathbf{Gsol}$ of soluble groups.

Finally, if $\mathbf{V}$ is a variety of monoids, the class of all semigroups S such that, for each idempotent $e \in S$, the "local" monoid eSe belongs to $\mathbf{V}$, form a variety of semigroups, denoted $\mathbf{LV}$. In particular, the variety $\mathbf{LI}$ is the variety of locally trivial semigroups, defined by the identity $x^\omega y x^\omega = x^\omega$.

1.6 Varieties of Languages

A *class of languages* $\mathcal{C}$ associates with each alphabet A a set $\mathcal{C}(A^*)$ of regular languages of A^*. A *positive variety of languages* is a class of languages $\mathcal{V}$ such that, for all alphabets A and B,

(1) $\mathcal{V}(A^*)$ is a lattice of languages closed under quotients,

(2) if $\varphi\colon A^* \to B^*$ is a morphism, then $L \in \mathcal{V}(B^*)$ implies $\varphi^{-1}(L) \in \mathcal{V}(A^*)$.

A *variety of languages* is a positive variety $\mathcal{V}$ such that, for each alphabet A, $\mathcal{V}(A^*)$ is closed under complement. We can now state Eilenberg's variety theorem [18] and its counterpart for ordered monoids [39].

Theorem 1.1. *Let $\mathbf{V}$ be a variety of monoids. For each alphabet A, let $\mathcal{V}(A^*)$ be the set of all languages of A^* whose syntactic monoid is in $\mathbf{V}$. Then $\mathcal{V}$ is a variety of languages. Further, the correspondence $\mathbf{V} \to \mathcal{V}$ is a bijection between varieties of monoids and varieties of languages.*

Theorem 1.2. *Let $\mathbf{V}$ be a variety of ordered monoids. For each alphabet A, let $\mathcal{V}(A^*)$ be the set of all languages of A^* whose syntactic ordered monoid is in $\mathbf{V}$. Then $\mathcal{V}$ is a positive variety of languages. Further, the correspondence $\mathbf{V} \to \mathcal{V}$ is a bijection between varieties of ordered monoids and positive varieties of languages.*

A slightly more general definition was introduced by Straubing [71]. Let $\mathcal{C}$ be a class of morphisms between free monoids, closed under composition and containing all length-preserving morphisms. Examples include the classes of all *length-preserving* morphisms, of all *length-multiplying* morphisms (morphisms such that, for some integer k, the image of any letter is a word of length k), all *non-erasing* morphisms (morphisms for which the image of each letter is a nonempty word), all *length-decreasing* morphisms (morphisms for which the image of each letter is either a letter or the empty word) and all morphisms.

A *positive $\mathcal{C}$-variety of languages* is a class $\mathcal{V}$ of recognisable languages satisfying the first condition defining a positive variety of languages and a second condition

(2′) if $\varphi\colon A^* \to B^*$ is a morphism in $\mathcal{C}$, $L \in \mathcal{V}(B^*)$ implies $\varphi^{-1}(L) \in \mathcal{V}(A^*)$.

A *$\mathcal{C}$-variety of languages* is a positive $\mathcal{C}$-variety of languages closed under complement. When $\mathcal{C}$ is the class of non-erasing morphisms (for which the image of a letter is a nonempty word), we use the term *ne*-variety. These *ne*-varieties are essentially the same thing as Eilenberg's +-varieties (see [49, p. 260–261] for a detailed discussion) and they correspond to varieties of semigroups.

1.7 Relational Morphisms

A *relational morphism* between two monoids M and N is a function τ from M into $\mathcal{P}(N)$ such that:

(1) for all $M \in M$, $\tau(m) \neq \emptyset$,

(2) $1 \in \tau(1)$,

(3) for all $m, n \in M$, $\tau(m)\tau(n) \subseteq \tau(mn)$

Let $\mathbf{V}$ be a variety of [ordered] semigroups. A [relational] morphism $\tau : M \to N$ is said to be a [relational] $\mathbf{V}$*-morphism* if for every [ordered] semigroup R of N belonging to $\mathbf{V}$, the [ordered] semigroup $\tau^{-1}(R)$ also belongs to $\mathbf{V}$.

Let me point out an important subtlety. The definition of a [relational] $\mathbf{V}$-morphism adopted in this paper is taken from [44] and differs from the original definition given for instance in [68,42]. The original definition only requires that, for each idempotent e, the [ordered] semigroup $\tau^{-1}(e)$ also belongs to $\mathbf{V}$. In many cases the two definitions are equivalent: for instance, when $\mathbf{V}$ is one of the varieties $\mathbf{A}$, $[\![x^\omega y x^\omega = x^\omega]\!]$, $[\![x^\omega y = x^\omega]\!]$, $[\![y x^\omega = x^\omega]\!]$ or $\mathbf{LH}$ where $\mathbf{H}$ is a variety of groups. However, the two definitions are *not equivalent* for the variety $[\![x^\omega y x^\omega \leqslant x^\omega]\!]$.

2 Theme and Variations: The Concatenation Product

We now come to the main topic of this article. Just like a piece of classical music, the concatenation product includes theme and variations.

2.1 Main Theme

The *product* (or *concatenation product*) of the languages $L_0, L_1, \ldots, L_n$ of A^* is the language

$$L_0 L_1 \cdots L_n = \{u_0 u_1 \cdots u_n \mid u_0 \in L_0, u_1 \in L_1, \cdots, u_n \in L_n\}$$

A language L of A^* is a *marked product* of the languages $L_0, L_1, \ldots, L_n$ if

$$L = L_0 a_1 L_1 \cdots a_n L_n$$

for some letters $a_1, \ldots, a_n$ of A.

2.2 Three Variations

Variations include the unambiguous, deterministic, bideterministic and modular products, that are defined below.

Unambiguous Product

A marked product $L = L_0a_1L_1 \cdots a_nL_n$ is said to be *unambiguous* if every word of L admits a unique decomposition of the form $u = u_0a_1u_1 \cdots a_nu_n$ with $u_0 \in L_0$, …, $u_n \in L_n$. For instance, the marked product $\{a, c\}^*a\{1\}b\{b, c\}^*$ is unambiguous.

Deterministic Product

A word x is a *prefix* [*suffix*] of a word u if there is a word v such that $u = xv$ [$u = vx$]. It is a *proper* prefix [suffix] if $x \neq u$. A subset C of A^+ is a *prefix* [*suffix*] *code* if if no element of C is a proper prefix [suffix] of another element of C.

A marked product $L = L_0a_1L_1 \cdots a_nL_n$ of n nonempty languages $L_0, L_1, \ldots, L_n$ of A^* is *left* [*right*] *deterministic* if, for $1 \leqslant i \leqslant n$, the set $L_0a_1L_1 \cdots L_{i-1}a_i$ [$a_iL_i \cdots a_nL_n$] is a prefix [suffix] code. This means that every word of L has a unique prefix [suffix] in $L_0a_1L_1 \cdots L_{i-1}a_i$ [$a_iL_i \cdots a_nL_n$]. It is observed in [9, p. 495] that the marked product $L_0a_1L_1 \cdots a_nL_n$ is deterministic if and only if, for $1 \leqslant i \leqslant n$, the language $L_{i-1}a_i$ is a prefix code. Since the product of two prefix codes is a prefix code, any left [right] deterministic product of left [right] deterministic products is left [right] deterministic.

A marked product is said to be *bideterministic* if it is both left and right deterministic.

Modular Product of Languages

Let $L_0, \ldots, L_n$ be languages of A^*, let $a_1, \ldots, a_n$ be letters of A and let r and p be integers such that $0 \leqslant r < p$. We define the *modular product* of the languages $L_0, \ldots, L_n$ with respect to r and p, denoted $(L_0a_1L_1 \cdots a_nL_n)_{r,p}$, as the set of all words u in A^* such that the number of factorizations of u in the form $u = u_0a_1u_1 \cdots a_nu_n$, with $u_i \in L_i$ for $0 \leqslant i \leqslant n$, is congruent to r modulo p.

A language is a *p-modular product* of the languages $L_0, \ldots, L_n$ if it is of the form $(L_0a_1L_1 \cdots a_nL_n)_{r,p}$ for some r.

3 Classical Area

The most important results on the concatenation product are due to Schützenberger. They concern the smallest Boolean algebra of languages closed under marked product or one of its variants.

Recall that the set of *star-free languages* is the smallest Boolean algebra of languages of A^* which is closed under marked product.

Theorem 3.1 (Schützenberger [60]). *A regular language is star-free if and only if its syntactic monoid is aperiodic.*

There are essentially two proofs of this result. Schützenberger's original proof [60,35], slightly simplified in [30], works by induction on the $\mathcal{J}$-depth of the syntactic semigroup. Schützenberger's proof actually gives a stronger result since it shows that the star-free languages form the smallest Boolean algebra of languages of A^* which is closed under marked products of the form $L \to LaA^*$ and A^*aL. In other words, marked products with A^* suffice to generate all star-free languages.

The other proof [17,28] makes use of a weak form of the Krohn-Rhodes theorem: every aperiodic semigroup divides a wreath product of copies of the monoid $U_2 = \{1, a, b\}$, given by the multiplication table $aa = a$, $ab = b$, $ba = b$ and $bb = b$.

Theorem 3.1 provides an algorithm to decide whether a given regular language is star-free. The complexity of this algorithm is analysed in [16,65].

Let us define in the same way the set of *unambiguous* [*right deterministic, left deterministic*] *star-free languages* as the smallest Boolean algebra of languages of A^* containing the languages of the form B^*, for $B \subseteq A$, which is closed under unambiguous [left deterministic, right deterministic] marked product. The algebraic characterizations of these classes are also known.

Theorem 3.2 (Schützenberger [61]). *A regular language is unambiguous star-free if and only if its syntactic monoid belongs to* $\mathbf{DA}$.

One can show that the set of unambiguous star-free languages of A^* is the smallest set of languages of A^* containing the languages of the form B^*, for $B \subseteq A$, which is closed under finite disjoint union and unambiguous marked product. The languages corresponding to $\mathbf{DA}$ admit several other nice characterizations: see [72] for a survey.

Deterministic products were also studied in [61]. Alternative descriptions of these languages can be found in [18,13].

Theorem 3.3 ([18]). *A regular language is left* [*right*] *deterministic star-free if and only if its syntactic monoid is $\mathcal{R}$-trivial* [*$\mathcal{L}$-trivial*].

Similar results are known for the p-modular product [18,66,73,76,29,78,79,80].

Theorem 3.4. *Let p be a prime. A language of A^* belongs to the smallest Boolean closed under p-modular product if and only if its syntactic monoid is a p-group.*

Theorem 3.5. *A language of A^* belongs to the smallest Boolean closed under p-modular product for all prime p if and only if its syntactic monoid is a soluble group.*

Finally, one may consider the product and the p-modular products simultaneously.

Theorem 3.6. *A language of A^* belongs to the smallest Boolean closed under product and under p-modular product for all prime p if and only if all the groups in its syntactic monoid are soluble.*

See also [75] for another description of this variety of languages.

4 The Ground Bass: Schützenberger Products

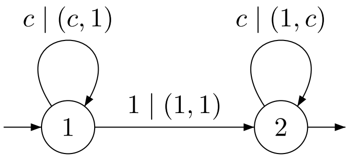
The Schützenberger product is the first algebraic tool used to study the concatenation product. It was first defined by Schützenberger [60] and later generalized by Straubing [67]. An intuitive construction, related to the linear representation of a suitable transducer, was given in [46,47] and is briefly sketched below. More information on transducers and their linear representations can be found in Sakarovitch's book [59].

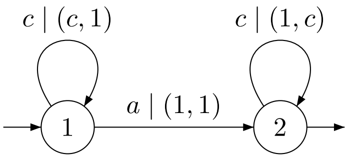
4.1 Transducers for the Product

The construction given in [46,47] relies on the following observation. Let τ and τ_a be the transductions from A^* to $A^* \times A^*$ defined by

$$\tau(u) = \{(u_1, u_2) \mid u_1u_2 = u\}$$
$$\tau_a(u) = \{(u_1, u_2) \mid u_1au_2 = u\}$$

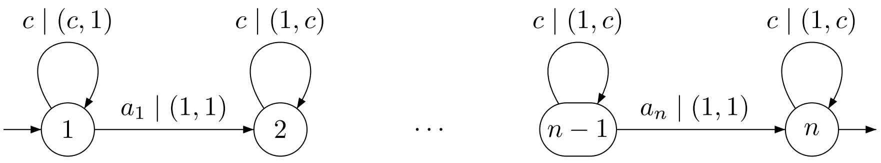
It is easy to see that the two transducers pictured below realise these transductions. In these figures, c is a generic letter and the symbol $\mid$ is a separator between the input letter and the output.





The transducer on the left [right] realizes τ [τ_a]. Now $L_0L_1 = \tau^{-1}(L_0 \times L_1)$ and $L_0aL_1 = \tau_a^{-1}(L_0 \times L_1)$ and this equality allows one to compute a monoid recognising L_0L_1 and L_0aL_1, given monoids recognising L_0 and L_1.

This construction can be readily extended to (marked) products of several languages. For instance, given $a_1, \ldots, a_n \in A$, the transduction σ defined by $\sigma(u) = \{(u_0, \cdots, u_n) \in (A^*)^{n+1} \mid u_0a_1u_1 \cdots a_nu_n = u\}$ is realised by the transducer



and the marked product $L_0a_1L_1 \cdots a_nL_n$ is equal to $\sigma^{-1}(L_0 \times L_1 \times \cdots \times L_n)$. A bit of algebra is now required to make full use of this transduction.

4.2 Linear Representations

The R be the semiring $\mathcal{P}(A^* \times A^*)$. Then for each word u in A^*, $\tau_a(u) = \mu(u)_{1,2}$, where $\mu : A^* \to M_2(R)$ is defined by

$$\mu(a) = \begin{pmatrix} (c,1) & (1,1) \\ 0 & (1,c) \end{pmatrix} \quad \text{and} \quad \mu(c) = \begin{pmatrix} (c,1) & 0 \\ 0 & (1,c) \end{pmatrix} \text{ if } c \neq a$$

Indeed, for each $u \in A^*$, one gets

$$\mu(u) = \begin{pmatrix} (u,1) & \{(u_0,u_1) \mid u_0au_1 = u\} \\ 0 & (1,u) \end{pmatrix}$$

which gives the result. Let now $\pi_0 : A^* \to M_0$ $[\pi_1 : A^* \to M_1]$ be a monoid morphism recognising the language L_0 $[L_1]$ and let $M = M_0 \times M_1$. Let $\pi = \pi_0 \times \pi_1$. Then π is a monoid morphism from $A^* \times A^*$ into M, which can be first extended to a semiring morphism from $A^* \times A^*$ to $\mathcal{P}(M)$ and then to a semiring morphism from $M_2(A^* \times A^*)$ to $M_2(\mathcal{P}(M))$, also denoted by π. It follows that $\pi \circ \mu$ is a morphism from A^* into $M_2(\mathcal{P}(M))$ and it is not difficult to see that this morphism recognises the language $\tau_a^{-1}(L_0 \times L_1)$, that is, L_0aL_1. Further, if u is a word of A^*, the matrix $\pi \circ \mu(u)$ has the form

$$\begin{pmatrix} (m_0,1) & P \\ 0 & (1,m_1) \end{pmatrix}$$

for some $m_0 \in M_0$, $m_1 \in M_1$ and $P \subseteq M_0 \times M_1$. In particular, L_0aL_1 is recognised by the monoid of matrices of this form. This monoid is the *Schützenberger product* of the monoids M_0 and M_1.

A similar representation can be given for the transducer σ and this leads to the definition of the Schützenberger product of $n+1$ monoids $M_0, \ldots, M_n$. In fact, one can give a slightly more general definition. Let $M = M_0 \times \cdots \times M_n$, let k be a semiring and let $k[M]$ be the monoid algebra of M over k. The *Schützenberger product over k* of the monoids $M_0, \ldots, M_n$, is the submonoid of $M_{n+1}(k[M])$ made up of matrices $m = (m_{i,j})$ such that

(1) $m_{i,j} = 0$, for $i > j$,

(2) $m_{i,i} = (1, \ldots, 1, m_i, 1, \ldots, 1)$ for some $m_i \in M_i$,

(3) $m_{i,j} \in k[1 \times \cdots \times 1 \times M_i \times \cdots \times M_j \times 1 \times \cdots \times 1]$, for $i < j$.

This monoid is denoted $k\Diamond(M_0, \ldots, M_n)$. The first condition means that the matrices are uppertriangular, the second one that the entry $m_{i,i}$ can be identified with an element of M_i.

When k is the Boolean semiring, then $k[M]$ is isomorphic to $\mathcal{P}(M)$ and the Schützenberger product is simply denoted $\Diamond(M_0, \ldots, M_n)$. For instance, a matrix of $\Diamond_3(M_1, M_2, M_3)$ will have the form

$$\begin{pmatrix} s_1 & P_{1,2} & P_{1,3} \\ 0 & s_2 & P_{2,3} \\ 0 & 0 & s_3 \end{pmatrix}$$

with $s_i \in M_i$, $P_{1,2} \subseteq M_1 \times M_2$, $P_{1,3} \subseteq M_1 \times M_2 \times M_3$ and $P_{2,3} \subseteq M_2 \times M_3$. The first part of the next proposition is due to Schützenberger [60] for $n = 1$ and to Straubing [67] for the general case.

Proposition 4.1. *Let $L = L_0a_1L_1 \cdots a_nL_n$ be a marked product and let M_i be the syntactic monoid of L_i, for $0 \leqslant i \leqslant n$. Then the Schützenberger product $\diamondsuit_n(M_0, \ldots, M_n)$ recognises L.*

A similar result holds for the p-modular product, for a prime p, by taking $k = \mathbb{F}_p$, the field with p elements [34,37,79].

Proposition 4.2. *Let $L = (L_0a_1L_1 \cdots a_nL_n)_{r,p}$ be a p-modular product and let M_i be the syntactic monoid of L_i, for $0 \leqslant i \leqslant n$. Then the Schützenberger product $\mathbb{F}_p\diamondsuit_n(M_0, \ldots, M_n)$ recognises L.*

In view of Proposition 4.1, a natural question arises: what are the languages recognised by a Schützenberger product? In the Boolean case, the answer was first given by Reutenauer [58] for $n = 2$ and by the author [33] in the general case (see also [80,63]). The case $k = \mathbb{F}_p$ was treated by Weil [79, Theorem 2.2].

Theorem 4.3. *A language is recognised by the Schützenberger product of M_0, $\ldots$, M_n if and only if it belongs to the Boolean algebra generated by the marked products of the form $L_{i_0}a_1L_{i_1} \cdots a_sL_{i_s}$ where $0 \leqslant i_0 < i_1 < \cdots < i_s \leqslant n$ and L_{i_j} is recognised by M_{i_j} for $0 \leqslant j \leqslant s$.*

Theorem 4.4. *A language is recognised by the monoid $\mathbb{F}_p\diamondsuit(M_0, \ldots, M_n)$ if and only if it belongs to the Boolean algebra generated by the p-modular products of the form $(L_{i_0}a_1L_{i_1} \cdots a_sL_{i_s})_{r,p}$ where $0 \leqslant i_0 < i_1 < \cdots < i_s \leqslant n$ and L_{i_j} is recognised by M_{i_j} for $0 \leqslant j \leqslant s$.*

In the Boolean case, it is possible to give an ordered version of Theorem 4.3 [54,44]. Indeed, the (Boolean) Schützenberger product can be ordered by reverse inclusion: $P \leqslant P'$ if and only if for $1 \leqslant i \leqslant j \leqslant n$, $P_{i,j} \supseteq P'_{i,j}$. The corresponding ordered monoid is denoted $\diamondsuit_n^+(M_0, \ldots, M_n)$ and is called the *ordered Schützenberger product* of M_1, $\ldots$, M_n.

Theorem 4.5. *A language is recognised by the ordered Schützenberger product of M_0, $\ldots$, M_n if and only if it belongs to the lattice generated by the marked products of the form $L_{i_0}a_1L_{i_1} \cdots a_sL_{i_s}$ where $0 \leqslant i_0 < i_1 < \cdots < i_s \leqslant n$ and L_{i_j} is recognised by M_{i_j} for $0 \leqslant j \leqslant s$.*

4.3 Algebraic Properties of the Schützenberger Product

It follows from the definition of the Schützenberger product that the map sending a matrix to its diagonal is a morphism π from $k\diamondsuit(M_0, \ldots, M_n)$ to M. The properties of this morphism were first analysed by Straubing [67] and by the author [36,54,44] in the Boolean case and by Weil [80, Corollary 3.6] when $k = \mathbb{F}_p$. See also [4].

Proposition 4.6. *The morphism $\pi : \Diamond(M_0, \ldots, M_n) \to M$ is a $[\![x^\omega y x^\omega \leqslant x^\omega]\!]$-morphism.*

Proposition 4.7. *The morphism $\pi : \mathbb{F}_p\Diamond(M_0, \ldots, M_n) \to M$ is a $\mathbf{LG}_p$-morphism.*

5 Passacaglia: Pumping Properties

The second method to study the product is to use relational morphisms. This technique was initiated by Straubing [68] and later refined in [10,8,36,44,50,54]. We first state the main result under the form of a pumping lemma before turning to a more algebraic formulation.

Let $L = L_0 a_1 L_1 \cdots a_n L_n$ be a marked product of regular languages.

Theorem 5.1. *Let u and v be words of A^* satisfying the following properties:*
(1) *$u^2 \sim_L u$ and*
(2) *for each $i \in \{0, \ldots, n\}$, $u^2 \sim_{L_i} u$ and $uvu \leqslant_{L_i} u$.*
Then for all $x, y \in A^*$, the condition $xuy \in L$ implies $xuvuy \in L$.

Another possible formulation of the theorem is to say that, under the assumptions (1) and (2), L is closed under the rewriting system $u \to uvu$.

We now turn to the algebraic version of this statement. For each i, let L_i be a language of A^*, let $\eta_i : A^* \to M(L_i)$ be its syntactic morphism and let

$$\eta : A^* \to M(L_0) \times M(L_1) \times \cdots \times M(L_n)$$

be the morphism defined by $\eta(u) = (\eta_0(u), \eta_1(u), \ldots, \eta_n(u))$. Finally, let $\mu : A^* \to M(L)$ be the syntactic morphism of L. Theorem 5.1 can be reformulated as a property of the relational morphism (see picture below)

$$\tau = \eta \circ \mu^{-1} : M(L) \to M(L_0) \times M(L_1) \times \cdots \times M(L_n)$$

$$\begin{tikzcd}
& A^* \arrow[dl, "\mu"'] \arrow[dr, "\eta"] & \\
M(L) \arrow[rr, "\tau = \eta \circ \mu^{-1}"] & & M(L_0) \times M(L_1) \times \cdots \times M(L_n)
\end{tikzcd}$$

Theorem 5.2
(1) *The relational morphism τ is a relational $[\![x^\omega y x^\omega \leqslant x^\omega]\!]$-morphism.*
(2) *If the product is unambiguous, it is a relational $[\![x^\omega y x^\omega = x^\omega]\!]$-morphism.*
(3) *If the product is left deterministic, it is a relational $[\![x^\omega y = x^\omega]\!]$-morphism.*
(4) *If the product is right deterministic, it is a relational $[\![y x^\omega = x^\omega]\!]$-morphism.*

A similar result holds for the p-modular product.

Proposition 5.3. *Let $L = (L_0a_1L_1 \cdots a_nL_n)_{r,p}$ be a p-modular product. The relational morphism $\tau : M(L) \to M(L_0) \times \cdots \times M(L_n)$ is a relational $\mathbf{LG}_p$-morphism.*

Theorem 5.2 is often used in the following weaker form.

Corollary 5.4. *The relational morphism $\tau : M(L) \to M(L_0) \times M(L_1) \times \cdots \times M(L_n)$ is an aperiodic relational morphism.*

6 Chaconne: Closure Properties

The results of Section 3 give a description of the smallest Boolean algebra closed under marked product and its variants. The next step would be to characterize all Boolean algebras closed under marked product and its variants. A related problem is to describe the classes of regular languages closed under union and marked product.

Both problems have been solved in the case of a variety of languages, but the description of these results requires an algebraic definition. Let $\mathbf{V}$ be a variety of [ordered] monoids and let $\mathbf{W}$ be a variety of ordered semigroups. The class of all [ordered] monoids M such that there exists a $\mathbf{V}$-relational morphism from M into a monoid of $\mathbf{V}$ is a variety of [ordered] monoids, denoted $\mathbf{W}^{-1}\mathbf{V}$.

6.1 Varieties Closed under Product

Varieties closed under marked products were described by Straubing [66].

Theorem 6.1. *Let $\mathbf{V}$ be a variety of monoids and let $\mathcal{V}$ be the associated variety of languages. For each alphabet A, let $\mathcal{W}(A^*)$ be the smallest Boolean algebra containing $\mathcal{V}(A^*)$ and closed under product. Then $\mathcal{W}$ is a variety and the associated variety of monoids is $\mathbf{A}^{-1}\mathbf{V}$.*

This important result contains Theorem 3.1 as a particular case, when $\mathbf{V}$ is the trivial variety of monoids. Examples of varieties $\mathbf{V}$ satisfying the equality $\mathbf{A}^{-1}\mathbf{V} = \mathbf{V}$ also include the variety of monoids whose groups belong to a given variety of groups.

Theorem 6.1 has been extended to $\mathcal{C}$-varieties in [15, Theorem 4.1].

6.2 Varieties Closed under Modular Product

Finally, let us mention the results of Weil [80]. A set of languages $\mathcal{L}$ of A^* is *closed under p-modular product* if, for any language $L_0, \ldots, L_n \in \mathcal{L}$, for any letter $a_1, \ldots, a_n \in A$ and for any integer r such that $0 \leqslant r < p$, $(L_0a_1L_1 \cdots a_nL_n)_{r,p} \in \mathcal{L}$. A set of languages $\mathcal{L}$ of A^* is *closed under modular product* if it is closed under p-modular product, for each prime p.

Theorem 6.2. *Let p be a prime number, let $\mathbf{V}$ be a variety of monoids and let $\mathcal{V}$ be the associated variety of languages. For each alphabet A, let $\mathcal{W}(A^*)$*

be the smallest Boolean algebra containing $\mathcal{V}(A^*)$ and closed under p-modular product. Then $\mathcal{W}$ is a variety of languages and the associated variety of monoids is $\mathbf{LG}_p^{-1}\mathbf{V}$.

Theorem 6.3. *Let p be a prime number, let $\mathbf{V}$ be a variety of monoids and let $\mathcal{V}$ be the associated variety of languages. For each alphabet A, let $\mathcal{W}(A^*)$ be the smallest Boolean algebra containing $\mathcal{V}(A^*)$ and closed under product and p-modular product. Then $\mathcal{W}$ is a variety of languages and the associated variety of monoids is $\mathbf{LG}_p^{-1}\mathbf{V}$.*

Theorem 6.4. *Let $\mathbf{V}$ be a variety of monoids and let $\mathcal{V}$ be the associated variety of languages. For each alphabet A, let $\mathcal{W}(A^*)$ be the Boolean algebra containing $\mathcal{V}(A^*)$ and closed under modular product. Then $\mathcal{W}$ is a variety of languages and the associated variety of monoids is $\mathbf{LGsol}^{-1}\mathbf{V}$.*

6.3 Polynomial Closure

Let $\mathcal{L}$ be a lattice of languages. The *polynomial closure* of $\mathcal{L}$ is the set of languages that are finite unions of marked products of languages of $\mathcal{L}$. It is denoted $\mathrm{Pol}(\mathcal{L})$. Similarly, the *unambiguous polynomial closure* of $\mathcal{L}$ is the set of languages that are finite unions of unambiguous marked products of languages of $\mathcal{L}$. It is denoted $\mathrm{UPol}(\mathcal{L})$. The *left and right deterministic polynomial closure* are defined analogously, by replacing "unambiguous" by "left [right] deterministic". They are denoted $\mathrm{D}^l\mathrm{Pol}(\mathcal{V})$ $[\mathrm{D}^r\mathrm{Pol}(\mathcal{V})]$.

An algebraic characterization of the polynomial closure of a variety of languages was first given in [51,54]. It was extended to positive varieties in [44].

Theorem 6.5. *Let $\mathbf{V}$ be a variety of* [*ordered*] *monoids and let $\mathcal{V}$ be the associated* [*positive*] *variety of languages. Then $Pol(\mathcal{V})$ is a positive variety of languages and the associated variety of ordered monoids is $[\![x^\omega y x^\omega \leqslant x^\omega]\!]^{-1}\mathbf{V}$.*

Theorem 6.5 has been extended to $\mathcal{C}$-varieties in [49, Theorem 7.2]. For the unambiguous product, one has the following result [32,50,4].

Theorem 6.6. *Let $\mathbf{V}$ be a variety of monoids and let $\mathcal{V}$ be the associated variety of languages. Then $UPol(\mathcal{V})$ is a variety of languages and the associated variety of ordered monoids is $[\![x^\omega y x^\omega = x^\omega]\!]^{-1}\mathbf{V}$.*

For the left (resp. right) deterministic product, similar results hold [32,50].

Theorem 6.7. *Let $\mathbf{V}$ be a variety of monoids and let $\mathcal{V}$ be the associated variety of languages. Then $D^l Pol(\mathcal{V})$ (resp. $D^r Pol(\mathcal{V})$) is a variety of languages, and the associated variety of monoids is $[\![x^\omega y = x^\omega]\!]^{-1}\mathbf{V}$ (resp. $[\![y x^\omega = x^\omega]\!]^{-1}\mathbf{V}$).*

It is known that the smallest nontrivial variety of aperiodic monoids is the variety $\mathbf{J}_1 = [\![xy = yx, x = x^2]\!]$. One can show that $[\![x^\omega y = x^\omega]\!]^{-1}\mathbf{J}_1$ is equal to the variety $\mathbf{R}$ of all $\mathcal{R}$-trivial monoids, which is also defined by the identity $(xy)^\omega x = (xy)^\omega$. This leads to the following characterization [18,13].

Corollary 6.8. *For each alphabet A, $\mathcal{R}(A^*)$ consists of the languages which are disjoint unions of languages of the form $A_0^*a_1A_1^*a_2 \cdots a_nA_n^*$, where $n \geqslant 0$, $a_1, \ldots a_n \in A$ and the A_i's are subsets of A such that $a_i \notin A_{i-1}$, for $1 \leqslant i \leqslant n$.*

A dual result holds for $\mathcal{L}$-trivial monoids. Finally, $[\![x^\omega y x^\omega = x^\omega]\!]^{-1}\mathbf{J}_1 = \mathbf{DA}$, which leads to the description of the languages of **DA** given hereinabove.

6.4 Back to Identities

A general result of [52] permits to give identities defining the varieties of the form $\mathbf{V}^{-1}\mathbf{W}$. In particular, we get the following results.

Theorem 6.9. *Let* $\mathbf{V}$ *be a variety of monoids. Then*

(1) $\mathbf{A}^{-1}\mathbf{V}$ *is defined by the identities of the form $x^{\omega+1} = x^\omega$, where x is a profinite word such that* $\mathbf{V}$ *satisfies the identity $x = x^2$.*

(2) $[\![x^\omega y x^\omega = x^\omega]\!]^{-1}\mathbf{V}$ *is defined by the identities of the form $x^\omega y x^\omega = x^\omega$, where x, y are profinite words such that* $\mathbf{V}$ *satisfies the identity $x = y = x^2$.*

(3) $[\![x^\omega y x^\omega \leqslant x^\omega]\!]^{-1}\mathbf{V}$ *is defined by the identities of the form $x^\omega y x^\omega \leqslant x^\omega$, where x, y are profinite words such that* $\mathbf{V}$ *satisfies the identity $x = y = x^2$.*

7 Hierarchies and Bridges

The Boolean algebra $\mathcal{BL}$ generated by a lattice $\mathcal{L}$ is called its *Boolean closure*. In particular, $\mathcal{B}\mathrm{Pol}(\mathcal{L})$ denotes the Boolean closure of $\mathrm{Pol}(\mathcal{L})$.

Concatenation hierarchies are defined by alternating Boolean operations and polynomial operations (union and marked product). More precisely, let $\mathcal{L}$ be a set of regular languages (or more generally, a class of languages). The *concatenation hierarchy* built on $\mathcal{L}$ is the sequence $\mathcal{L}_n$ defined inductively as follows[1]: $\mathcal{L}_0 = \mathcal{L}$ and, for each $n \geqslant 0$:

(1) $\mathcal{L}_{2n+1}$ is the polynomial closure of the level $2n$,

(2) $\mathcal{L}_{2n+2}$ is the Boolean closure of the level $2n+1$.

The next results summarize the results of [5,6,54].

Proposition 7.1. *If $\mathcal{L}$ is a lattice of regular languages, then each even level is a lattice of regular languages and each odd level is a Boolean algebra. Further, if $\mathcal{L}$ is closed under quotients, then every level is closed under quotients.*

Since the polynomial closure of a $\mathcal{C}$-variety of languages is a positive $\mathcal{C}$-variety of languages [49, Theorem 6.2], a similar result holds for $\mathcal{C}$-varieties.

Proposition 7.2. *If $\mathcal{L}$ is a $\mathcal{C}$-variety of languages, then each even level is a positive $\mathcal{C}$-variety of languages and each odd level is a $\mathcal{C}$-variety of languages.*

[1] In the literature, concatenation hierarchies are usually indexed by half integers, but it seems simpler to use integers.

For instance, the *Straubing-Thérien' hierarchy* $\mathcal{V}_n$ [74,67,69] is built on the trivial Boolean algebra $\mathcal{V}_0 = \{\emptyset, A^*\}$. The starting point of Brzozowski's "*dot-depth*" hierarchy $\mathcal{B}_n$ [12] was originally defined as the Boolean algebra of finite and cofinite languages but it was later suggested to start with the Boolean algebra

$$\mathcal{B}_0(A^*) = \{FA^*G \cup H \mid F,\ G,\ H \text{ are finite languages}\}$$

This suggestion was motivated by Theorem 7.4 below.

Another series of concatenation hierarchies is obtained as follows. Let $\mathbf{H}$ be a variety of groups and let $\mathcal{H}$ be the associated variety of languages. The concatenation hierarchy built on $\mathcal{H}$ is denoted by $\mathcal{H}_n$ and these hierarchies are called *group hierarchies.*

It is not immediate to see that all these hierarchies do not collapse. This was first proved by Brzozowski and Knast [14] for the dot-depth hierarchy, but the result also holds for the other hierarchies [26].

Theorem 7.3. *The Straubing-Thérien' hierarchy, the dot-depth hierarchy and the group hierarchies are infinite.*

Let $\mathbf{V}_n$ be the variety of monoids corresponding to $\mathcal{V}_n$ and let $\mathbf{B}_n$ be the variety of semigroups corresponding to $\mathcal{B}_n$. There is a nice algebraic connection between $\mathbf{V}_n$ and $\mathbf{B}_n$, discovered by Straubing [69]. Given a variety of [ordered] monoids $\mathbf{V}$ and a variety of monoids [semigroups] $\mathbf{W}$, let $\mathbf{V} * \mathbf{W}$ be the variety of [ordered] monoids generated by the semidirect products $M * N$ with $M \in \mathbf{V}$ and $N \in \mathbf{W}$.

Theorem 7.4. *The equality* $\mathbf{B}_n = \mathbf{V}_n * \mathbf{LI}$ *holds for each* $n \geqslant 0$.

There is a similar bridge between $\mathbf{V}_n$ and $\mathbf{H}_n$ for each variety of groups $\mathbf{H}$ [43,44].

Theorem 7.5. *The equality* $\mathbf{H}_n = \mathbf{V}_n * \mathbf{H}$ *holds for each* $n \geqslant 0$.

It is still an outstanding open problem to know whether there is an algorithm to compute the concatenation level of a given regular language. Here is a brief summary of the known results. Let us start with the level 1 [26,39,41,56]. Let $\mathbf{G}$ be the variety of all groups.

Theorem 7.6. *The following relations hold:* $\mathbf{V}_1 = [\![x \leqslant 1]\!]$, $\mathbf{B}_1 = [\![x^\omega y x^\omega \leqslant x^\omega]\!]$ *and* $\mathbf{G}_1 = [\![x^\omega \leqslant 1]\!]$. *In particular, these varieties are decidable.*

The languages of $\mathcal{G}_1$ are also known to be the open regular sets for the progroup topology [26]. Extensions of this result to the varieties $\mathbf{H}_1$ where $\mathbf{H}$ is a variety of groups is the topic of intensive research. See in particular Steinberg's article [64].

The first decidability result for the level 2 was obtained by Simon [62].

Theorem 7.7. *A language belongs to* $\mathcal{V}_2$ *if and only if its syntactic monoid is* $\mathcal{J}$*-trivial.*

The corresponding result for $\mathcal{B}_2$ is due to Knast [24,25]

Theorem 7.8. *A language belongs to $\mathcal{B}_2$ if and only if its syntactic semigroup satisfies the identity*

$$(x^\omega py^\omega qx^\omega)^\omega x^\omega py^\omega sx^\omega(x^\omega ry^\omega sx^\omega)^\omega = (x^\omega py^\omega qx^\omega)^\omega(x^\omega ry^\omega sx^\omega)^\omega.$$

The corresponding result for $\mathcal{G}_2$ has a long story, related in detail in [38], where several other characterizations can be found.

Theorem 7.9. *A language belongs to $\mathcal{G}_2$ if and only if in its syntactic monoid, the submonoid generated by the idempotents is $\mathcal{J}$-trivial.*

Theorem 7.9 shows that $\mathcal{G}_2$ is decidable. Again, there is a lot of ongoing work to try to extend this result to varieties of the form $\mathcal{H}_2$. See in particular [7].

Since level 3 is the polynomial closure of level 2, Theorem 6.5 can be applied. One gets in particular the following decidability result [54]. Recall that the *content* of a word is the set of letters occurring in this word.

Theorem 7.10. *A language belongs to $\mathcal{V}_3$ if and only if its syntactic ordered monoid satisfies the identities $x^\omega yx^\omega \leqslant x^\omega$ for all profinite words x, y with the same content.*

The corresponding problem for $\mathcal{B}_3$ is studied in [20,22,56]. In fact, Theorem 7.4 can be used to prove the following more general decidability result [56,69].

Theorem 7.11. *For every integer n, the variety $\mathbf{B}_n$ is decidable if and only if $\mathbf{V}_n$ is decidable.*

It is still an open problem to know whether a similar reduction exists for the hierarchy $\mathbf{G}_n$.

For the level 4, several partial results are known [48,70] and several conjectures have been formulated and then disproved [54,64,55]. Due to the lack of space, we will not detail these results here. Some partial results are also known for the level 5 [21].

8 Harmony with Logic

One of the reasons why the decidability problem is particularly appealing is its close connection with finite model theory, first explored by Büchi in the early sixties. Büchi's logic comprises a relation symbol $<$ and, for each letter $a \in A$, a unary predicate symbol $\mathbf{a}$. The set $\mathbf{FO}[<]$ of first order formulas is built in the usual way by using these symbols, the equality symbol, first order variables, Boolean connectives and quantifiers.

A word u is represented as a structure $(\mathrm{Dom}(u), (\mathbf{a})_{a\in A}, <)$ where $\mathrm{Dom}(u) = \{1, \ldots, |u|\}$ and $\mathbf{a} = \{i \in \mathrm{Dom}(u) \mid u(i) = a\}$. The binary relation symbol $<$ is interpreted as the usual order. Thus, if $u = abbaab$, $\mathrm{Dom}(u) = \{1, \ldots, 6\}$, $\mathbf{a} = \{1, 4, 5\}$ and $\mathbf{b} = \{2, 4, 6\}$. Formulas can now be interpreted on words. For instance, the sentence

$$\varphi = \exists x\ \exists y\ \big((x < y) \wedge (\mathbf{a}x) \wedge (\mathbf{b}y)\big)$$

means "there exist two integers $x < y$ such that, in u, the letter in position x is an a and the letter in position y is a b". Therefore, the set of words satisfying φ is $A^*aA^*bA^*$. More generally, the language defined by a sentence φ is the set of words u such that φ satisfies u. The connection with star-free languages was established by McNaughton and Papert [27].

Theorem 8.1. *A language is* $\mathbf{FO}[<]$*-definable if and only if it is star-free.*

Thomas [77] (see also [31]) refined this result by showing that the concatenation hierarchy of star-free languages corresponds, level by level, to the Σ_n-hierarchy, defined inductively as follows:

(1) Σ_0 consists of the quantifier-free formulas.
(2) Σ_n consists of the formulas of the form $\exists^* \; \forall^* \; \exists^* \; \cdots \; \varphi$ with n alternating blocks of quantifiers and φ quantifier-free.
(3) $\mathcal{B}\Sigma_n$ denotes the class of formulas that are Boolean combinations of Σ_n-formulas.

For instance, $\exists x_1 \exists x_2 \forall x_3 \forall x_4 \forall x_5 \exists x_6 \, \varphi$, where φ is quantifier free, is in Σ_3. The next theorem is due to Thomas [77] (see also [31,40]).

Theorem 8.2

(1) *A language is* $\Sigma_n[<]$*-definable if and only if it belongs to* $\mathcal{V}_{2n-1}$.
(2) *A language is* $\mathcal{B}\Sigma_n[<]$*-definable if and only if it belongs to* $\mathcal{V}_{2n}$.

A slightly expanded logic is required for the dot-depth hierarchy. Let min [max] be a predicate symbol interpreted as the minimum [maximum] of the domain and let P [S] be a relation symbol interpreted as the predecessor [successor] relation. Let **Loc** be the signature $\{\min, \max, S, P\} \cup \{(\mathbf{a})_{a \in A}\}$.

Theorem 8.3

(1) *A language is* $\Sigma_n[\mathbf{Loc}]$*-definable if and only if it belongs to* $\mathcal{B}_{2n-1}$.
(2) *A language is* $\mathcal{B}\Sigma_n[\mathbf{Loc}]$*-definable if and only if it belongs to* $\mathcal{B}_{2n}$.

Thus deciding whether a language has level n is equivalent to a very natural problem in finite model theory.

9 Other Variations, Recent Advances

Some specialized topics require even more sophisticated algebraic tools, like the kernel category of a morphism. This is the case for instance for the bideterministic product [9,10,11] or for the marked product of two languages [4].

Another topic that we did not mention at all, but which is highly interesting, is the extension of these results to infinite words or even to words over ordinals or linear orders.

I would like to conclude with a recent result, which opens a new research direction. We have given in Section 6 various closure properties for varieties or even for $\mathcal{C}$-varieties. The next result of Branco and the author [8] is much more general.

Theorem 9.1. *If $\mathcal{L}$ is a lattice of languages closed under quotients, then $Pol(\mathcal{L})$ is defined by the set of equations of the form $x^\omega y x^\omega \leqslant x^\omega$, where x, y are profinite words such that the equations $x = x^2$ and $y \leqslant x$ are satisfied by $\mathcal{L}$.*

Work is in progress to extend the other results of Section 6 to this more general setting. The difficulty stems from the fact that definitions like $\mathbf{V}^{-1}\mathbf{W}$ are no longer available in this context and one has to work directly on profinite identities.

References

1. Almeida, J.: Finite semigroups and universal algebra. World Scientific Publishing Co. Inc., River Edge (1994); Translated from the, Portuguese original and revised by the author
2. Almeida, J.: Finite semigroups: an introduction to a unified theory of pseudovarieties. In: Gomes, G.M.S., Pin, J.-É., Silva, P. (eds.) Semigroups, Algorithms, Automata and Languages, pp. 3–64. World Scientific, Singapore (2002)
3. Almeida, J.: Profinite semigroups and applications. In: Structural Theory of Automata, Semigroups, and Universal Algebra. NATO Sci. Ser. II Math. Phys. Chem., vol. 207, pp. 1–45. Springer, Dordrecht (2005)
4. Almeida, J., Margolis, S., Steinberg, B., Volkov, M.: Representation theory of finite semigroups, semigroup radicals and formal language theory. Trans. Amer. Math. Soc. 361(3), 1429–1461 (2009)
5. Arfi, M.: Polynomial operations on rational languages. In: Brandenburg, F.J., Wirsing, M., Vidal-Naquet, G. (eds.) STACS 1987. LNCS, vol. 247, pp. 198–206. Springer, Heidelberg (1987)
6. Arfi, M.: Opérations polynomiales et hiérarchies de concaténation. Theoret. Comput. Sci. 91(1), 71–84 (1991)
7. Auinger, K., Steinberg, B.: Varieties of finite supersolvable groups with the M. Hall property. Math. Ann. 335(4), 853–877 (2006)
8. Branco, M.J., Pin, J.-É.: Equations for the polynomial closure. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 115–126. Springer, Heidelberg (2009)
9. Branco, M.J.J.: The kernel category and variants of the concatenation product. Internat. J. Algebra Comput. 7(4), 487–509 (1997)
10. Branco, M.J.J.: Two algebraic approaches to variants of the concatenation product. Theoret. Comput. Sci. 369(1-3), 406–426 (2006)
11. Branco, M.J.J.: Deterministic concatenation product of languages recognized by finite idempotent monoids. Semigroup Forum 74(3), 379–409 (2007)
12. Brzozowski, J.A.: Hierarchies of aperiodic languages. RAIRO Inform. Théor. 10(R-2), 33–49 (1976)
13. Brzozowski, J.A., Fich, F.E.: Languages of $\mathcal{R}$-trivial monoids. J. Comput. System Sci. 20(1), 32–49 (1980)
14. Brzozowski, J.A., Knast, R.: The dot-depth hierarchy of star-free languages is infinite. J. Comput. System Sci. 16(1), 37–55 (1978)
15. Chaubard, L., Pin, J.-E., Straubing, H.: Actions, Wreath Products of $\mathcal{C}$-varieties and Concatenation Product. Theoret. Comput. Sci. 356, 73–89 (2006)
16. Cho, S., Huỳnh, D.T.: Finite-automaton aperiodicity is PSPACE-complete. Theoret. Comput. Sci. 88(1), 99–116 (1991)

17. Cohen, R.S., Brzozowski, J.A.: On Star-Free Events. In: Kinariwala, B.K., Kuo, F.F. (eds.) Proceedings of the Hawaii International Conference on System Sciences, Honolulu, HI, January 29-31, pp. 1–4 (1968)
18. Eilenberg, S.: Automata, Languages and Machines, vol. B. Academic Press, New York (1976)
19. Gehrke, M., Grigorieff, S., Pin, J.-É.: Duality and equational theory of regular languages. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 246–257. Springer, Heidelberg (2008)
20. Glaßer, C., Schmitz, H.: Languages of dot-depth 3/2. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 555–566. Springer, Heidelberg (2000)
21. Glaßer, C., Schmitz, H.: Level 5/2 of the straubing-thérien hierarchy for two-letter alphabets. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 251–261. Springer, Heidelberg (2002)
22. Glaßer, C., Schmitz, H.: Languages of dot-depth 3/2. Theory Comput. Syst. 42(2), 256–286 (2008)
23. Kleene, S.C.: Representation of events in nerve nets and finite automata. Automata studies, pp. 3–41. Princeton University Press, Princeton (1956); Annals of mathematics studies, no. 34
24. Knast, R.: A semigroup characterization of dot-depth one languages. RAIRO Inform. Théor. 17(4), 321–330 (1983)
25. Knast, R.: Some theorems on graph congruences. RAIRO Inform. Théor. 17(4), 331–342 (1983)
26. Margolis, S., Pin, J.-E.: Products of group languages. In: Budach, L. (ed.) FCT 1985. LNCS, vol. 199, pp. 285–299. Springer, Heidelberg (1985)
27. McNaughton, R., Papert, S.: Counter-free automata. The MIT Press, Cambridge (1971); With an appendix by William Henneman, M.I.T. Research Monograph, no. 65
28. Meyer, A.R.: A note on star-free events, J. Assoc. Comput. Mach. 16, 220–225 (1969)
29. Péladeau, P.: Sur le produit avec compteur modulo un nombre premier. RAIRO Inform. Théor. Appl. 26(6), 553–564 (1992)
30. Perrin, D.: Finite automata. In: Handbook of Theoretical Computer Science, vol. B, pp. 1–57. Elsevier, Amsterdam (1990)
31. Perrin, D., Pin, J.-E.: First order logic and star-free sets. J. Comput. System Sci. 32, 393–406 (1986)
32. Pin, J.-E.: Propriétés syntactiques du produit non ambigu. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 483–499. Springer, Heidelberg (1980)
33. Pin, J.-E.: Hiérarchies de concaténation. RAIRO Informatique Théorique 18, 23–46 (1984)
34. Pin, J.-E.: Finite group topology and p-adic topology for free monoids. In: Brauer, W. (ed.) ICALP 1985. LNCS, vol. 194, pp. 445–455. Springer, Heidelberg (1985)
35. Pin, J.-E.: Varieties of formal languages. North Oxford, London and Plenum, New-York (1986) (Traduction de Variétés de langages formels)
36. Pin, J.-E.: A property of the Schützenberger product. Semigroup Forum 35, 53–62 (1987)
37. Pin, J.-E.: Topologies for the free monoid. J. of Algebra 137, 297–337 (1991)
38. Pin, J.-E.: PG = BG, a success story. In: Fountain, J. (ed.) NATO Advanced Study Institute Semigroups, Formal Languages and Groups, pp. 33–47. Kluwer academic publishers, Dordrecht (1995)

39. Pin, J.-E.: A variety theorem without complementation. Russian Mathematics (Iz. VUZ) 39, 80–90 (1995)
40. Pin, J.-E.: Logic, Semigroups and Automata on Words. Annals of Mathematics and Artificial Intelligence 16, 343–384 (1996)
41. Pin, J.-E.: Polynomial closure of group languages and open sets of the Hall topology. Theoret. Comput. Sci. 169, 185–200 (1996)
42. Pin, J.-E.: Syntactic semigroups. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, ch. 10, pp. 679–746. Springer, Heidelberg (1997)
43. Pin, J.-É.: Bridges for concatenation hierarchies. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 431–442. Springer, Heidelberg (1998)
44. Pin, J.-E.: Algebraic tools for the concatenation product. Theoret. Comput. Sci. 292, 317–342 (2003)
45. Pin, J.-E.: Profinite methods in automata theory. In: Albers, S., Marion, J.-Y. (eds.) 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009), pp. 31–50. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl (2009)
46. Pin, J.-E., Sakarovitch, J.: Some operations and transductions that preserve rationality. In: Cremers, A.B., Kriegel, H.-P. (eds.) GI-TCS 1983. LNCS, vol. 145, pp. 277–288. Springer, Heidelberg (1982)
47. Pin, J.-E., Sakarovitch, J.: Une application de la représentation matricielle des transductions. Theoret. Comput. Sci. 35, 271–293 (1985)
48. Pin, J.-E., Straubing, H.: Monoids of upper triangular boolean matrices. In: Semigroups (Szeged 1981). Colloq. Math. Soc. János Bolyai, vol. 39, pp. 259–272. North-Holland, Amsterdam (1985)
49. Pin, J.-E., Straubing, H.: Some results on $\mathcal{C}$-varieties. Theoret. Informatics Appl. 39, 239–262 (2005)
50. Pin, J.-E., Straubing, H., Thérien, D.: Locally trivial categories and unambiguous concatenation. J. of Pure and Applied Algebra 52, 297–311 (1988)
51. Pin, J.-E., Weil, P.: Polynomial closure and unambiguous product. In: Fülöp, Z. (ed.) ICALP 1995. LNCS, vol. 944, pp. 348–359. Springer, Heidelberg (1995)
52. Pin, J.-E., Weil, P.: Profinite semigroups, Mal'cev products and identities. J. of Algebra 182, 604–626 (1996)
53. Pin, J.-E., Weil, P.: A Reiterman theorem for pseudovarieties of finite first-order structures. Algebra Universalis 35, 577–595 (1996)
54. Pin, J.-E., Weil, P.: Polynomial closure and unambiguous product. Theory Comput. Systems 30, 383–422 (1997)
55. Pin, J.-E., Weil, P.: A conjecture on the concatenation product. Theoret. Informatics Appl. 35, 597–618 (2001)
56. Pin, J.-E., Weil, P.: The wreath product principle for ordered semigroups. Communications in Algebra 30, 5677–5713 (2002)
57. Reiterman, J.: The Birkhoff theorem for finite algebras. Algebra Universalis 14(1), 1–10 (1982)
58. Reutenauer, C.: Sur les variétés de langages et de monoïdes. In: Weihrauch, K. (ed.) GI-TCS 1979. LNCS, vol. 67, pp. 260–265. Springer, Heidelberg (1979)
59. Sakarovitch, J.: Elements of automata theory. Cambridge University Press, Cambridge (2009); Translated from The French original by Reuben Thomas
60. Schützenberger, M.-P.: On finite monoids having only trivial subgroups. Information and Control 8, 190–194 (1965)
61. Schützenberger, M.-P.: Sur le produit de concaténation non ambigu. Semigroup Forum 18, 331–340 (1976)

62. Simon, I.: Piecewise testable events. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 214–222. Springer, Heidelberg (1975)
63. Simon, I.: The product of rational languages. In: Lingas, A., Carlsson, S., Karlsson, R. (eds.) ICALP 1993. LNCS, vol. 700, pp. 430–444. Springer, Heidelberg (1993)
64. Steinberg, B.: Polynomial closure and topology. Internat. J. Algebra Comput. 10(5), 603–624 (2000)
65. Stern, J.: Characterizations of some classes of regular events. Theoret. Comput. Sci. 35(1), 17–42 (1985)
66. Straubing, H.: Families of recognizable sets corresponding to certain varieties of finite monoids. J. Pure Appl. Algebra 15(3), 305–318 (1979)
67. Straubing, H.: A generalization of the Schützenberger product of finite monoids. Theoret. Comput. Sci. 13(2), 137–150 (1981)
68. Straubing, H.: Relational morphisms and operations on recognizable sets. RAIRO Inform. Théor. 15(2), 149–159 (1981)
69. Straubing, H.: Finite semigroup varieties of the form $\mathbf{V} * \mathbf{D}$. J. Pure Appl. Algebra 36(1), 53–94 (1985)
70. Straubing, H.: Semigroups and languages of dot-depth two. Theoret. Comput. Sci. 58(1-3), 361–378 (1988); Thirteenth International Colloquium on Automata, Languages and Programming (Rennes, 1996)
71. Straubing, H.: On logical descriptions of regular languages. In: Rajsbaum, S. (ed.) LATIN 2002. LNCS, vol. 2286, pp. 528–538. Springer, Heidelberg (2002)
72. Tesson, P., Therien, D.: Diamonds Are Forever: The Variety DA. In: Semigroups, Algorithms, Automata and Languages, Coimbra, Portugal, pp. 475–500. World Scientific, Singapore (2001)
73. Thérien, D.: Languages of Nilpotent and Solvable Groups (Extended Abstract). In: Maurer, H.A. (ed.) ICALP 1979. LNCS, vol. 71, pp. 616–632. Springer, Heidelberg (1979)
74. Thérien, D.: Classification of finite monoids: the language approach. Theoret. Comput. Sci. 14(2), 195–208 (1981)
75. Thérien, D.: Sur les monoïdes dont tous les groupes sont résolubles. Semigroup Forum 26(1-2), 89–101 (1983)
76. Thérien, D.: A language theoretic interpretation of the Schützenberger representations with applications to certain varieties of languages. Semigroup Forum 28(1-3), 235–248 (1984)
77. Thomas, W.: Classifying regular events in symbolic logic. J. Comput. System Sci. 25(3), 360–376 (1982)
78. Weil, P.: An extension of the Schützenberger product. In: Lattices, Semigroups, and Universal Algebra, Lisbon, pp. 315–321. Plenum, New York (1988)
79. Weil, P.: Products of languages with counter. Theoret. Comput. Sci. 76, 251–260 (1990)
80. Weil, P.: Closure of varieties of languages under products with counter. J. Comput. System Sci. 45, 316–339 (1992)

Some Combinatorial Applications of Gröbner Bases

Lajos Rónyai[1,2,⋆] and Tamás Mészáros[3]

[1] Computer and Automation Research Institute, Hungarian Academy of Sciences
[2] Institute of Mathematics, Budapest University of Technology and Economics
lajos@ilab.sztaki.hu
[3] Department of Mathematics, Central European University
Meszaros_Tamas@ceu_budapest.edu

Abstract. Let $\mathbb{F}$ be a field, $V \subseteq \mathbb{F}^n$ be a (combinatorially interesting) finite set of points. Several important properties of V are reflected by the polynomial functions on V. To study these, one often considers $I(V)$, the vanishing ideal of V in the polynomial ring $\mathbb{F}[x_1, \dots, x_n]$. Gröbner bases and standard monomials of $I(V)$ appear to be useful in this context, leading to structural results on V.

Here we survey some work of this type. At the end of the paper a new application of this kind is presented: an algebraic characterization of shattering-extremal families and a fast algorithm to recognize them.

Keywords: Gröbner basis, standard monomial, lexicographic order, vanishing ideal, Hilbert function, inclusion matrix, rank formula, combinatorial Nullstellensatz, S-extremal set family.

1 Introduction

Throughout the paper n will be a positive integer, and $[n]$ stands for the set $\{1, 2, \dots, n\}$. The set of all subsets of $[n]$ is denoted by $2^{[n]}$. Subsets of $2^{[n]}$ are called *set families* or *set systems*. Let $\binom{[n]}{m}$ denote the family of all *m-subsets* of $[n]$ (subsets which have cardinality m), and $\binom{[n]}{\le m}$ is the family of those subsets that have at most m elements. $\mathbb{N}$ denotes the set of the nonnegative integers, $\mathbb{Z}$ is the set of integers, $\mathbb{Q}$ is the field of rational numbers, and $\mathbb{F}_p$ is the field of p elements, where p is a prime.

Let $\mathbb{F}$ be a field. As usual, we denote by $\mathbb{F}[x_1, \dots, x_n] = \mathbb{F}[\mathbf{x}]$ the ring of polynomials in variables $x_1, \dots, x_n$ over $\mathbb{F}$. To shorten our notation, we write $f(\mathbf{x})$ for $f(x_1, \dots, x_n)$. Vectors of length n are denoted by boldface letters, for example $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}^n$. If $\mathbf{w} \in \mathbb{N}^n$, we write $\mathbf{x}^{\mathbf{w}}$ for $x_1^{w_1} \dots x_n^{w_n} \in \mathbb{F}[\mathbf{x}]$. For a subset $M \subseteq [n]$, the monomial x_M is $\prod_{i \in M} x_i$ (and $x_\emptyset = 1$).

Suppose that $V \subseteq \mathbb{F}^n$. Then the *vanishing ideal* $I(V)$ *of* V consists of the polynomials in $\mathbb{F}[\mathbf{x}]$, which, as functions, vanish on V. In our applications, we consider finite sets V, and use the Gröbner bases, and standard monomials of $I(V)$ (see the next subsection for the definitions) to prove claims on V.

⋆ Research supported in part by OTKA grants NK72845, K 77476 and K 77778.

Let $\mathbf{v}_F \in \{0,1\}^n$ denote the *characteristic vector of a set* $F \subseteq [n]$, that is, the ith coordinate of $\mathbf{v}_F$ is 1 iff $i \in F$. For a system of sets $\mathcal{F} \subseteq 2^{[n]}$, let us put $V_{\mathcal{F}}$ for the set of the characteristic vectors of elements of $\mathcal{F}$. By $I(\mathcal{F})$ we understand the vanishing ideal $I(V_{\mathcal{F}})$, as it will make no confusion.

In Sect. 2 we collected some basic facts about Gröbner bases and related notions, such as standard monomials, reduction and Hilbert functions. Section 3 is devoted to the complete uniform families and their extensions. Here we discuss results describing the Gröbner bases and standard monomials of the ideals $I(\mathcal{F})$, where $\mathcal{F}$ is a complete uniform family $\binom{[n]}{d}$ for some $0 \leq d \leq n$. We outline some combinatorial applications of these results, including an extension of Wilson's rank formula. Generalizations and extensions are also considered. Section 4 gives a brief explanation of the lex game method, which gives a powerful technique to determine lex standard monomials both in theory and practice. In Sect. 5 we consider ideals and Gröbner bases attached to more complex objects, such as partitions, permutations and graph colorings. The latter topic is particularly rich in results involving polynomial ideals. Section 6 briefly introduces a powerful algebraic technique of combinatorics, the combinatorial Nullstellensatz by Noga Alon, together with the resulting non-vanishing theorem. The last section gives an algebraic characterization of shattering-extremal set families. The characterization involves Gröbner bases and, together with the lex game method, it provides an efficient algorithm for recognizing shattering-extremal families.

Ideals I of $\mathbb{F}[\mathbf{x}]$ generated by monomials are perhaps the most important objects in algebraic combinatorics. Their study, initiated by Stanley, has led to some spectacular results, in particular, in the area of simplicial complexes and convex polytopes. Gröbner basis methods are also applicable there. In this paper we avoid the area of monomial ideals, as there are many excellent treatments of this subject. We refer the interested reader to the recent volume of Herzog and Hibi [30], and the sources cited therein.

2 Gröbner Bases, Standard Monomials and Hilbert Functions

We recall now some basic facts concerning Gröbner bases in polynomial rings over fields. For details we refer to [11], [12], [13], [14], and [2].

A total order $\prec$ on the monomials composed from variables $x_1, x_2, \ldots, x_n$ is a *term order*, if 1 is the minimal element of $\prec$, and $\prec$ is compatible with multiplication with monomials (if m_1, m_2, m_3 are monomials, $m_1 \prec m_2$, then $m_1 m_3 \prec m_2 m_3$). Two important term orders are the *lexicographic* (*lex* for short) and the *degree compatible lexicographic* (*deglex*) orders. We have $\mathbf{x}^{\mathbf{w}} \prec_{\text{lex}} \mathbf{x}^{\mathbf{u}}$ if and only if $w_i < u_i$ holds for the smallest index i such that $w_i \neq u_i$. As for deglex, we have that a monomial of smaller degree is smaller in deglex, and among monomials of the same degree lex decides the order. Also in general, $\prec$ is *degree compatible*, if $\deg(\mathbf{x}^{\mathbf{w}}) < \deg(\mathbf{x}^{\mathbf{u}})$ implies $\mathbf{x}^{\mathbf{w}} \prec \mathbf{x}^{\mathbf{u}}$.

The *leading monomial* $\mathrm{lm}(f)$ of a nonzero polynomial $f \in \mathbb{F}[\mathbf{x}]$ is the largest monomial (with respect to $\prec$) which appears with nonzero coefficient in f, when written as the usual linear combination of monomials. We denote the set of all leading monomials of polynomials of a given ideal $I \trianglelefteq \mathbb{F}[\mathbf{x}]$ by $\mathrm{Lm}\,(I) = \{\mathrm{lm}(f) : f \in I\}$, and we simply call them the *leading monomials of* I.

A monomial is called a *standard monomial* of I, if it is not a leading monomial of any $f \in I$. Let $\mathrm{Sm}\,(I)$ denote the set of standard monomials of I. Obviously, a divisor of a standard monomial is again in $\mathrm{Sm}\,(I)$.

A finite subset $G \subseteq I$ is a *Gröbner basis* of I, if for every $f \in I$ there exists a $g \in G$ such that $\mathrm{lm}(g)$ divides $\mathrm{lm}(f)$. It is not hard to verify that G is actually a basis of I, that is, G generates I as an ideal of $\mathbb{F}[\mathbf{x}]$. It is a fundamental fact that every nonzero ideal I of $\mathbb{F}[\mathbf{x}]$ has a Gröbner basis.

A Gröbner basis $G \subseteq I$ is *reduced*, if for all $g \in G$, the *leading coefficient* of g (i.e. the coefficient of $\mathrm{lm}(g)$) is 1, and $g \neq h \in G$ implies that no nonzero monomial in g is divisible by $\mathrm{lm}(h)$. For any fixed term order and any nonzero ideal of $\mathbb{F}[\mathbf{x}]$ there exists a unique reduced Gröbner basis. A Gröbner basis is *universal*, if it is a Gröbner basis for every term order $\prec$ on the monomials.

Suppose that $f \in \mathbb{F}[\mathbf{x}]$ contains a monomial $\mathbf{x}^{\mathbf{w}} \cdot \mathrm{lm}(g)$, where g is some other polynomial with leading coefficient c. Then we can *reduce* f *with* g (and obtain $\hat{f}$), that is, we can replace $\mathbf{x}^{\mathbf{w}} \cdot \mathrm{lm}(g)$ in f with $\mathbf{x}^{\mathbf{w}} \cdot \big(\mathrm{lm}(g) - \frac{1}{c}g\big)$. Clearly if $g \in I$, then f and $\hat{f}$ represent the same coset in $\mathbb{F}[\mathbf{x}]/I$. Also note that $\mathrm{lm}\big(\mathbf{x}^{\mathbf{w}} \cdot \big(\mathrm{lm}(g) - \frac{1}{c}g\big)\big) \prec \mathbf{x}^{\mathbf{w}} \cdot \mathrm{lm}(g)$. As $\prec$ is a well founded order, this guarantees that if we reduce f repeatedly with a set of polynomials G, then we end up with a *reduced* $\hat{f}$ in finitely many steps, that is a polynomial such that none of its monomials is divisible by any $\mathrm{lm}(g)$ $(g \in G)$.

If G is a Gröbner basis of an ideal I, then it can be shown that the reduction of any polynomial with G is unique. It follows that for a nonzero ideal I the set $\mathrm{Sm}\,(I)$ is a linear basis of the $\mathbb{F}$-vector space $\mathbb{F}[\mathbf{x}]/I$. If $I(V)$ is a vanishing ideal of a finite set V of points in $\mathbb{F}^n$, then $\mathbb{F}[\mathbf{x}]/I(V)$ can be interpreted as the space of functions $V \to \mathbb{F}$. An immediate consequence is that the number of standard monomials of $I(V)$ is $|V|$. In particular, for every family of sets we have $|\mathcal{F}| = |\mathrm{Sm}\,(I(\mathcal{F}))|$.

Another property of the standard monomials of $I(\mathcal{F})$ we will meet several times: for an arbitrary set family $\mathcal{F}$, one has $x_i^2 - x_i \in I(\mathcal{F})$, therefore all the elements of $\mathrm{Sm}\,(I(\mathcal{F}))$ are square-free monomials.

We write $\mathbb{F}[\mathbf{x}]_{\leq m}$ for the vector space of polynomials over $\mathbb{F}$ with degree at most m. Similarly, if $I \trianglelefteq \mathbb{F}[\mathbf{x}]$ is an ideal, then $I_{\leq m} = I \cap \mathbb{F}[\mathbf{x}]_{\leq m}$ is the linear subspace of polynomials from I with degree at most m. The *Hilbert function* of the $\mathbb{F}$-algebra $\mathbb{F}[\mathbf{x}]/I$ is $H_I : \mathbb{N} \to \mathbb{N}$, where

$$H_I(m) = \dim_{\mathbb{F}} \left(\mathbb{F}[\mathbf{x}]_{\leq m} / I_{\leq m}\right).$$

Let $\prec$ be any degree compatible term order (deglex for instance). One can easily see that the set of standard monomials with respect to $\prec$ of degree at most m forms a linear basis of $\mathbb{F}[\mathbf{x}]_{\leq m} / I_{\leq m}$. Hence we can obtain $H_I(m)$ by determining the set $\mathrm{Sm}\,(I)$ with respect to any degree compatible term ordering.

In the combinatorial literature $H_{I(\mathcal{F})}(m)$ is usually given in terms of inclusion matrices. For two families $\mathcal{F}, \mathcal{G} \subseteq 2^{[n]}$ the *inclusion matrix* $I(\mathcal{F}, \mathcal{G})$ is a matrix of size $|\mathcal{F}| \times |\mathcal{G}|$, whose rows and columns are indexed by the elements of $\mathcal{F}$ and $\mathcal{G}$, respectively. The entry at position (F, G) is 1, if $G \subseteq F$, and 0 otherwise ($F \in \mathcal{F}, G \in \mathcal{G}$). It is a simple matter to verify that the Hilbert function of $\mathcal{F}$ is given by

$$H_{I(\mathcal{F})}(m) = \dim_{\mathbb{F}} \left(\mathbb{F}\left[\mathbf{x}\right]_{\leq m} / I(\mathcal{F})_{\leq m} \right) = \operatorname{rank}_{\mathbb{F}} I \left(\mathcal{F}, \binom{[n]}{\leq m} \right).$$

3 Complete Uniform Families, Applications and Extensions

3.1 Gröbner Bases and Standard Monomials for Complete Uniform Families

We start here with an explicit description of the (reduced) Gröbner bases for the ideals $I_{n,d} := I(\mathcal{F})$, where $\mathcal{F} = \binom{[n]}{d}$ for some integer $0 \leq d \leq n$. That is, we consider the vanishing ideal of the set of all 0,1-vectors in $\mathbb{F}^n$ whose Hamming weight is d.

Let t be an integer, $0 < t \leq n/2$. We set $\mathcal{H}_t$ as the set of those subsets $\{s_1 < s_2 < \cdots < s_t\}$ of $[n]$ for which t is the smallest index j with $s_j < 2j$.

We have $\mathcal{H}_1 = \{\{1\}\}, \mathcal{H}_2 = \{\{2, 3\}\}$, and $\mathcal{H}_3 = \{\{2, 4, 5\}, \{3, 4, 5\}\}$. It is clear that if $\{s_1 < \ldots < s_t\} \in \mathcal{H}_t$, then $s_t = 2t - 1$, and $s_{t-1} = 2t - 2$ if $t > 1$.

For a subset $J \subseteq [n]$ and an integer $0 \leq i \leq |J|$ we denote by $\sigma_{J,i}$ the i-th elementary symmetric polynomial of the variables x_j, $j \in J$:

$$\sigma_{J,i} := \sum_{T \subseteq J, |T| = i} x_T \ \in \mathbb{F}[x_1, \ldots, x_n] .$$

In particular, $\sigma_{J,0} = 1$. Now let $0 < t \leq n/2$, $0 \leq d \leq n$, and $H \in \mathcal{H}_t$. Set

$$H' = H \cup \{2t, 2t + 1, \ldots, n\} \subseteq [n] .$$

We write

$$f_{H,d} = f_{H,d}(x_1, \ldots, x_n) := \sum_{k=0}^{t} (-1)^{t-k} \binom{d-k}{t-k} \sigma_{H',k} .$$

As an example, with $U = \{2, 3, \ldots, n\}$ we have

$$f_{\{2,3\},d} = \sigma_{U,2} - (d - 1)\sigma_{U,1} + \binom{d}{2} .$$

Gröbner bases of $I_{n,d}$ have been described in [26]:

Theorem 1. *Let $0 \leq d \leq n/2$ be integers, $\mathbb{F}$ a field, and $\prec$ be an arbitrary term order on the monomials of $\mathbb{F}[\mathbf{x}]$ for which $x_n \prec x_{n-1} \prec \ldots \prec x_1$. Then the following set G of polynomials is a Gröbner basis of the ideal $I_{n,d}$:*

$$G = \{x_1^2 - x_1, \ldots, x_n^2 - x_n\} \cup \{x_J : \ J \in \binom{[n]}{d+1}\} \cup$$
$$\cup \{f_{H,d} : \ H \in \mathcal{H}_t \ \textit{for some}\ 0 < t \leq d\} .$$

A similar description is valid for $I_{n,n-d}$ in the place of $I_{n,d}$. The standard monomials for the complete uniform families have also been obtained. The next theorem is valid for an arbitrary term order $\prec$ such that $x_n \prec x_{n-1} \prec \ldots \prec x_1$. For the lex order it was proved in [6], and later it was extended to general term orders in [26].

Theorem 2. *Let $0 \le d \le n/2$ and denote by $\mathcal{M} = \mathcal{M}_d$ the set of all monomials x_G such that $G = \{s_1 < s_2 < \ldots < s_j\} \subset [n]$ for which $j \le d$ and $s_i \ge 2i$ holds for every i, $1 \le i \le j$. Then $\mathcal{M}$ is the set of standard monomials for $I_{n,d}$ as well as for $I_{n,n-d}$ with respect to any term order $\prec$ as above.*

In particular, $|\mathcal{M}| = \binom{n}{d}$ and $\mathcal{M}$ is an $\mathbb{F}$ basis of the space of functions from $V_{\mathcal{F}}$ to $\mathbb{F}$. Also, Theorem 1 allows one to determine the reduced Gröbner bases of the ideals $I_{n,d}$. Here we note only the fact that a suitable *subset* of G turns out to be the reduced Gröbner basis of $I_{n,d}$ for $0 \le d \le \frac{n}{2}$.

3.2 Some Combinatorial Applications to q-Uniform Families

Let p be a prime, k an integer, and $q = p^{\alpha}$, $\alpha \ge 1$. Put

$$\mathcal{F}(k,q) = \{K \subseteq [n] : \ |K| \equiv k \pmod{q}\}\,.$$

In [27] the following rank inequality is proved for the inclusion matrices of $\mathcal{F}(k,q)$:

Theorem 3. *Let p be a prime and k an integer. Let $q = p^{\alpha} > 1$. If $\ell \le q-1$ and $2\ell \le n$, then*

$$rank_{\mathbb{F}_p}\, I\left(\mathcal{F}(k,q), \binom{[n]}{\le \ell}\right) \le \binom{n}{\ell}\,.$$

This result is a generalization of a theorem of Frankl [19] covering the case $\alpha = 1$. Theorem 3 is a direct consequence of the next inclusion relation involving deglex standard monomials. In simple words, it states that the low degree standard monomials of $\mathcal{F}(k,q)$ are contained among the standard monomials of the *complete uniform* families.

Theorem 4. *Let p be a prime and $q = p^{\alpha} > 1$. Let $\prec$ be the deglex order on the monomials of $\mathbb{F}[\mathbf{x}]$ with $\mathbb{F} = \mathbb{F}_p$. Suppose further that $k, \ell \in \mathbb{N}$, for which $0 \le k, \ell < q$, and $2\ell \le n$. Then*

$$\mathrm{Sm}\,(I(\mathcal{F}(k,q))) \cap \mathbb{F}\,[\mathbf{x}]_{\le \ell} \subseteq \mathcal{M}_{\ell}\,,$$

hence

$$|\,\mathrm{Sm}\,(I(\mathcal{F}(k,q))) \cap \mathbb{F}\,[\mathbf{x}]_{\le \ell}\,| \le \binom{n}{\ell}\,.$$

The crucial point of the proof is the fact that we know quite explicitly a Gröbner basis for the complete uniform families. To be a bit more specific here, suppose that k' is an integer such that $0 \le k' \le n$ and $k \equiv k' \pmod{q}$. Using simple properties of binomial coefficients one can infer that $f_{H,k} \equiv f_{H,k'} \pmod p$, i.e., the coefficients of the two polynomials are the same modulo p. This fact, together with a Gröbner reduction argument leads to a proof of Theorem 4.

Babai and Frankl conjectured the following in [7], p. 115.

Theorem 5. *Let k be an integer and $q = p^\alpha$, $\alpha \geq 1$ a prime power. Suppose that $2(q-1) \leq n$. Assume that $\mathcal{F} = \{A_1, \ldots, A_m\}$ is a family of subsets of $[n]$ such that*

$$(a)\ |A_i| \equiv k \ (mod\ q)\ for\ i = 1, \ldots, m$$

$$(b)\ |A_i \cap A_j| \not\equiv k \ (mod\ q)\ for\ 1 \leq i, j \leq m,\ \ i \neq j\,.$$

Then

$$m \leq \binom{n}{q-1}.$$

We briefly sketch a proof from [27]: let $\mathbf{v}_i \in \mathbb{Z}^n$ denote the characteristic vector of A_i, and write

$$f_i(x_1, \ldots, x_n) = \binom{\mathbf{x} \cdot \mathbf{v}_i - k - 1}{q-1}.$$

This is a polynomial in n rational variables $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Q}^n$. By conditions (a) and (b) the integer $f_i(\mathbf{v}_j)$ is divisible by p iff $i \neq j$.

Let f_i' be the square-free reduction of f_i for $i = 1, \ldots, m$. Then $f_i' \in \mathbb{Z}[\mathbf{x}]$, because $f_i(\mathbf{v}) \in \mathbb{Z}$ for each $\mathbf{v} \in \{0,1\}^n$. Let $g_i \in \mathbb{F}_p[\mathbf{x}]$ be the reduction of f_i' modulo p, and $h_i \in \mathbb{F}_p[\mathbf{x}]$ be the reduction of g_i by a deglex Gröbner basis for the ideal $I(\mathcal{F}(k,q))$ over $\mathbb{F}_p$.

For $1 \leq i, j \leq m$ we have then

$$f_i(\mathbf{v}_j) = f_i'(\mathbf{v}_j) \equiv g_i(\mathbf{v}_j) \equiv h_i(\mathbf{v}_j) \pmod p\,.$$

These imply, that the polynomials h_i are linearly independent mod p. They have degree at most $q-1$ and they are and spanned by $\mathrm{Sm}\,(I(\mathcal{F}(k,q)))$. By Theorem 4 their number is at most $\binom{n}{q-1}$.

3.3 Wilson's Rank Formula

Consider the inclusion matrix $A = I\left(\binom{[n]}{d}, \binom{[n]}{m}\right)$, where $m \leq d \leq n-m$.

A famous theorem of Richard M. Wilson [46, Theorem 2] describes a diagonal form of A over $\mathbb{Z}$. As a corollary, he obtained the following rank formula:

Theorem 6. *Let p be a prime. Then*

$$rank_{\mathbb{F}_p}(A) = \sum_{\substack{0 \leq i \leq m \\ p \nmid \binom{d-i}{m-i}}} \binom{n}{i} - \binom{n}{i-1}.$$

In [22] a simple proof is given which uses polynomial functions, and some basic notions related to Gröbner bases. The starting point is the observation that the rank of A is exactly the dimension of the linear space $\mathcal{P}_{d,m}$ over $\mathbb{F}_p$ of the functions from $V_{\binom{[n]}{d}}$ to $\mathbb{F}_p$ which are spanned by the monomials x_M with $|M| = m$.

The approach allows a considerable generalization of the rank formula. Here is a result of this kind from [22]:

Theorem 7. *Suppose that* $0 \le m_1 < m_2 \cdots < m_r \le d \le n - m_r$ *and let* p *be a prime. Consider the set family* $\mathcal{F} = \binom{[n]}{m_1} \cup \binom{[n]}{m_2} \cup \cdots \cup \binom{[n]}{m_r}$. *Then*

$$rank_{\mathbb{F}_p}\left(I\left(\binom{[n]}{d}, \mathcal{F}\right)\right) = \sum_{\substack{0 \le i \le m_r \\ p \nmid n_i}} \binom{n}{i} - \binom{n}{i-1},$$

where $n_i = \gcd\left(\binom{d-i}{m_1-i}, \binom{d-i}{m_2-i}, \ldots, \binom{d-i}{m_r-i}\right)$.

3.4 Generalizations of Uniform Families

Let n, k, ℓ be integers with $0 \le \ell - 1 \le k \le n$. The complete ℓ-*wide* family is

$$\mathcal{F}^{k,\ell} = \{F \subseteq [n] : k - \ell < |F| \le k\}.$$

Theorem 1 was extended in [21] to complete ℓ-wide families. Gröbner bases and standard monomials are described there over an arbitrary ground field $\mathbb{F}$. As in the case $\ell = 1$, the bases are largely independent of the term order considered.

A part of these results has been extended even further in [16]. Let q be a power of a prime p, and let n, d, ℓ be integers such that $1 \le n$, $1 \le \ell < q$. Consider the modulo q complete ℓ-wide family:

$$\mathcal{G} = \{F \subseteq [n] : \exists f \in \mathbb{Z} \text{ s. t. } d \le f < d + \ell \text{ and } |F| \equiv f \pmod{q}\}.$$

In [16] a Gröbner basis of the vanishing ideal $I(\mathcal{G})$ has been computed over fields of characteristic p. As before, it turns out that this set of polynomials is a Gröbner basis for all term orderings $\prec$, for which the order of the variables is $x_n \prec x_{n-1} \prec \cdots \prec x_1$. The standard monomials and the Hilbert function of $I(\mathcal{G})$ were also obtained. In this work the lex game method (see Sect. 4) was substantially used. As corollaries, several combinatorial applications follow. One of them is described next. It is a generalization of Theorem 5.

Let L be a subset of integers and $\mathcal{F}$ be a system of sets. Then $\mathcal{F}$ is *modulo* q L-*avoiding* if $G \in \mathcal{F}$ and $f \in L$ implies $|G| \not\equiv f \pmod{q}$. We call $\mathcal{F}$ L-*intersecting* if for any two distinct sets $G_1, G_2 \in \mathcal{F}$ the congruence $|G_1 \cap G_2| \equiv f \pmod{q}$ holds for some $f \in L$. A set $L \subseteq \{0, \ldots, q-1\}$ is called a *modulo* q *interval* if it is either an interval of integers or a union of two intervals L_1 and L_2, such that $0 \in L_1$ and $q - 1 \in L_2$.

Theorem 8. *Let q be a power of a prime, L be a modulo q interval and $\mathcal{F} \subseteq 2^{[n]}$ be a modulo q L-avoiding, L-intersecting family of sets. If $|L| \leq n - q + 2$, then*

$$|\mathcal{F}| \leq \sum_{k=|L|}^{q-1} \binom{n}{k} .$$

More generally, one may consider arbitrary fully symmetric set families. Let $D \subseteq [n]$ be arbitrary, and put

$$\mathcal{F}_D := \{Z \subseteq [n] \,:\, |Z| \in D\} .$$

Thus, $\mathcal{F}_D$ consists of all subsets of $[n]$ whose size is in D. It would be quite interesting to describe Gröbner bases and related structures for general set families of the form $\mathcal{F}_D$. Only some preliminary results are available, the most important of them being a beautiful theorem of Bernasconi and Egidi from [9]. It provides the deglex Hilbert function $h_{I(\mathcal{F}_D)}(m)$ of $I(\mathcal{F}_D)$ over $\mathbb{Q}$.

Theorem 9. *Let $0 \leq m \leq n$, and suppose that*

$$D = \{l_1, \ldots, l_s\} \cup \{m_1, \ldots, m_t\} ,$$

where $l_j \leq m$ and $m < m_1 < m_2 < \cdots < m_t$. Assume also, that

$$\{0, 1, \ldots, m\} \setminus \{l_1, \ldots, l_s\} = \{n_1, n_2, \ldots, n_{m+1-s}\} ,$$

with $n_1 > n_2 > \cdots > n_{m+1-s}$ and $u = \min\{t, m+1-s\}$. Then we have

$$h_{I(\mathcal{F}_D)}(m) = \sum_{j=1}^{s} \binom{n}{l_j} + \sum_{j=1}^{u} \min\{\binom{n}{m_j}, \binom{n}{n_j}\} .$$

A combinatorial description of the deglex standard monomials for $I(\mathcal{F}_D)$ over $\mathbb{Q}$ was obtained in [42] in the case when D has the following property: for each integer i, at most one of i and $n-i$ is in D. This characterization uses generalized ballot sequences. It would be of interest to extend this to more general sets D. Multivalued generalizations of uniform families are considered in [29].

4 The Lex Game and Applications

Based on [17], we outline a combinatorial approach to the lexicographic standard monomials of the vanishing ideal of a finite sets of points $V \subseteq \mathbb{F}^n$. This technique can be applied to compute the lex standard monomials of sets of combinatorial interest. The idea has been extended to general zero dimensional ideals in [18].

In this section, we use the lex term order. As before, let $\mathbb{F}$ be a field, $V \subseteq \mathbb{F}^n$ a finite set, and $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{N}^n$ an n dimensional vector of natural numbers. With these data as parameters, we define the Lex Game $\mathrm{Lex}(V; \mathbf{w})$, which is played by two persons, Lea and Stan.

Both Lea and Stan know V and $\mathbf{w}$. Their moves are:

1 Lea chooses w_n elements of $\mathbb{F}$.
  Stan picks a value $y_n \in \mathbb{F}$, different from Lea's choices.
2 Lea now chooses w_{n-1} elements of $\mathbb{F}$.
  Stan picks a $y_{n-1} \in \mathbb{F}$, different from Lea's (last w_{n-1}) choices.
... (The game proceeds in this way until the first coordinate.)
n Lea chooses w_1 elements of $\mathbb{F}$.
  Stan finally picks a $y_1 \in \mathbb{F}$, different from Lea's (last w_1) choices.

The winner is Stan, if during the game he could select a vector $\mathbf{y} = (y_1, \ldots, y_n)$ such that $\mathbf{y} \in V$, otherwise Lea wins the game. (If in any step there is no proper choice y_i for Stan, then Lea wins also.)

Example. Let $n = 5$, and $\alpha, \beta \in \mathbb{F}$ be different elements. Let V be the set of all α-β sequences in $\mathbb{F}^5$ in which the number of the α coordinates is 2 or 3. We claim that Lea can win with the question vector $\mathbf{w} = (11100)$, but for $\mathbf{w} = (00110)$ Stan has a winning strategy.

First consider $\mathbf{w} = (11100)$. To have $\mathbf{y} \in V$, Stan is forced to select values from $\{\alpha, \beta\}$. If Stan gives only β for the last 2 coordinates, then Lea will choose α in the first three, therefore $\mathbf{y}$ cannot contain any α coordinates. However, if Stan gives at least one α for the last 2 coordinates, then Lea, by keeping on choosing β, can prevent $\mathbf{y}$ from having at least two β coordinates.

For $\mathbf{w} = (00110)$ Stan's winning strategy is to pick $y_5 = \beta$, and choose from $\{\alpha, \beta\}$ (for the 4th and 3rd coordinates). If he selected so far α twice, then he can win by setting the first two coordinates to β. Otherwise he wins with the moves $y_1 = y_2 = \alpha$.

The game allows a nice characterization of the lexicographic leading monomials and standard monomials for V:

Theorem 10. *Let $V \subseteq \mathbb{F}^n$ be a finite set and $\mathbf{w} \in \mathbb{N}^n$. Stan wins* $\mathrm{Lex}(V; \mathbf{w})$ *if and only if $\mathbf{x}^{\mathbf{w}} \in \mathrm{Sm}\,(I(V))$. Equivalently, Lea wins the game if and only if $\mathbf{x}^{\mathbf{w}}$ is a leading monomial for $I(V)$.*

The theorem leads to a fast combinatorial algorithm to list those vectors $\mathbf{w} \in \mathbb{N}^n$ for which $\mathbf{x}^{\mathbf{w}} \in \mathrm{Sm}\,(I(V))$. The method uses constant times $|V|\,nk$ comparisons of field elements in the worst case, where k is the maximum number of different elements which appear in a fixed coordinate of points of V; see [17]. In particular, if $V \subseteq \{0, 1\}^n$ then $k \leq 2$ and hence we have a linear time algorithm.

The problem of computing lexicographic standard monomials for finite sets has had a long history starting with the seminal paper by Buchberger and Möller [40]. Their algorithm, as well as the subsequent methods of Marinari, Möller and Mora [36] and Abbott, Bigatti, Kreuzer and Robbiano [1] give also a Gröbner basis of $I(V)$. For the arithmetic complexity of these methods we have the bound $O(n^2m^3)$ when V is a subset of $\mathbb{F}^n$ and $|V| = m$ (see Sect. 3 in [15] for a related discussion). The Lex Game provides only the standard monomials, but in return it appears to lead to a much faster algorithm (see [17] for the details). In general we have the bound $O(nm^2)$. In some important special cases, such as the case of small finite ground fields which appear naturally in coding applications, we have a linear bound $O(nm)$ on the time cost of the algorithm.

5 Partitions and Colorings

5.1 Permutations, Trees and Partitions

Let $\alpha_1, \ldots, \alpha_n$ be n different elements of $\mathbb{F}$ and put

$$\Pi_n := \Pi_n(\alpha_1, \ldots, \alpha_n) := \{(\alpha_{\pi(1)}, \ldots, \alpha_{\pi(n)}) : \ \pi \in S_n\} .$$

Π_n is the set of all permutations of the α_i, considered as a subset of $\mathbb{F}^n$.

We recall the definition of the complete symmetric polynomials. Let i be a nonnegative integer and write

$$h_i(x_1, \ldots, x_n) = \sum_{a_1+\cdots+a_n=i} x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} .$$

Thus, $h_i \in \mathbb{F}[x_1, \ldots, x_n]$ is the sum of all monomials of total degree i. For $0 \leq i \leq n$ we write σ_i for the i-th elementary symmetric polynomial:

$$\sigma_i(x_1, \ldots, x_n) = \sum_{S \subseteq [n], \ |S|=i} x_S .$$

For $1 \leq k \leq n$ we introduce the polynomials $f_k \in \mathbb{F}[\mathbf{x}]$ as follows:

$$f_k = \sum_{i=0}^{k} (-1)^i h_{k-i}(x_k, x_{k+1}, \ldots, x_n) \sigma_i(\alpha_1, \ldots, \alpha_n) .$$

We remark, that $f_k \in \mathbb{F}[x_k, x_{k+1}, \ldots, x_n]$. Moreover, $\deg f_k = k$ and the leading monomial of f_k is x_k^k with respect to any term order $\prec$ for which $x_1 \succ x_2 \succ \ldots \succ x_n$. In [25] the following was proved:

Theorem 11. *Let $\mathbb{F}$ be a field and let $\prec$ be an arbitrary term order on the monomials of $\mathbb{F}[x_1, \ldots, x_n]$ such that $x_n \prec \ldots \prec x_1$. Then the reduced Gröbner basis of $I(\Pi_n)$ is $\{f_1, f_2, \ldots, f_n\}$. Moreover, the set of standard monomials is*

$$\{x_1^{\alpha_1} \ldots x_n^{\alpha_n} : \ 0 \leq \alpha_i \leq i-1, \text{ for } 1 \leq i \leq n\} .$$

We remark, that [25] gives also the reduced Gröbner basis of the set Y_m of characteristic vectors of oriented trees with vertex set $[m]$. Here we have $Y_m \subseteq \mathbb{F}^n$ with $n = m(m-1)$ and the coordinates are indexed by the edges of the complete digraph KD_m. The term order $\prec$ involved is a lexicographic order. It would be interesting to understand a Gröbner basis of Y_m with respect to a deglex (or other degree compatible) order.

Recall, that a sequence $\lambda = (\lambda_1, \ldots, \lambda_k)$ of positive integers is a *partition* of n, if $\lambda_1 + \lambda_2 + \cdots + \lambda_k = n$ and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k > 0$. Let $\mathbb{F}$ be a field, and $\alpha_0, \ldots, \alpha_{k-1}$ be k distinct elements of $\mathbb{F}$. Let $\lambda = (\lambda_1, \ldots, \lambda_k)$ be a partition of n and V_λ be the set of all vectors $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}^n$ such that

$$|\{j \in [n] : \ v_j = \alpha_i\}| = \lambda_{i+1}$$

for $0 \leq i \leq k-1$.

In their study of the q-Kostka polynomials, Garsia and Procesi have described the deglex standard monomials of $I(V_\lambda)$ (Proposition 3.2 in [23]). They worked over $\mathbb{Q}$, but their argument is valid over an arbitrary field. The associated graded ring $\mathrm{gr}\mathbb{F}[\mathbf{x}]/I(V_\lambda)$ is also described there.

In [28] it is shown that the lexicographic standard monomials of $I(V_\lambda)$ are the same as the deglex standard monomials over an arbitrary $\mathbb{F}$. In the proof a new description of the orthogonal complement $(S^\lambda)^\perp$ (with respect to the James scalar product) of the Specht module S^λ is given. As applications, a basis of $(S^\lambda)^\perp$ is exhibited, and a combinatorial description of the Hilbert function of V_λ is provided. This approach provides a new, simpler proof of the Garsia-Procesi theorem on the deglex standard monomials. An interesting feature of the results is that both in the lex and deglex cases the standard monomials are independent of the specific choice of $\alpha_0, \dots, \alpha_{k-1}$, or the field $\mathbb{F}$ itself.

These results partially extend the special cases we treated here earlier: the complete uniform set families, i.e., $\lambda = (n-d, d)$, see Theorem 2, and the permutations (the case $\lambda = (1^n)$), see Theorem 11. For general λ it seems to be difficult to give explicit Gröbner bases of $I(V_\lambda)$.

5.2 Graph Colorings

The algebraic study of graph colorings also employs fruitfully some Gröbner basis techniques. Here we briefly discuss some of these. Let G be a simple undirected graph on the vertex set $V = [n]$ and with edge set E. Let k be a fixed positive integer, and $\mathbb{F}$ be a field which contains k distinct k-th roots of unity. The set of those k-th roots of unity will be denoted by C_k. The *graph polynomial* $f_G \in \mathbb{F}[\mathbf{x}]$ is the polynomial

$$f_G := \prod_{(i,j)\in E,\ i<j} (x_i - x_j)\,.$$

Recall that a *k-coloring* of G is a map μ from $V(G)$ to C_k such that $\mu(i) \neq \mu(j)$, whenever $(i,j) \in E$. Moreover, a k coloring can be viewed as an element of $C_k^n \subseteq \mathbb{F}^n$. Let $\mathcal{K}$ be the set of graphs whose vertex set is $[n]$, which consist of a $k+1$-clique and $n-k-1$ isolated vertices. We introduce some important ideals from $\mathbb{F}[\mathbf{x}]$:

$$J_{n,k} := \langle f_H :\ H \in \mathcal{K}\rangle$$

is the ideal generated by the graph polynomials of $k+1$-cliques on $[n]$. Put

$$I_{n,k} := \langle x_i^k - 1 :\ i \in V\rangle\,.$$

It is easy to show that $I_{n,k}$ is actually $I(C_k^n)$, in fact, $\{x_1^k - 1, \dots x_n^k - 1\}$ is a universal Gröbner basis of $I_{n,k}$. Finally set

$$I_{G,k} := I_{n,k} + \langle x_i^{k-1} + x_i^{k-2}x_j + \cdots + x_j^{k-1} :\ (i,j) \in E\rangle\,.$$

It is a simple matter to verify, that $I_{G,k}$ is the ideal of the k-colorings of G: $\mu \in \mathbb{F}^n$ is a common zero for all polynomials from $I_{G,k}$ iff μ is a valid k-coloring of G.

The fact that G is *not* k colorable admits an algebraic characterization:

Theorem 12. *The next statements are equivalent:*
(1) G is not k-colorable.
(2) The constant polynomial 1 belongs to $I_{G,k}$.
(3) The graph polynomial f_G belongs to $I_{n,k}$.
(4) The graph polynomial f_G belongs to $J_{n,k}$.

The equivalence of (1) and (2) is due to Bayer [8], (1) $\Leftrightarrow$ (3) is from Alon and Tarsi [5], this was reproved by Gröbner basis techniques by de Loera [34] and Mnuk [39]. The equivalence (1) $\Leftrightarrow$ (4) is due to Kleitman and Lovász [35]. The following beautiful theorem is due to de Loera [34]:

Theorem 13. *The set of polynomials $\{f_H : H \in \mathcal{K}\}$ is a universal Gröbner basis of the ideal $J_{n,k}$.*

Let μ be a k-coloring of G, $\ell \leq k$ be the number of colors actually used by μ. The class $cl(i)$ is the set of vertices with the same color as i. Let

$$m_1 < m_2 < \cdots < m_\ell = n$$

be the maximal elements (coordinates) of the color classes.

For a set U of indices let h_U^d denote the complete symmetric polynomial of degree d in the variables whose indices are in U.

We define the polynomials g_i as follows:

$$g_i = \begin{cases} x_i^k - 1 & \text{if } i = m_\ell, \\ h_{\{m_j,\ldots,m_\ell\}}^{k-\ell+j} & \text{if } i = m_j \text{ for some } j \neq \ell, \\ x_i - x_{\max cl(i)} & \text{otherwise.} \end{cases}$$

Let $A_\mu := \langle g_1, g_2, \ldots, g_n \rangle$ be the ideal generated by the polynomials g_i. Hillar and Windfeldt [31] obtained the following:

Theorem 14. *We have*

$$I_{G,k} = \cap_\mu A_\mu \, ,$$

where μ runs over the k-colorings of G.

In the course of the proof they established, that for term orders $\prec$ with $x_1 \succ x_2 \succ \cdots \succ x_n$ the set $\{g_1, g_2, \ldots, g_n\}$ is actually a Gröbner basis of A_μ. By using these facts, they developed an algebraic characterization of unique k-colorability:

Theorem 15. *Let μ be a k-coloring of G that uses all k colors, $g_1, g_2, \ldots, g_n$ be the corresponding basis of A_μ. The following are equivalent.*
(1) G is uniquely k-colorable.
(2) The polynomials $g_1, g_2, \ldots, g_n$ generate $I_{G,k}$.
(3) The polynomials $g_1, g_2, \ldots, g_n$ are in $I_{G,k}$.
(4) The graph polynomial f_G is in the ideal $I_{n,k} : \langle g_1, g_2, \ldots, g_n \rangle$.
(5) $dim_{\mathbb{F}} \mathbb{F}[\mathbf{x}] / I_{G,k} = k!$

Condition (5) leads easily to an algebraic algorithm for testing the unique k-colorability of G. The left hand side is the number of the standard monomials for $I_{G,k}$ with respect to an arbitrary term order, hence (5) can be checked by standard techniques for computing Gröbner bases. See Sect. 6 in [31] for more details and data on computational experiments.

6 Alon's Combinatorial Nullstellensatz

Alon's Combinatorial Nullstellensatz, and in particular the resulting non-vanishing criterion from [4] is one of the most powerful algebraic tools in combinatorics, with dozens of important applications.

Let $\mathbb{F}$ be a field and $S_1, \dots, S_n$ be nonempty, finite subsets of $\mathbb{F}$, $|S_i| = t_i$. Put $S = S_1 \times \cdots \times S_n$ and define $g_i(x_i) = \prod_{s \in S_i}(x_i - s)$.

Theorem 16. *(Theorem 1.1 from [4].) Let $f = f(\mathbf{x})$ be a polynomial from $\mathbb{F}[\mathbf{x}]$ that vanishes over all the common zeros of $g_1, \dots, g_n$ (that is, if $f(\mathbf{s}) = 0$ for all $s \in S$). Then there exist polynomials $h_1, \dots, h_n \in \mathbb{F}[\mathbf{x}]$ satisfying $deg(h_i) \leq deg(f) - deg(g_i)$ so that*

$$f = \sum_{i=1}^{n} h_i g_i \,.$$

Moreover, if $f, g_1, \dots, g_n$ lie in $R[\mathbf{x}]$ for some subring R of $\mathbb{F}$, then there are polynomials $h_i \in R[\mathbf{x}]$ as above.

The Combinatorial Nullstellensatz can be reformulated in terms of Gröbner bases. It states that $\{g_1, g_2, \dots, g_n\}$ is a universal Gröbner basis of the ideal $I(S)$. The most important corollary of Theorem 16 is a non-vanishing criterion:

Theorem 17. *(Theorem 1.2 from [4].) Let $f = f(\mathbf{x})$ be a polynomial in $\mathbb{F}[\mathbf{x}]$. Suppose the degree of f is $\sum_{i=1}^{n} d_i$, where $d_i < t_i$ for all i and the coefficient of $\prod_{i=1}^{n} x_i^{d_i}$ in f is nonzero. Then there is a point $\mathbf{s} \in S$ such that $f(\mathbf{s}) \neq 0$.*

The theorem has numerous applications in combinatorial number theory, graph theory and combinatorics. To demonstrate the amazing versatility of Theorem 17, we give here an argument form [4] to prove the Cauchy and Davenport inequality from additive number theory. The inequality states that if p is a prime and A, B are two nonempty subsets of $\mathbb{Z}_p$, then

$$|A + B| \geq \min\{p, |A| + |B| - 1\} \,.$$

We remark first, that the case $|A| + |B| > p$ is easy. We may then assume that $|A| + |B| \leq p$. Assume for contradiction, that there is a subset $C \subset \mathbb{F}_p$ such that $C \supseteq A + B$, and $|C| = |A| + |B| - 2$. Put

$$f(x, y) = \prod_{c \in C}(x + y - c) \in \mathbb{F}_p[x, y] \,.$$

Clearly f is identically zero on $A \times B$. Now set $n = 2$, $S_1 = A$, $S_2 = B$, $t_1 = |A|$ and $t_2 = |B|$. We have $\deg f = t_1 - 1 + t_2 - 1$; the coefficient of $x^{t_1-1}y^{t_2-1}$ is $\binom{t_1-1+t_2-1}{t_1-1}$, which is not 0 in $\mathbb{F}_p$, as $t_1 - 1 + t_2 - 1 < p$. By Theorem 17 f can not be identically zero on $A \times B$. This is a contradiction proving the Cauchy-Davenport inequality.

There are natural ways to generalize Theorems 16 and 17. One is to prove a variant of the Nullstellensatz over rings instead of fields, an other is to consider the non-vanishing problem for multisets and not merely sets. Extensions along these lines are considered in [32], [33], and [38].

7 Gröbner Bases and S-Extremal Set Systems

Gröbner basis methods may be useful when studying extremal problems of combinatorics (see Frankl [20] for a survey of extremal questions on set families). We give here a new application of this kind.

We say that a set system $\mathcal{F} \subseteq 2^{[n]}$ *shatters* a given set $S \subseteq [n]$ if

$$2^S = \{F \cap S : \ F \in \mathcal{F}\}.$$

The family of subsets of $[n]$ shattered by $\mathcal{F}$ is denoted by $\mathrm{Sh}(\mathcal{F})$. The notion of shattering occurs in various fields of mathematics, such as combinatorics, statistics, computer science, and logic. As an example, one can mention the Vapnik-Chervonenkis dimension of a set system $\mathcal{F}$, i.e. the size of the largest S shattered by $\mathcal{F}$. Sauer [43], Shelah [44] and Vapnik and Chervonenkis [45] proved that if $\mathcal{F}$ is a family of subsets of $[n]$ with no shattered set of size k (i.e. $\mathrm{VC}-\dim\mathcal{F} < k$), then

$$|\mathcal{F}| \leq \binom{n}{k-1} + \binom{n}{k-2} + \cdots + \binom{n}{0},$$

and this inequality is best possible. The result is known as Sauer's lemma and has found applications in a variety of contexts, including applied probability.

It was proved by several authors (Aharoni and Holzman [3], Pajor [41], Sauer [43], Shelah [44]) that for every set system $\mathcal{F} \subseteq 2^{[n]}$ we have that $|\mathrm{Sh}(\mathcal{F})| \geq |\mathcal{F}|$. Accordingly, we define a set system $\mathcal{F}$ to be *S-extremal*, if $|\mathrm{Sh}(\mathcal{F})| = |\mathcal{F}|$. We refer to Bollobás and Radcliffe [10] for some basic properties of S-extremal set families, where they mention the lack of a good structural description of these families.

It turns out, that shattered sets are in close connection with the standard monomials of the ideal $I(\mathcal{F})$ for different term orders. To make this connection explicit, we first have to define a special family of term orders. At the beginning we have already defined the lex term order. By reordering the variables one can define another lex term order, so from now on we will talk about lex term orders based on some permutation of the variables $x_1, x_2, \ldots, x_n$. There are $n!$ possible lexicographic orders on n variables.

For a pair of sets $G \subseteq H \subseteq [n]$ we define the polynomial $f_{H,G}$ as

$$f_{H,G} = (\prod_{j \in G} x_j)(\prod_{i \in H \setminus G} (x_i - 1)) .$$

Lemma 1. *a) If $x_H \in \mathrm{Sm}\,(I(\mathcal{F}))$ for some term order, then $H \in \mathrm{Sh}(\mathcal{F})$.*
b) If $H \in \mathrm{Sh}(\mathcal{F})$, then there is a lex term order for which $x_H \in \mathrm{Sm}\,(I(\mathcal{F}))$.

Proof. a) Let $x_H \in \mathrm{Sm}\,(I(\mathcal{F}))$, and suppose that H is not shattered by $\mathcal{F}$. This means that there exists a $G \subseteq H$ for which there is no $F \in \mathcal{F}$ such that $G = H \cap F$. Now $f_{H,G}(\mathbf{v}_F) \neq 0$ only if $H \cap F = G$. According to our assumption, there is no such set $F \in \mathcal{F}$, so $f_{H,G}(\mathbf{x}) \in I(\mathcal{F})$. This implies that $x_H \in \mathrm{Lm}(I(\mathcal{F}))$ for all term orders, since $\mathrm{lm}(f_{H,G}) = x_H$ for all term orders, giving a contradiction.

b) We prove that a lex order, where the variables of x_H are the smallest ones, satisfies the claim. Suppose the contrary, that $x_H \in \mathrm{Lm}(I(\mathcal{F}))$ for this term order. Then there is a polynomial $f(\mathbf{x})$ vanishing on $\mathcal{F}$ with leading monomial x_H. Since the variables in x_H are the smallest according to this term order, there cannot appear any other variable in $f(x)$. So we may assume that $f(\mathbf{x})$ has the form $\sum_{G \subseteq H} \alpha_G x_G$. Take a subset $G_0 \subseteq H$ which appears with a nonzero coefficient in $f(\mathbf{x})$, and is minimal w.r.t. this property. $\mathcal{F}$ shatters H, so there exists a set $F_0 \in \mathcal{F}$ such that $G_0 = F_0 \cap H$. For this we have $x_{G_0}(\mathbf{v}_{F_0}) = 1$, and since G_0 was minimal, $x_G(\mathbf{v}_{F_0}) = 0$ for every other set G in the sum. So $f(\mathbf{v}_{F_0}) = \alpha_{G_0} \neq 0$, which contradicts $f \in I(\mathcal{F})$). This contradiction proves the statement. □

Combining the two parts of Lemma 1, we obtain that $\mathrm{Sm}\,(I(\mathcal{F})) \subseteq \mathrm{Sh}(\mathcal{F})$ for every term order, and

$$\mathrm{Sh}(\mathcal{F}) = \bigcup_{\text{term orders}} \mathrm{Sm}\,(I(\mathcal{F})) . \tag{1}$$

(Here, by identifying a squarefree monomial x_H with the set of indices $H \subseteq [n]$, we view $\mathrm{Sm}\,(I(\mathcal{F}))$ as a set family over $[n]$.) Note that on the right hand side it is sufficient to take the union over the lex term orders only. Using the lex game method, one can efficiently compute $\mathrm{Sm}\,(I(\mathcal{F}))$ for any lex term order. However, as the number of lex orders is $n!$, (1) does not immediately provide an efficient way to calculate $\mathrm{Sh}(\mathcal{F})$. Nevertheless Lemma 1 implies at once a simple algebraic characterization of S-extremal set systems:

Theorem 18. *$\mathcal{F}$ is S-extremal if and only if $\mathrm{Sm}\,(I(\mathcal{F}))$ is the same for all lex term orders.*

Theorem 18 leads to an algebraic characterization of S-extremal set systems, involving the Gröbner bases of $I(\mathcal{F})$.

Theorem 19. *$\mathcal{F} \subseteq 2^{[n]}$ is S-extremal if and only if there are polynomials of the form $f_{S,H}$, which together with $\{x_i^2 - x_i :\ i \in [n]\}$ form a Gröbner basis of $I(\mathcal{F})$ for all term orders.*

Proof. Suppose first, that $\mathcal{F}$ is S-extremal. Consider all minimal sets $S \subseteq [n]$ for which $S \notin \mathrm{Sh}(\mathcal{F})$, with a corresponding polynomial $f_{S,H}$. Here $H \subseteq S$ is a set which is not of the form $S \cap F$ for any $F \in \mathcal{F}$. Denote the set of these sets S by $\mathcal{S}$ and fix an arbitrary term order. We prove that these polynomials together with $\{x_i^2 - x_i : \ i \in [n]\}$ form a Gröbner basis of $I(\mathcal{F})$. In order to show this, we have to prove that for all monomials $m \in \mathrm{Lm}(I(\mathcal{F}))$, there is a monomial in

$$\{x_S : \ S \in \mathcal{S}\} \cup \{x_i^2 : \ i \in [n]\}$$

that divides m. If m is not square-free, then this is trivial. Now suppose m is square-free, say $m = x_F$ for a subset $F \subseteq [n]$. $\mathcal{F}$ is extremal, thus we have $|\mathrm{Sh}(\mathcal{F})| = |\mathcal{F}| = |\,\mathrm{Sm}\,(I(\mathcal{F}))\,|$ and hence $\mathrm{Sm}\,(I(\mathcal{F})) = \mathrm{Sh}(\mathcal{F})$. We have then $F \notin \mathrm{Sh}(\mathcal{F})$, as m is a leading monomial. Then there is an $S \in \mathcal{S}$ with $S \subseteq F$. This proves that our basis is a Gröbner basis.

For the opposite direction, suppose that there is a common Gröbner basis G for all term orders of the desired form. G is a Gröbner basis of $I(\mathcal{F})$, so

$$\mathrm{Lm}(G) = \{x_S : \ S \in \mathcal{S}\} \cup \{x_i^2 : \ i \in [n]\}$$

determines $\mathrm{Lm}(I(\mathcal{F}))$ and hence $\mathrm{Sm}\,(I(\mathcal{F}))$. This clearly implies that $\mathrm{Sm}\,(I(\mathcal{F}))$ is the same for all term orders, since G is a common Gröbner basis for all term orders. □

We remark that in the theorem the phrase *all term orders* may be replaced by *a term order*. To see this, please note that the standard monomials of $\mathcal{F}$ are then precisely the monomials x_F where there is no polynomial $f_{S,H}$ in the basis with $S \subseteq F$. This is independent of the term order considered.

In addition to this characterization, Theorem 18 leads also to an efficient algorithm for testing the S-extremality. The test is based on the theorem below.

Theorem 20. *Take n orderings of the variables such that for every index i there is one in which x_i is the greatest element, and take the corresponding lex term orders. If $\mathcal{F}$ is not extremal, then among these we can find two term orders for which the standard monomials of $I(\mathcal{F})$ differ.*

Proof. Let us fix one of the above mentioned lex orders. Suppose that $\mathcal{F}$ is not S-extremal. Then there is a set $H \in \mathcal{F}$ shattered by $\mathcal{F}$ for which x_H is not a standard monomial but a leading one. $\mathrm{Sm}\,(I(\mathcal{F}))$ is a basis of the linear space $\mathbb{F}[\mathbf{x}]/I(\mathcal{F})$, and since all functions from $V_{\mathcal{F}}$ to $\mathbb{F}$ are polynomials, every leading monomial can be written uniquely as an $\mathbb{F}$-linear combination of standard monomials, as a function on $V_{\mathcal{F}}$. This holds for x_H as well. As functions on $V_{\mathcal{F}}$ we have

$$x_H = \sum \alpha_G x_G \, .$$

Suppose that for all sets G in the sum we have $G \subseteq H$. Take a minimal G_0 with a nonzero coefficient. Since H is shattered by $\mathcal{F}$, there is an $F \in \mathcal{F}$ such that $G_0 = F \cap H$. For this $x_{G_0}(\mathbf{v}_F) = 1$. From the minimality of G_0 we have that $x_{G'}(\mathbf{v}_F) = 0$ for every other $G' \subseteq H$, giving that

$$\sum \alpha_G x_G(\mathbf{v}_F) = \alpha_{G_0} .$$

On the other hand $x_H(\mathbf{v}_F) = 0$, since $H \cap F = G_0$, but $H \neq G$ because x_H is a leading monomial, and x_G is a standard monomial, giving a contradiction. Therefore in the above sum there is a set G with nonzero coefficient such that $G \backslash H \neq \emptyset$. Now let us fix an index $i \in G \backslash H$. For the term order where x_i is the greatest variable, x_H cannot be the leading monomial of the polynomial $x_H - \sum \alpha_G x_G$. Then the leading monomial is another $x_{G'}$, which, for the original term order was a standard monomial. So we have found two term orders for which the standard monomials differ. $\square$

In view of the preceding theorem, it is enough to calculate the standard monomials e.g. for a lexicographic term order and its cyclic permutations, and to check, whether they differ or not. The standard monomials can be calculated in time $O(n|\mathcal{F}|)$ for one lexicographic term order, see [17]. We have n term orders, therefore the total running time of the algorithm is $O(n^2|\mathcal{F}|)$.

Theorem 21. *Given a set family $\mathcal{F} \subseteq 2^{[n]}$, $|\mathcal{F}| = m$ by a list of characteristic vectors, we can decide in $O(n^2 m)$ time whether $\mathcal{F}$ is extremal or not.*

This improves the algorithm given in [24] by Greco, where the time bound is $O(nm^3)$. But it is still open whether it is possible to test S-extremality in linear time (i.e. in time $O(nm)$).

We note that the results discussed here can be generalized to a multivalued setting, see [37]. The starting point is Theorem 18. We define a set $V \subseteq \mathbb{F}^n$ to be *S-extremal*, if $\mathrm{Sm}\,(I(V))$ is independent of the term order, i.e. it stays the same for all term orders.

References

1. Abbott, J., Bigatti, A., Kreuzer, M., Robbiano, L.: Computing Ideals of Points. J. Symbolic Comput. 30, 341–356 (2000)
2. Adams, W.W., Loustaunau, P.: An Introduction to Gröbner bases. Graduate Studies in Mathematics, vol. 3. American Mathematical Society, Providence (1994)
3. Aharoni, R., Holzman, R.: Personal communication, cited in [24]
4. Alon, N.: Combinatorial Nullstellensatz. Combinatorics, Probability and Computing 8, 7–29 (1999)
5. Alon, N., Tarsi, M.: Colorings and Orientation of Graphs. Combinatorica 12, 125–134 (1992)
6. Anstee, R.P., Rónyai, L., Sali, A.: Shattering News. Graphs and Combinatorics 18, 59–73 (2002)
7. Babai, L., Frankl, P.: Linear Algebra Methods in Combinatorics. Prel. vers (1992)
8. Bayer, D.: The Division Algorithm and the Hilbert Scheme. PhD. Thesis. Harvard University (1982)
9. Bernasconi, A., Egidi, L.: Hilbert Function and Complexity Lower Bounds for Symmetric Boolean Functions. Information and Computation 153, 1–25 (1999)
10. Bollobás, B., Radcliffe, A.J.: Defect Sauer Results. Journal of Combinatorial Theory, Series A 72, 189–208 (1995)

11. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Doctoral thesis, University of Innsbruck (1965), English Translation: An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal. Journal of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions 41, 475–511 (2006)
12. Buchberger, B.: Ein algorithmisches Kriterium fur die Lösbarkeit eines algebraischen Gleichungssystems. Aequationes Mathematicae 4, 374–383 (1970); English translation: An Algorithmic Criterion for the Solvability of Algebraic Systems of Equations. In: Buchberger, B., Winkler, F. (eds.) Gröbner Bases and Applications. London Mathematical Society Lecture Note Series, vol. 251, pp. 535–545. Cambridge University Press, Cambridge (1998)
13. Buchberger, B.: Gröbner-Bases: An Algorithmic Method in Polynomial Ideal Theory. In: Bose, N.K. (ed.) Multidimensional Systems Theory - Progress, Directions and Open Problems in Multidimensional Systems Theory, pp. 184–232. Reidel Publishing Company, Dordrecht (1985)
14. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms. Springer, Heidelberg (1992)
15. Farr, J.B., Gao, S.: Computing Gröbner Bases for Vanishing Ideals of Finite Sets of Points. In: Fossorier, P.C.M., Imai, H., Lin, S., Poli, A. (eds.) AAECC 2006. LNCS, vol. 3857, pp. 118–127. Springer, Heidelberg (2006)
16. Felszeghy, B., Hegedűs, G., Rónyai, L.: Algebraic Properties of Modulo q complete ℓ-wide Families. Combinatorics, Probability and Computing 18, 309–333 (2009)
17. Felszeghy, B., Ráth, B., Rónyai, L.: The lex game and some applications. J. Symbolic Computation 41, 663–681 (2006)
18. Felszeghy, B., Rónyai, L.: On the lexicographic standard monomials of zero dimensional ideals. In: Proc. 10th Rhine Workshop on Computer Algebra (RWCA), pp. 95–105 (2006)
19. Frankl, P.: Intersection Theorems and mod p Rank of Inclusion Matrices. Journal of Combinatorial Theory, Series A 54, 85–94 (1990)
20. Frankl, P.: Extremal set systems. In: Graham, R.L., Grötschel, M., Lovász, L. (eds.) Handbook of Combinatorics, vol. 2, pp. 1293–1329. MIT Press, Cambridge (1996)
21. Friedl, K., Hegedűs, G., Rónyai, L.: Gröbner Bases for Complete ℓ-wide Families. Publ. Math. Debrecen. 70, 271–290 (2007)
22. Friedl, K., Rónyai, L.: Order Shattering and Wilson's Theorem. Discrete Mathematics 270, 127–136 (2003)
23. Garsia, A.M., Procesi, C.: On Certain Graded S_n-modules and the q-Kostka Polynomials. Advances in Mathematics 94, 82–138 (1992)
24. Greco, G.: Embeddings and Trace of Finite Sets. Information Processing Letters 67, 199–203 (1998)
25. Hegedűs, G., Nagy, A., Rónyai, L.: Gröbner Bases for Permutations and Oriented Trees. Annales Univ. Sci. Budapest, Sectio Computatorica 23, 137–148 (2004)
26. Hegedűs, G., Rónyai, L.: Gröbner Bases for Complete Uniform Families. Journal of Algebraic Combinatorics 17, 171–180 (2003)
27. Hegedűs, G., Rónyai, L.: Standard Monomials for q-uniform Families and a Conjecture of Babai and Frankl. Central European Journal of Mathematics 1, 198–207 (2003)
28. Hegedűs, G., Rónyai, L.: Standard Monomials for Partitions. Acta Mathematica Hungarica 111, 193–212 (2006)

29. Hegedűs, G., Rónyai, L.: Multivalued Generalizations of the Frankl–Pach Theorem. To appear, Journal of Algebra and its Applications, http://arxiv.org/pdf/1008.4660
30. Herzog, J., Hibi, T.: Monomial Ideals. GTM, vol. 260. Springer, Heidelberg (2010)
31. Hillar, C.J., Windfeldt, T.: Algebraic Characterization of Uniquely Vertex Colorable Graphs. Journal of Combinatorial Theory, Series B 98, 400–414 (2007)
32. Kós, G., Rónyai, L.: Alon's Nullstellensatz for multisets, http://arxiv.org/pdf/1008.2901
33. Kós, G., Mészáros, T., Rónyai, L.: Some Extensions of Alon's Nullstellensatz, http://arxiv.org/abs/1103.4768
34. de Loera, J.A.: Gröbner Bases and Graph Colorings. Beiträge zur Algebra und Geometrie 36, 89–96 (1995)
35. Lovász, L.: Stable sets and Polynomials. Discrete Mathematics 124, 137–153 (1994)
36. Marinari, M.G., Möller, H.M., Mora, T.: Gröbner Bases of Ideals Defined by Functionals with an Application to Ideals of Projective Points. Appl. Algebra Engrg. Comm. Comput. 4, 103–145 (1993)
37. Mészáros, T.: S-extremal Set Systems and Gröbner Bases. MSc Thesis, BME, Budapest (2010), http://www.math.bme.hu/~slovi/thesiswork.pdf
38. Michałek, M.: A Short Proof of Combinatorial Nullstellensatz. American Mathematical Monthly 117, 821–823 (2010)
39. Mnuk, M.: On an Algebraic Description of Colorability of Planar Graphs. In: Nakagawa, K. (ed.) Logic, Mathematics and Computer Science: Interactions. Proc. of the Symposium in Honor of Bruno Buchberger's 60th Birthday, pp. 177–186 (2002)
40. Möller, H.M., Buchberger, B.: The Construction of Multivariate Polynomials with Preassigned Zeros. In: Calmet, J. (ed.) ISSAC 1982 and EUROCAM 1982. LNCS, vol. 144, pp. 24–31. Springer, Heidelberg (1982)
41. Pajor, A.: Sous-espaces l_1^n des espaces de Banach, Travaux en Cours. Hermann, Paris (1985)
42. Pintér, D., Rónyai, L.: Standard Monomials of some Symmetric Sets. Acta Universitatis Apulensis. Math. Inform. (10), 331–344 (2005)
43. Sauer, N.: On the Density of Families of Sets. Journal of Combinatorial Theory, Series A 13, 145–147 (1972)
44. Shelah, S.: A Combinatorial Problem: Stability and Order for Models and Theories in Infinitary Language. Pacific Journal of Mathematics 41, 247–261 (1972)
45. Vapnik, V.N., Chervonenkis, A.Y.: On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. Theory of Probability and its Applications 16, 264–280 (1971)
46. Wilson, R.M.: A Diagonal Form for the Incidence Matrices of t-subsets vs. k-subsets. European Journal of Combinatorics 11, 609–615 (1990)

Comparing Necessary Conditions for Recognizability of Two-Dimensional Languages*

Marcella Anselmo[1] and Maria Madonia[2]

[1] Dipartimento di Informatica, Università di Salerno I-84084 Fisciano (SA) Italy
anselmo@dia.unisa.it

[2] Dip. Matematica e Informatica, Università di Catania, Viale Andrea Doria 6/a, 95125 Catania, Italy
madonia@dmi.unict.it

Abstract. The main problem in this paper is to find feasible conditions to prove/disprove that a given two-dimensional language is tiling recognizable. We focus on two known conditions necessarily satisfied by a recognizable language, that are based on the idea to reduce the problem from picture to string languages. We emphasize that they are grounded on two lower bound techniques for regular string languages. Starting from a stronger lower bound, named the nondeterministic message complexity technique, we are able to state a new necessary condition for the recognizability of two-dimensional languages. We compare the three conditions and find that the new one extends the previous two yielding a greater accuracy.

Keywords: Two-dimensional languages, Lower bound techniques on NFA.

1 Introduction

Picture languages are a generalization of string languages to two dimensions: a picture is a two-dimensional array of elements from a finite alphabet. The increasing interest for pattern recognition and image processing has motivated the research on two-dimensional languages, and especially tile-based models. The aim is to generalize or extend the well-founded theory of formal languages. Since the sixties, the problem has been approached from different points of view: finite automata, grammars, logics and regular expressions have been proposed. Among the various classes of languages defined, probably the most successful, as far as theoretical aspects are concerned, is the class of *tiling recognizable languages*, also known as REC (see [11,12]). A two-dimensional language is said (tiling) recognizable when it is the alphabetic projection of a local language defined in

* This work was partially supported by MIUR Project *"Aspetti matematici e applicazioni emergenti degli automi e dei linguaggi formali"* (2007), by ESF Project *"AutoMathA"* (2005-2010), by 60% Projects of University of Catania and Salerno (2008, 2009).

terms of a finite set of 2×2 pictures, the allowed tiles; the recognition is given by a so called *tiling system*.

Since its introduction, family REC has been intensively studied (see e.g. [1,4,8,12]). The definition in terms of tiling systems turns out to be robust in comparison with the other models in the literature: REC has a characterization in terms of logical monadic second-order formulas [12,14]; it has a counterpart as machine model in the two-dimensional on-line tessellation acceptor [20], whereas other models of automata are proposed in [2,7,21]. Tiling systems can be also simulated by domino systems [20], Wang tiles [9] and grammars [8].

Nevertheless the problem of establishing whether a language is tiling recognizable or not still deserves to be investigated. There do not exist, in the literature, feasible characterizations of REC based on the "nature" of the language itself, that could be handily used for this goal. In the string language theory, the problem of deciding whether a given language is regular, can be solved considering some congruence classes and the Myhill-Nerode Theorem; while a very useful tool to prove that a language is not regular is given by the Pumping Lemma ([17]). The problem seems much more complex in two dimensions. Recall that the runs of a Turing machine form a recognizable two-dimensional language (cf. [12]) so that one cannot expect simple periodicity results.

Recently, useful tools to prove the non-recognizability of a picture language, were given in [1,3,5,13]. The starting idea is to view the pictures with same number of rows m, as strings over the alphabet of the columns of height m. Hence a picture language can be considered as the sequence, when m grows, of such "string" languages. More in details, a first step in this direction was done in [22], where O. Matz isolated a technique for showing that a language is not recognizable. It consists in considering, for any recognizable picture language L and integer m, the string language $L(m)$ of all pictures in L of fixed height m. Then if $L \in$ REC it is possible to associate to any tiling system recognizing L a family $\{A_m\}$, where each A_m is an automaton accepting $L(m)$ with $2^{O(m)}$ states. Using some known lower bound on the size of an automaton, he proved a necessary condition for the belonging of a picture language to REC (as stated in Lemma 2).

In [3] a further step forward was done: Matz's technique was firstly used together with some bound on the Hankel matrices of the string languages $L(m)$. The combination of these two ideas (Matz's technique and Hankel matrices) has allowed to obtain necessary conditions for the belonging of a language to other meaningful sub-classes of REC [1]. Recently in [13], several necessary conditions for the belonging of a language to REC or to other related families, have been put in a uniform framework, by introducing some complexity functions (the "row", "permutation" and "rank" complexity functions) of a picture language expressing the growth of some parameters of the associated Hankel matrices. Note that other necessary conditions for non-recognizabilty have been proved in [5,10] for a specific class of unary picture languages definable by functions (see Section 4).

Also note that, in [22], the author wondered whether its necessary condition for REC (Lemma 2) was also sufficient, in order to have a handy characterization of REC. Very recently, Matz's condition was considered in [5] in its formulation by Hankel matrices, together with another recognizability condition. Unfortunately, both conditions were showed to be not sufficient.

Recognizability conditions are the starting point of the investigation in this paper. We will enucleate in the proofs of such conditions the use, often not explicit, of known lower bound techniques for regular string languages. In fact, the condition stating a bound on the permutation complexity function of a language in REC, follows from the "fooling set" technique; while the condition stating a bound on the fooling complexity function of a language in REC follows from the "extended fooling set" technique (see [5,16]).

Deepening the investigation of such lower bounds (see [16,19]), we will also consider the "nondeterministic message complexity" technique. It allows us to state another condition necessary for a picture language be in REC, as an upper bound to its "covering" complexity function, here introduced. This new condition implies and extends the previous two, as it follows from the comparison of the three functions. Moreover the comparison of the asymptotic growth of the functions shows that the cover complexity function provides a more accurate bound than the other two, also in the two-dimensional case.

The computation of the three functions is algorithmically possible, but, in general, not easy. So that it is important to consider all the complexity functions, since the computation, in some cases, could be easier for a function than another. At last we will find that even this condition is not sufficient for the recognizability of a picture language, but it could constitute a stronger condition to prove that a language is not tiling recognizable.

2 Preliminaries

In this section we recall some definitions about two-dimensional recognizable languages. More details can be mainly found in [12].

A *two-dimensional string* (or a *picture*) over a finite alphabet Σ is a two-dimensional rectangular array of elements of Σ. The set of all pictures over Σ is denoted by Σ^{**} and a *two-dimensional language* over Σ is a subset of Σ^{**}.

Given a picture $p \in \Sigma^{**}$, let $p_{(i,j)}$ denote the symbol in p with coordinates (i,j), $|p|_r$ the number of rows and $|p|_c$ the number of columns of p; the pair $(|p|_r, |p|_c)$ is the *size* of p. Note that when a one-letter alphabet is concerned, a picture p is totally defined by its size (m,n), and we will write $p = (m,n)$. Remark that the set Σ^{**} includes also all the empty pictures, i.e. all pictures of size $(m,0)$ or $(0,n)$ for all $m,n \geq 0$, that we denote by $\lambda_{m,0}$ and $\lambda_{0,n}$ respectively. The set of all pictures over Σ of size (m,n) is denoted by $\Sigma^{m,n}$, while $\Sigma^{m,*}$ ($\Sigma^{*,n}$, resp.) denotes the set of all pictures over Σ with m rows (with n columns, resp.). It will be needed to identify the symbols on the boundary of a given picture: for any picture p of size (m,n), we consider the *bordered picture* $\widehat{p}$ of size $(m+2, n+2)$ obtained by surrounding p with a special *boundary symbol* $\# \notin \Sigma$.

A *tile* is a picture of size $(2,2)$ and $B_{2,2}(p)$ is the set of all sub-blocks of size $(2,2)$ of a picture p. Given an alphabet Γ, a two-dimensional language $L \subseteq \Gamma^{**}$ is *local* if there exists a set Θ of tiles over $\Gamma \cup \{\#\}$ such that $L = \{p \in \Gamma^{**} | B_{2,2}(\widehat{p}) \subseteq \Theta\}$ and we will write $L = L(\Theta)$.

A *tiling system* is a quadruple $(\Sigma, \Gamma, \Theta, \pi)$ where Σ and Γ are finite alphabets, Θ is a finite set of tiles over $\Gamma \cup \{\#\}$ and $\pi : \Gamma \rightarrow \Sigma$ is a projection. A two-dimensional language $L \subseteq \Sigma^{**}$ is *tiling recognizable* if there exists a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ such that $L = \pi(L(\Theta))$ (extending π in the usual way). For any $p \in L$, a local picture $p' \in L(\Theta)$, such that $p = \pi(p')$, is called a *pre-image* of p. We denote by *REC* the family of all *tiling recognizable* picture languages.

The *column concatenation* of p and q (denoted by $p \obar q$) and the *row concatenation* of p and q (denoted by $p \ominus q$) are partial operations, defined only if $|p|_r = |q|_r$ and if $|p|_c = |q|_c$, respectively, and are given by:

$$p \obar q = \begin{array}{|c|c|} \hline \quad p \quad & \quad q \quad \\ \hline \end{array} \qquad\qquad p \ominus q = \begin{array}{|c|} \hline \quad p \quad \\ \hline q \\ \hline \end{array} .$$

These definitions of picture concatenations can be extended to define two-dimensional languages concatenations; furthermore, by iterating the concatenation operations, we obtain the column and row *closure* or *star*. More precisely: the *column closure* of L (denoted by $L^{*\obar}$) and the *row closure* of L (denoted by $L^{*\ominus}$) are defined respectively as $L^{*\obar} = \bigcup_i L^{i\obar}$ and $L^{*\ominus} = \bigcup_i L^{i\ominus}$ where $L^{0\obar} = \{\lambda_{m,0} \mid m \geq 0\}$, $L^{n\obar} = L^{(n-1)\obar} \obar L$ and $L^{0\ominus} = \{\lambda_{0,m} \mid m \geq 0\}$, $L^{n\ominus} = L^{(n-1)\ominus} \ominus L$. REC family is closed under row and column concatenation and their closures, under union, intersection and under rotation (see [12] for all the proofs).

Example 1. Consider the language L of square pictures over a one-letter alphabet, say $\Sigma = \{a\}$, that is pictures with same number of rows as columns. L is not a local language, but it belongs to REC. Indeed it can be obtained as projection of the local language of squares over the alphabet $\{0,1\}$ in which all the symbols in the main diagonal are 1, whereas the remaining positions carry symbol 0. Below it is given a picture $p \in L$ together with its pre-image p'. The reader can infer the set of all tiles by taking all possible 2×2 sub-pictures of the bordered picture $\widehat{p'}$.

$$p = \begin{array}{|c|c|c|c|} \hline a & a & a & a \\ \hline a & a & a & a \\ \hline a & a & a & a \\ \hline a & a & a & a \\ \hline \end{array} \qquad p' = \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Consider now the language $L_{mult} = \{(m, km) \mid m \geq 0, k \geq 0\}$: we have $L_{mult} = L^{*\obar}$ and, since REC is closed under the column closure, $L_{mult} \in$ REC.

Example 2. Let $\Sigma = \{a, b\}$ and let us consider the language $L_{2col} = \{p \in \Sigma^{*,2} \mid$ there exist $1 \leq i_1 \leq i_2 \leq |p|_r$ such that $p_{(i_1,1)} = p_{(i_2,2)} = b$, $p_{(i,1)} = a$ for every $i \neq i_1$ and $p_{(i,2)} = a$ for every $i \neq i_2\}$. L_{2col} is the language of pictures p with two columns such that there is only one symbol b in each column and the entry that carries the symbol b in the second column of p is not higher than the one in the first column.

$L_{2col} \in$ REC. Indeed it can be obtained as the projection of the local language of two-columns pictures over the alphabet $\{X, Y, \overline{X}, \overline{Y}, X_b, Y_b\}$ such that in the first column (second column, resp.) there is only one occurrence of the symbol X_b (Y_b, resp.) and all the positions above it carry symbol X (Y, resp.) whereas all the positions below it carry symbol $\overline{X}$ ($\overline{Y}$, resp.). The projection π is so defined: $\pi(X) = \pi(Y) = \pi(\overline{X}) = \pi(\overline{Y}) = a$ and $\pi(X_b) = \pi(Y_b) = b$. Below it is given a picture $p \in L_{2col}$ together with its pre-image p'.

$$p = \begin{array}{|c|c|}\hline a & a \\ \hline b & a \\ \hline a & a \\ \hline a & b \\ \hline a & a \\ \hline \end{array} \qquad\qquad p' = \begin{array}{|c|c|}\hline X & Y \\ \hline X_b & Y \\ \hline \overline{X} & Y \\ \hline \overline{X} & Y_b \\ \hline \overline{X} & \overline{Y} \\ \hline \end{array}$$

3 Recognizability Conditions for Picture Languages

In this section we, first, recall two necessary conditions for the recognizability of picture languages, stated in [13] and [5], and, then, we introduce another condition using similar arguments. All these conditions are based on the idea of O. Matz [22] of reducing the problem from two dimensions to one dimension and, then, using some known lower bounds on string languages. Different lower bounds give raise to different necessary conditions. Moreover, they can be expressed using some complexity functions of picture languages that are defined in terms of Hankel matrices associated to the picture languages.

Let $L \subseteq \Sigma^{**}$ be a picture language. For any $m \geq 1$, we can consider the subset $L(m) \subseteq L$ containing all pictures in L with exactly m rows. Note that the language $L(m)$ can be viewed as a string language over the alphabet of the columns of height m. Using this reduction from two dimensions to one dimension, O. Matz stated the following recognizability condition for picture languages.

Lemma 1. *[22] If L is in REC then it is possible to associate to any tiling system recognizing L a family $\{\mathcal{A}_m\}$, where each $\mathcal{A}_m$ is an automaton accepting $L(m)$ with $2^{O(m)}$ states.*

This result is the starting point for all the other necessary conditions we are going to deal with. Using Lemma 1 together with some lower bound on the size of a nondeterministic automaton accepting a regular *string* language, it is possible to obtain the following recognizability condition founded on a property of the language itself. In the follow the cardinality of a set S will be denoted by $|S|$.

Lemma 2. *[22] Let $L \subseteq \Sigma^{**}$, $L \in$ REC and let $\{P_m\}$ be a sequence such that, for any m, $P_m \subseteq \Sigma^{m,*} \times \Sigma^{m,*}$ and*

a) for all $(p,q) \in P_m$, $p \obar q \in L$
b) for all $(p,q),(p',q') \in P_m$, with $(p,q) \neq (p',q')$, $\{p \obar q', p' \obar q\} \not\subseteq L$.
Then $|P_m|$ is $2^{O(m)}$.

Remark that if $P_m \subseteq \Sigma^{m,*} \times \Sigma^{m,*}$ satisfies conditions a) and b) of Lemma 2, then, for all $(p,q),(p',q') \in P_m$, with $(p,q) \neq (p',q')$, we have $p \neq p'$ and $q \neq q'$.

The lower bound used by O. Matz is indeed the technique known in communication complexity as the "extended fooling set" technique (cf. [6]). Furthermore, in [13] one can find a second recognizability condition for picture languages, that, when carefully analyzed, is a consequence of the lower bound known in communication complexity as the "fooling set" technique (cf. [15]). Let us recall these techniques.

Lemma 3. *Let $L \subseteq \Sigma^*$ be a regular language and suppose there exists a set of pairs $S = \{(x_i, y_i) \mid 1 \leq i \leq n\}$ such that $x_i y_i \in L$ for $1 \leq i \leq n$. Then*

i) if $i \neq j$ implies $x_i y_j \notin L$ for $1 \leq i,j \leq n$, then any NFA accepting L has at least n states; here S is called a fooling set*.*

ii) if $i \neq j$ implies $x_i y_j \notin L$ or $x_j y_i \notin L$, for $1 \leq i,j \leq n$, then any NFA accepting L has at least n states; here S is called an extended fooling set*.*

The condition in [13] is formulated in terms of the Hankel matrices of a language and some related complexity functions. Let us recall such notions.

For any string language $L \subseteq \Sigma^*$, one can define the infinite boolean Hankel matrix associated to L, as $M_L = \|a_{\alpha\beta}\|_{\alpha \in \Sigma^*, \beta \in \Sigma^*}$ where $a_{\alpha\beta} = 1$ if and only if $\alpha\beta \in L$ (see [19]). Observe that, when L is a regular language, the number of different rows of M_L is finite. A sub-matrix $M_{(U,V)}$ of an Hankel matrix M_L is a matrix specified by a pair of languages (U,V), with $U,V \subseteq \Sigma^*$, that is obtained by intersecting all rows and all columns of M_L that are indexed by the strings in U and V, respectively.

Remark that, if L is a picture language in REC, then by Lemma 1, for any m, $L(m)$ is a regular language and, therefore, the Hankel matrix $M_{L(m)}$ has a finite number of different rows and columns. In the following examples, for an Hankel matrix M with a finite number of different rows and columns, the sub-matrix indexed by the distinct rows and the distinct columns (in a proper order) will be referred to as the "finite part" of M.

Example 3. Consider again the language $L_{mult} = \{(m, km) \mid m \geq 0, k \geq 0\}$ defined in Example 1 and, for any m, the Hankel matrix $M = M_{L_{mult}(m)}$. M can be obtained as a repetition of the following block both on the rows and on the columns.

$$\begin{array}{c|c|c|c|c|c|c|}
 & (m,1) & (m,2) & \cdots & (m,m-2) & (m,m-1) & (m,m) \\ \hline
(m,0) & 0 & 0 & \cdots & 0 & 0 & 1 \\ \hline
(m,1) & 0 & 0 & \cdots & 0 & 1 & 0 \\ \hline
(m,2) & 0 & 0 & \cdots & 1 & 0 & 0 \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline
(m,m-2) & 0 & 1 & \cdots & 0 & 0 & 0 \\ \hline
(m,m-1) & 1 & 0 & \cdots & 0 & 0 & 0 \\ \hline
\end{array} .$$

The investigation of the growth of some parameters of the Hankel matrices associated to $L(m)$, as functions of m, will provide a general setting for some recognizability conditions. Let us introduce some definitions.

A *permutation matrix* is a boolean matrix that has exactly one 1 in each row and in each column. A boolean matrix A is a *fooling matrix* if there exists a permutation of its rows such that, in the resulting matrix $B = \|b_{ij}\|$, we have, for any i, $b_{ii} = 1$ and, for any i, j with $i \neq j$, if $b_{ij} = 1$ then $b_{ji} = 0$ (cf. [5]). Then, given a picture language L, one can define the following functions from $I\!N$ to $I\!N \cup \{\infty\}$: the *row complexity function* $R_L(m)$ gives the number of distinct rows of $M_{L(m)}$, the *permutation complexity function* $P_L(m)$ gives the size of the maximal permutation matrix that is a sub-matrix of $M_{L(m)}$ (cf. [13]), while the *fooling complexity function* $F_L(m)$ gives the size of the maximal fooling matrix that is a sub-matrix of $M_{L(m)}$ (cf. [5]).

The above mentioned condition in [13] says that: if a picture language L is in REC, then $P_L(m) = 2^{O(m)}$. In the same framework Matz's condition in Lemma 2 becomes: if a picture language L is in REC, then $F_L(m) = 2^{O(m)}$ (cf. [5]).

We are going to deal with the technique known as the "nondeterministic message complexity" technique (cf. [18]; in terms of graphs it is named the "biclique edge cover" technique [16]). Note that the bound given by this technique is always as good as the logarithm of the nondeterministic state complexity, while the gaps between the best bounds for the other two can be arbitrarily large.

Consider a boolean matrix $A = \|a_{ij}\|$; A is a *1-monochromatic matrix* if, for any i, j, $a_{ij} = 1$.

Definition 1. *Let $A = \|a_{ij}\|$ be a boolean matrix and $S = \{A_1, \ldots, A_k\}$ be a set of 1-monochromatic sub-matrices of A. S is a* cover *for A if, for any $1 \leq i, j \leq k$ such that $a_{ij} = 1$, there exists an integer $1 \leq t \leq k$ such that the entry indexed by (i, j) belongs to A_t.*

Definition 2. *Let L be a picture language. The* covering complexity function *$C_L(m)$ gives the cardinality of a minimal cover for $M_{L(m)}$.*

In analogy with the other recognizability conditions, we obtain a new necessary condition for a picture language to be in REC. Its proof proceeds similarly to an analogous one in [16], in the graph setting.

Theorem 1. *Let $L \subseteq \Sigma^{**}$. If $L \in REC$ then $C_L(m)$ is $2^{O(m)}$.*

Proof. If L is in REC then, from Lemma 1, it is possible to associate to any tiling system recognizing L a family $\{\mathcal{A}_m\}$, where, for some constant c, each $\mathcal{A}_m$ is an automaton accepting $L(m)$ with a number of states that is at most c^m. For any m, consider the NFA $\mathcal{A}_m = (Q_m, q_m^0, F_m, \delta_m)$ over the alphabet $\Sigma^{m,1}$. For any $q \in Q_m$ consider the sets X_q and Y_q of strings over $\Sigma^{m,1}$, defined as follows: $X_q = \{x \in (\Sigma^{m,1})^* : q \in \delta_m^*(q_m^0, x)\}$ and $Y_q = \{y \in (\Sigma^{m,1})^* : \delta_m^*(q, y) \cap F_m \neq \emptyset\}$.

Let $M_{L(m)}$ be the Hankel matrix of the language $L(m)$ and denote M_q its sub-matrix specified by the pair of picture/string languages (X_q, Y_q). M_q is a 1-monochromatic sub-matrix of $M_{L(m)}$ since $xy \in L(m)$ for all $x \in X_q$ and $y \in Y_q$.

Moreover $S_m = \{M_q : q \in Q_m\}$ is a cover of $M_{L(m)}$. Indeed, if an entry of $M_{L(m)}$, indexed by a pair $(\overline{x}, \overline{y})$, carries a symbol 1, then $\overline{x}\ \overline{y} \in L(m)$ and then there exists at least a state $\overline{q} \in Q_m$ such that $\overline{q} \in \delta^*_m(q^0_m, \overline{x})$ and $\delta^*_m(\overline{q}, \overline{y}) \cap F_m \neq \emptyset$. Therefore $\overline{x} \in X_{\overline{q}}$ and $\overline{y} \in Y_{\overline{q}}$ and the entry indexed by $(\overline{x}, \overline{y})$ belongs to $M_{\overline{q}}$.

Since $C_L(m)$ is the cardinality of a minimal cover for $M_{L(m)}$, we have $C_L(m) \leq |S_m| = |Q_m|$. But $|Q_m| \leq c^m$ and the result follows. □

Let us conclude the section with an example. Note that, as pointed out in [16], the computation of the bounds given by the three techniques (fooling set, extended fooling set and nondeterministic message complexity) is algorithmically possible, but it is computationally hard. In the sequel, we will show some examples where the calculation is possible, directly or using next Proposition 1.

Example 4. Consider the language $L = L_{mult}$ as defined in Example 1. For any m, the finite part of the Hankel matrix $M = M_{L(m)}$ (see Example 3) is both a maximal permutation matrix and a maximal fooling matrix as sub-matrix of M. Its size is m and, hence, $P_L(m) = F_L(m) = m$. Moreover, it is easy to see that any cover for M must have at least m elements and therefore $C_L(m) = m$.

Consider now the language $L_1 = \{(m, km) \mid m \geq 0, k \geq 1\}$ that is a slight variation of L. For any m, the Hankel matrix $M_1 = M_{L_1(m)}$ can be obtained by adding the column indexed by the picture $(m, 0)$ to the Hankel matrix of $L(m)$. The finite part of M_1 is the following.

$$\begin{array}{c|c|c|c|c|c|c|c|}
 & (m,0) & (m,1) & (m,2) & \cdots & (m,m-2) & (m,m-1) & (m,m) \\ \hline
(m,0) & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ \hline
(m,1) & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ \hline
(m,2) & 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline
(m,m-2) & 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \hline
(m,m-1) & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \hline
(m,m) & 1 & 0 & 0 & \cdots & 0 & 0 & 1 \\ \hline
\end{array} .$$

It is easy to see that the sub-matrix that contains the first m rows and the last m columns of M_1 is a maximal permutation matrix, therefore $P_{L_1}(m) = m$. Moreover, M_1 is a fooling matrix and it is a maximal one. Hence $F_{L_1}(m) = m+1$. At last, it is possible to obtain a cover of M_1 with $m+1$ elements and, it can be easily proved that $C_{L_1}(m) = m+1$. Hence, in line with Theorem 1, $C_{L_1}(m)$ is linear in m, and then $C_{L_1}(m) = 2^{O(m)}$.

4 Comparison of Recognizability Conditions for Picture Languages

In Section 3 we have shown a necessary condition for the recognizability of picture languages, given as a bound on the cover complexity function, and in the same setting as two other known conditions. Now we compare all the three conditions and find that the new condition extends the other ones. Moreover the cover complexity function provides a more accurate bound than the other two, also in the two-dimensional case.

Proposition 1. *Let* $L \subseteq \Sigma^{**}$. *Then, for all* $m \geq 1$, $P_L(m) \leq F_L(m) \leq C_L(m) \leq R_L(m)$.

Proof. Since every permutation matrix is a fooling matrix, it holds $P_L(m) \leq F_L(m)$. Moreover, $F_L(m) \leq C_L(m)$. Indeed, for any m, let $S = \{M_1, M_2, \ldots M_n\}$ be a cover for the Hankel matrix $M_{L(m)}$ and let $A = \|a_{ij}\|$, $1 \leq i, j \leq k$, be a sub-matrix of $M_{L(m)}$ that is a fooling matrix. If A is specified by the pair of picture languages (X, Y) with $X = \{x_1, x_2, \ldots x_k\}$ and $Y = \{y_1, y_2, \ldots y_k\}$, we can suppose w.l.o.g. that, for any $1 \leq i \leq k$, $a_{x_i y_i} = 1$ and, for any $1 \leq i, j \leq k$ with $i \neq j$, if $a_{x_i y_j} = 1$ then $a_{x_j y_i} = 0$. Then consider two entries in A indexed by the pairs (x_i, y_i) and (x_j, y_j), with $i \neq j$: since they carry the symbol 1 and S is a cover of $M_{L(m)}$, these entries belong to some matrix in S. But they cannot belong to the same matrix: S contains only 1-monochromatic matrices and we cannot have, at the same time, $a_{x_i y_j} = a_{x_j y_i} = 1$. Therefore, for any $1 \leq i \leq k$, the entries in A indexed by the pair (x_i, y_i) belong to different elements of S i.e. the size of A is less or equal to the cardinality of S. The last inequality follows from the fact that it is always possible to obtain a cover for the Hankel matrix $M_{L(m)}$ with as many elements as the number of rows. □

Using Proposition 1 and Theorem 1, we obtain another proof of the necessary conditions for the recognizability of picture languages proved in [13] and [5].

Corollary 1. *Let* $L \subseteq \Sigma^{**}$. *If* $L \in REC$ *then* $P_L(m)$ *and* $F_L(m)$ *are* $2^{O(m)}$.

Remark 1. There exist languages such that, for any m, $P_L(m) = F_L(m) = C_L(m)$. Consider for example the language $L = L_{mult}$: we have $P_L(m) = F_L(m) = C_L(m) = m$ (see Example 4).

Theorem 1 extends the conditions in Corollary 1. Nevertheless it is important to consider all the tools we have to prove non-recognizability of a picture language: in some cases the calculation of the values of one complexity function could be simpler than for another one, and at the meantime sufficient to apply a method and disprove recognizability. From Proposition 1, we know that $P_L(m) \leq F_L(m) \leq C_L(m)$. Now we want to understand how bigger can $F_L(m)$ be with respect to $P_L(m)$ and $C_L(m)$ with respect to $F_L(m)$. In [16] it is shown that the gap between the best bounds in the three techniques can be arbitrarily large, for some given string languages. Inspired from the examples of string languages in [16], we will show the analogous result for picture languages.

Proposition 2. *There exists* $L \in REC$ *such that* $P_L(m) = 3$ *and* $F_L(m) = m+2$.

Proof. Consider the language $L = L_{2col}$ of Example 2. For any $m \geq 1$, consider the language $L(m)$ and the corresponding Hankel matrix $M_{L(m)}$. For any $1 \leq i \leq m$, let us denote by p_i the one-column picture, i.e. $p_i \in \Sigma^{m,1}$, with symbol b in its i-th row and symbol a in the other ones, by p the picture $p_1 \oplus p_1$ and by

λ the picture $\lambda_{m,0}$. The Hankel matrix $M_{L(m)}$ is given by the following matrix glued with other $|L|-1$ rows all equal to that one indexed by p, also glued with other $|L|-1$ columns all equal to that one indexed by p and, at last, surrounded by an infinite number of columns and rows of 0s.

$$\begin{array}{c|c|c|c|c|c|c|c|}
 & p_1 & p_2 & p_3 & \cdots & p_m & \lambda & p \\ \hline
p_1 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 \\ \hline
p_2 & 0 & 1 & 1 & \cdots & 1 & 0 & 0 \\ \hline
p_3 & 0 & 0 & 1 & \cdots & 1 & 0 & 0 \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline
p_m & 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ \hline
p & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ \hline
\lambda & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ \hline
\end{array} .$$

Consider now a permutation matrix $A = \|a_{ij}\|$, $1 \leq i,j \leq k$, that is a sub-matrix of $M_{L(m)}$. If A is specified by the pair of string/picture languages (X,Y) with $X = \{x_1, x_2, \ldots x_k\}$ and $Y = \{y_1, y_2, \ldots y_k\}$, we can suppose w.l.o.g. that, for any $1 \leq i \leq k$, $a_{x_i y_i} = 1$ and, for any $1 \leq i,j \leq k$ with $i \neq j$, $a_{x_i y_j} = 0$ i.e. for any $1 \leq i \leq k$, $x_i y_i \in L(m)$ and, for any $1 \leq i,j \leq k$ with $i \neq j$, $x_i y_j \notin L(m)$.

Remark that, for any $1 \leq i \leq k$, we have $x_i, y_i \in \Sigma^{m,0} \cup \Sigma^{m,1} \cup \Sigma^{m,2}$. Since a permutation matrix cannot have two equal rows or columns, we can have only one index i such that $x_i = \lambda$. Moreover, we can have only one index i such that $x_i \in \Sigma^{m,2}$: indeed in the case $x_i \in \Sigma^{m,2}$, since $a_{x_i y_i} = 1$, we must have $x_i y_i \in L(m)$ and this is possible if and only if $y_i = \lambda$. Hence $x_i = x_i y_i \in L(m)$. But all rows in $M_{L(m)}$ indexed in $L(m)$ are equal whereas a permutation matrix cannot have two equal rows. At last, we can have only one index i such that $x_i \in \Sigma^{m,1}$: if there were two different indexes, say i_1 and i_2, such that $x_{i_1}, x_{i_2} \in \Sigma^{m,1}$ then, since $x_{i_1} y_{i_1}, x_{i_2} y_{i_2} \in L(m)$, we must have $x_{i_1} = p_{j_1}$, $x_{i_2} = p_{j_2}$ and $y_{i_2} = p_{j_2'}$ with $j_2 \leq j_2'$. We can suppose w.l.o.g. $j_1 \leq j_2$. But this implies $x_{i_1} y_{j_2} = p_{j_1} p_{j_2'} \in L(m)$ and, hence, the row indexed by $x_{i_1} = p_{j_1}$ would have two symbols 1 that is a contradiction.

Therefore any permutation matrix that is a sub-matrix of $M_{L(m)}$, has size at most 3. The bound is tight: it suffices to consider in the figure above the sub-matrix specified by the pair of picture languages (X,X) with $X = \{p_m, p, \lambda\}$.

For the fooling complexity of L consider the sub-matrix $F = \|f_{ij}\|$ of $M_{L(m)}$ above depicted. F is a fooling matrix: indeed we have $f_{\lambda p} = f_{p\lambda} = 1$ and, for any $1 \leq i \leq m$, $f_{p_i p_i} = 1$. Moreover, for any $1 \leq i,j \leq m$, $f_{p_i p_j} = 1$ if and only if $i < j$ (and in this case $f_{p_i p_j} = 0$) and, for any $1 \leq i \leq m$, $f_{\lambda p_i} = f_{p_i \lambda} = 0$. Since, from Proposition 1, the size of a fooling sub-matrix of $M_{L(m)}$ must be less than or equal to $R_L(m) = m+2$, it follows that F, that has size $m+2$, is a maximal fooling sub-matrix of $M_{L(m)}$ and, hence, $F_L(m) = m+2$. □

Remark 2. Remark that, for the language $L = L_{2col}$ considered in the proof of Proposition 2, we have $C_L(m) = m+2$. Indeed, from Proposition 1, $F_L(m) \leq C_L(m) \leq R_L(m) = m+2$. From $F_L(m) = m+2$, it easily follows $C_L(m) = m+2$.

Proposition 3. *There exists L $\in$REC such that $F_L(m) = 3$ and $C_L(m)$ is logarithmic in m.*

Proof. Consider the language $L = \overline{L_1}$, where L_1 is the language defined in Example 4, $L = \{(m, n) \mid n$ is not a multiple of m or $n = 0\}$. For any m, consider the Hankel matrix $M_{L(m)}$: it can be obtained by exchanging entries 0 with entries 1 in the Hankel matrix for L_1. We will show that $F_L(m) = 3$.

Let F be a fooling sub-matrix of $M_{L(m)}$. Any row of F has one symbol 0 at most, with the exception of one row that can contain two occurrences of 0 at most. Suppose F of size k, for some integer k, and let b_0 the number of symbols 0 that occur in F. Obviously $b_0 \leq k + 1$. The number of symbols 1 that occur in F, apart from the k on the counter-diagonal positions, is $k^2 - k - b_0$. Since B is fooling, it must be $k^2 - k - b_0 \leq b_0$, i.e. recalling $b_0 \leq k + 1$, $k^2 - 3k - 2 \leq 0$ that is $k \leq 3$. Hence $F_L(m) \leq 3$. But it is easy to find a fooling sub-matrix of $M_{L(m)}$ of size 3 (consider for example the sub-matrix specified by the pair (X, Y) with $X = \{(m, 1), (m, 2), (m, 3)\}$ and $Y = \{(m, m - 1), (m, m - 2), (m, m - 3)\}$) and hence the equality $F_L(m) = 3$ follows.

For the covering complexity, one can prove that there exists a strict relation between the Hankel matrix of $L(m)$ and the bipartite graph associated to the string language $L(m)$ and, in particular, that $C_L(m)$ is equal to the bipartite dimension of the graph. The definition of bipartite dimension was introduced in [16], where the authors also show that, for $L(m)$ as above, it is equal to the smallest integer k such that $m \leq \binom{k}{\lfloor k \backslash 2 \rfloor}$ and, then, that it is logarithmic in m. □

Let us now consider a one-letter alphabet. Recall that, given a function f, *the picture language defined by* f, is the set $L_f = \{(m, f(m)) \mid m \geq 0\}$ (cf. [10], see also [12]). In [10], it is proved that, if $f(m)$ is a super-exponential function, then $L_f \notin$ REC. Moreover, in [5] it is shown that, in the same hypothesis, $\overline{L_f} \notin$ REC. This result is obtained by using Lemma 2 and some ad-hoc arguments concerning automata over one-letter alphabet. Remark that the same result could be inferred by some properties of the monadic second-order definable functions [23] and by their correspondence with REC family as stated in [14].

Proposition 4. *The necessary condition stated in Theorem 1 is not sufficient.*

Proof. Examples of languages not in REC with covering complexity less than exponential are all languages $\overline{L_f}$ where $f(m)$ is a super-exponential function and $f(m) = 2^{O(c^m)}$, for some constant m (an example is $f(m) = m!$). Indeed, if $f(m)$ is a super-exponential function, then $\overline{L_f} \notin$ REC [5]. Moreover, for any m, the Hankel matrix associated to the language $\overline{L_f}(m)$ is given by the following matrix "surrounded" by an infinite number of columns and rows of 1's.

$$\begin{array}{c|c|c|c|c|c|c|}
 & (m,0) & (m,1) & \cdots & (m,f(m-2)) & (m,f(m-1)) & (m,f(m)) \\ \hline
(m,0) & 1 & 1 & \cdots & 1 & 1 & 0 \\ \hline
(m,1) & 1 & 1 & \cdots & 1 & 0 & 1 \\ \hline
(m,2) & 1 & 1 & \cdots & 0 & 1 & 1 \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline
(m,f(m-1)) & 1 & 0 & \cdots & 1 & 1 & 1 \\ \hline
(m,f(m)) & 0 & 1 & \cdots & 1 & 1 & 1 \\ \hline
\end{array} .$$

Now it can be proved, with similar reasoning as in Proposition 3, that $C_L(m)$ is logarithmic in $f(m)$ and, hence, it is $O(c^m)$. $\square$

5 Conclusions and Open Problems

In this paper we highlighted the possibility to gain necessary conditions for the recognizability of two-dimensional languages, from lower bound methods for regular string languages. Indeed to estimate the number of states of a minimal NFA for a regular language is still an open problem, while computing such an NFA is PSPACE-complete. This field is therefore an active research area, where problems are tackled by different methods (communication complexity as well as graph theory, for instance). Hopefully, further results in the area could provide new insights also on two-dimensional languages.

In particular, consider the unary language given in the proof of Proposition 4 as an example of a language whose non-recognizability cannot be proved using the function C_L. Its non-recognizability follows from some result on automata over a one-letter alphabet. It should be interesting to see whether for unary languages one can obtain a stronger non-recognizability condition by reformulating in terms of Hankel matrices some known result of the automata theory.

All results given along the paper are based on the investigation of the sequence of languages $L(m)$, the languages of pictures with fixed number of rows m. They can be straightaway translated to get further recognizability conditions concerning the languages of pictures with fixed number of columns. The combination of both bounds (on fixed number of rows and columns) could be a possible way to strengthen the conditions, in such a way to much more exploit the two-dimensional nature of picture languages.

References

1. Anselmo, M., Giammarresi, D., Madonia, M.: Deterministic and Unambiguous Families within Recognizable Two-dimensional Languages. Fund. Inform. 98(2-3), 143–166 (2010)
2. Anselmo, M., Giammarresi, D., Madonia, M.: A computational model for tiling recognizable two-dimensional languages. Theoret. Comput. Sci. 410(37), 3520–3529 (2009)
3. Anselmo, M., Giammarresi, D., Madonia, M., Restivo, A.: Unambiguous Recognizable Two-dimensional Languages. RAIRO: Theoret. Informatics Appl. 40(2), 227–294 (2006)

4. Anselmo, M., Madonia, M.: Deterministic and unambiguous two-dimensional languages over one-letter alphabet. Theoret. Comput. Sci. 410, 1477–1485 (2009)
5. Anselmo, M., Madonia, M.: Classes of two-dimensional languages and recognizability conditions. RAIRO: Theoret. Informatics Appl. (to appear)
6. Birget, J.-C.: Intersection and union of regular languages and state complexity. Inform. Proces. Lett. 43, 185–190 (1992)
7. Bozapalidis, S., Grammatikopoulou, A.: Recognizable picture series. Journal of Automata, Languages and Combinatorics, Special Vol. on Weighted Automata (2004)
8. Crespi Reghizzi, S., Pradella, M.: Tile rewriting grammars and picture languages. Theoret. Comput. Sci. 340(2), 257–272 (2005)
9. De Prophetis, L., Varricchio, S.: Recognizability of rectangular pictures by wang systems. Journal of Automata, Languages, Combinatorics 2, 269–288 (1997)
10. Giammarresi, D.: Two-dimensional languages and recognizable functions. In: Rozenberg, G., Salomaa, A. (eds.) Procs. DLT, vol. 93, pp. 290–301. World Scientific Publishing Co., Singapore (1994)
11. Giammarresi, D., Restivo, A.: Recognizable picture languages. Int. Journal Pattern Recognition and Artificial Intelligence 6(2-3), 241–256 (1992)
12. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. III, pp. 215–268. Springer, Heidelberg (1997)
13. Giammarresi, D., Restivo, A.: Matrix based complexity functions and recognizable picture languages. In: Grädel, E., Flum, J., Wilke, T. (eds.) Logic and Automata: History and Perspectives. Texts in Logic and Games, vol. 2, pp. 315–337. University Press, Amsterdam (2007)
14. Giammarresi, D., Restivo, A., Seibert, S., Thomas, W.: Monadic second order logic over pictures and recognizability by tiling systems. Inform. and Comput. 125(1), 32–45 (1996)
15. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Proces. Lett. 59, 75–77 (1996)
16. Gruber, H., Holzer, M.: Finding Lower Bounds for Nondeterministic State Complexity Is Hard. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 363–374. Springer, Heidelberg (2006)
17. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation, 2nd edn. Addison-Wesley, Reading (2000)
18. Hromkovic, J.: Communication Complexity and Parallel Computing. Springer, Heidelberg (1997)
19. Hromkovic, J., Karhumäki, J., Klauck, H., Schnitger, G., Seibert, S.: Communication Complexity Method for Measuring Nondeterminism in Finite Automata. Inform. and Comput. 172, 202–217 (2002)
20. Inoue, K., Takanami, I.: A characterization of recognizable picture languages. In: Nakamura, A., Saoudi, A., Inoue, K., Wang, P.S.P., Nivat, M. (eds.) ICPIA 1992. LNCS, vol. 654, pp. 133–143. Springer, Heidelberg (1992)
21. Lonati, V., Pradella, M.: Picture Recognizability with Automata Based on Wang Tiles. In: van Leeuwen, J., Muscholl, A., Peleg, D., Pokorný, J., Rumpe, B. (eds.) SOFSEM 2010. LNCS, vol. 5901, pp. 576–587. Springer, Heidelberg (2010)
22. Matz, O.: On piecewise testable, starfree, and recognizable picture languages. In: Nivat, M. (ed.) FOSSACS 1998. LNCS, vol. 1378, pp. 203–210. Springer, Heidelberg (1998)
23. Matz, O., Schweikardt, N., Thomas, W.: The monadic quantifier alternation hierarchy over grids and graphs. Inform. and Comput. 179, 356–383 (2002)

Typed Monoids – An Eilenberg-Like Theorem for Non Regular Languages

Christoph Behle, Andreas Krebs, and Stephanie Reifferscheid[⋆]

Wilhelm-Schickard-Institut, Universität Tübingen
{behlec,krebs,reiffers}@informatik.uni-tuebingen.de

Abstract. Based on different concepts to obtain a finer notion of language recognition via finite monoids we develop an algebraic structure called typed monoid. This leads to an algebraic description of regular and non regular languages.

We obtain for each language a unique minimal recognizing typed monoid, the typed syntactic monoid. We prove an Eilenberg-like theorem for varieties of typed monoids as well as a similar correspondence for classes of languages with weaker closure properties than varieties.

1 Introduction

We present an algebraic viewpoint on the study of formal languages and extend the known approach to use language recognition via monoids and morphisms by equipping the monoids with additional information to obtain a finer notion of recognition.

In the well established algebraic theory of automata, a language $L \subseteq \Sigma^*$ is recognized by a monoid M if there exists a morphism $h : \Sigma^* \to M$ and a subset A of M such that $L = h^{-1}(A)$. The syntactic monoid, which is the smallest monoid recognizing L, has emerged as an important tool to study classes of regular languages. However, even for regular languages this instrument lacks precision, and it is poorly suited to the study of non regular languages. Although there has been progress to extend the theory to infinite monoids that are torsion using topological methods [18].

One problem is that not only the syntactic monoid but also the syntactic morphism, i.e. the morphism recognizing L via the syntactic monoid, plays an important role: consider the two languages L_{parity}, L_{even} over the alphabet $\Sigma = \{a, b\}$ where the first one consists of all words with an even number of a's while the latter consists of all words of even length. Both languages have C_2, the cyclic group of order two, as syntactic monoid. But there are well-known results [9,1] that in the model of circuit complexity classes the language L_{parity} is harder than L_{even}. This reflects in the recognition via morphism through the fact that

[⋆] Supported by Proyecto MTM2010-19938-C03-02, Ministerio de Ciencia y Tecnologia, Spain.

in L_{parity} the morphism needs to distinguish between an “a” as input and a “b” as input, whereas in L_{even} we count only the number of inputs.

So one of our goals is to introduce an algebraic structure where we have control over the morphisms allowed, especially over the images of single letters. The concept of limiting the images of morphisms in the setting of finite monoids has been studied in [8]. It was shown that this concept is useful and yields nice algebraic objects. A similar approach but with a different algebraic object was used in [17].

Limiting the morphisms is a useful tool, but especially if we want to study non regular languages we need to add more information. Consider the language L_{Maj} over the alphabet $\Sigma = \{a, b\}$ of all words with more a's than b's. This is clearly a non regular language and hence not recognized by a finite monoid. The syntactic monoid of L_{Maj} is $\mathbb{Z}$, but $\mathbb{Z}$ is also the syntactic monoid of an undecidable language in unary encoding.

The standard approach to recognize L_{Maj} by $\mathbb{Z}$ is the following: $\eta : \Sigma^* \to \mathbb{Z}$ is defined by $\eta(a) = +1$ and $\eta(b) = -1$. Then $w \in L \Leftrightarrow \eta(w) > 1$. Hence L_{Maj} is recognized by $\mathbb{Z}, \eta$, and the accepting set $\mathbb{Z}^+$ (the positive numbers). While in general various, even undecidable, languages can be recognized by $\mathbb{Z}$ with suitable accepting sets, what happens if we restrict the accepting set to $\mathbb{Z}^+$? Let $h : \Sigma^* \to \mathbb{Z}$ be a morphism, then $h(w) = h(a) \cdot \#_a(w) + h(b) \cdot \#_b(w)$, since $\mathbb{Z}$ is commutative. Hence, any language accepted by such a morphism is defined by a linear inequality of the ratio of the letters occurring.

The idea to limit the allowed accepting subset has been studied in [20] and applied to context free languages. We use a different approach here: instead of one accepting subset, we will consider a set of subsets and then consider the Boolean algebra generated by these sets. Each element of that Boolean algebra can be an accepting subset. We loose precision there, because this forces us to be closed under complementation, but our aim is the application of our approach to circuit complexity and descriptive complexity. The use of a Boolean algebra helps a lot to obtain a neat definition for the block product which is an important tool to characterize such classes algebraically.

If we combine the two approaches, i.e. fix the set of acceptance subsets of the monoid and limit the allowed morphisms we obtain even better possibilities. If we take $\mathbb{Z}$ and allow only morphisms mapping a letter to $\{-1, +1\}$, and have $\mathbb{Z}^+$ as accepting subset then we can only recognize languages that partition the alphabet into two sets A and B and test if the letters of set A occur more often than the letters of set B. All these languages are “close” to L_{Maj} in the sense that they reduce by a length preserving morphism to it.

These two observations lead us to the definition of a typed monoid. A *typed monoid* is a triple $(S, \mathfrak{S}, \mathcal{E})$, where S is a finitely generated monoid, $\mathfrak{S}$ is a finite Boolean algebra over S, and $\mathcal{E} \subseteq S$ is a finite set. The elements of $\mathfrak{S}$ are called *types* and the elements of $\mathcal{E}$ are called *units* (see Definition 1).

A language is recognized by $(S, \mathfrak{S}, \mathcal{E})$ if there is a morphism h from $\Sigma^* \to S$, $h(\Sigma) \subseteq \mathcal{E}$, and $L = h^{-1}(\boldsymbol{\mathcal{S}})$ for a type $\boldsymbol{\mathcal{S}} \in \mathfrak{S}$. More generally, we use the units to limit the allowed morphisms while only types may be acceptance sets.

The syntactic monoid plays an important role in the study of (regular) languages. In the theory of finite monoids it can be shown that the syntactic monoid is the smallest monoid recognizing a language with respect to division. Furthermore, if a monoid M divides a monoid N, then all languages recognized by M are recognized by N and hence the partial order of division on monoids is meaningful in terms of language recognition. We will show the same properties for our typed monoids. On the other hand, we are even able to distinguish the typed syntactic monoids of languages like majority and prime numbers, although they have the same syntactic monoid, namely $\mathbb{Z}$, in the conventional case.

In the finite case the theorem of Eilenberg, stating a one-to-one correspondence between varieties of (finite) monoids and varieties of (regular) languages, is the origin of the algebraic study of formal languages. Recall that by results of Schützenberger and McNaughton and Papert [21,14] the (regular) starfree languages are exactly the languages that can be recognized by (finite) aperiodic monoids, i.e. monoids that contain only trivial subgroups. This result lead to the decidability of the question whether a given regular language is starfree, and there are similar results for many varieties of regular languages.

In Section 6 we will define varieties of typed monoids and formulate a version of Eilenberg's theorem for typed monoids. Using this it is possible to obtain algebraic counterparts of varieties of (non-regular) formal languages [13,12].

Another important tool in the study of formal languages is their description via logic. So it was shown that the starfree languages are exactly the languages describable by a first order logic fragment, namely FO[<]. The fact that FO is connected to the circuit class AC^0 which cannot recognize a group language [1,9] led to an investigation of the links between subclasses of regular languages and classes of circuits. These studies exhibited some interesting connections between some varieties of regular languages and certain circuit classes. For a survey we recommend [23].

While varieties play an important role, many language classes defined by logic classes do not form a variety. Consider for instance FO[<, mod]: this class can describe all words of even length L_{even}, but it is known [1,9] that this class cannot express L_{parity}. Hence FO[<, mod] cannot be closed under arbitrary inverse morphisms, because there is a non length preserving morphism h such that $L_{parity} = h^{-1}(L_{even})$.

Eilenberg studied classes of transformation semigroups with weaker closure properties than varieties. We define weakly closed classes of typed monoids in Section 5 and show that for each weakly closed class of languages there exists a corresponding weakly closed class of typed monoids. This gives a weaker version of the variety theorem for classes of languages with closure properties like FO[<, mod].

We sum up the structure of the paper: in Section 3 we give the basic definition s of our algebraic objects, morphisms, and language recognition. In the following sections we transfer the Eilenberg program for (finite) monoids into the world of typed monoids. The typed syntactic monoid and its minimality are treated in Section 4. We define closure properties in Section 5 and show that weakly closed classes of typed monoids correspond to weakly closed classes of languages. In Section 6 we consider varieties and show that the correspondence of the previous section is one-to-one for varieties.

2 Preliminaries

We define here most notions in a very basic way and presume the reader to be familiar with Eilenberg's variety theory, namely the concepts of language recognition via monoids and varieties. For a complete survey of monoids and language recognition we recommend [15]. For readers who wish an in-depth study of monoids and varieties [2] is a good source. We will bring up some logic classes from time to time, mostly used in examples and as motivation for some classes of languages. The reader can skip these parts but for those interested we refer to [22]. Most examples can be found there and we use the same notation. It also displays the connections of descriptive complexity, circuit complexity, and algebra.

Let A, B be nonempty sets; A mapping $f : A \mapsto B$ is called *surjective* if $h(A) = B$ and *injective* if for every $b \in B$ there is at most one $a \in A$ such that $h(a) = b$; f is *bijective* iff f is injective and surjective.

A *semigroup* S is a nonempty set equipped with a binary relation $\cdot$ which is associative. We call a semigroup M a *monoid* if it has a (unique) *neutral element* 1_M, i.e. an element such that $1_M \cdot x = x \cdot 1_M = x$ for all $x \in M$. A monoid G is called *group* if for each $g \in G$ there is a (unique) element $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = 1_G$. As usual we write xy for $x \cdot y$ and 1 for 1_M if the context is understood.

A monoid M is *finite* if M is a finite set. A subset P of M *generates* M, denoted by $M = \langle P \rangle$, if each element in M can be written as a finite product of elements in P and the neutral element. M is *finitely generated* if there exists a finite subset P of M generating M. The *free monoid* with generator set A is usually denoted by A^*, i.e. we take the set of all finite sequences of elements of A together with the empty word as neutral element, the multiplication is defined as concatenation. Let M, N be monoids, a *(monoid-)morphism* h from M to N is a mapping $h : M \to N$ such that for all $m_1, m_2 \in M\colon h(m_1 m_2) = h(m_1)h(m_2)$ and $h(1) = 1$. A subset M' is a *submonoid* of M, if $xy \in M'$ for all $x, y \in M'$ and $1 \in M'$. A monoid N *divides* a monoid M ($N \preceq M$) if it is a morphic image of a submonoid of M, that is there exists a submonoid M' of M and a surjective morphism from M' to N.

A *congruence* on a monoid M is an equivalence relation $\sim$ on M, such that $x \sim y$ and $x' \sim y'$ imply $xx' \sim yy'$. We denote by $[x]$ the congruence class of x, i.e. $\{y \mid y \sim x\}$. Given a congruence $\sim$ on M, the set of equivalence classes together with the multiplication $[x][y] = [xy]$ form a monoid, the *quotient monoid* denoted by $M/\sim$. The mapping $x \mapsto [x]$ is a morphism, the so called *canonical epimorphism.* Conversely, a morphism $h : M \to N$ defines a congruence $\sim_h$ on M by setting $x \sim_h y \Leftrightarrow h(x) = h(y)$.

The set of integers $\mathbb{Z}$ with the usual addition is an infinite monoid generated by $\{-1, 1\}$; for each natural number k we denote the quotient monoid $\mathbb{Z}/k\mathbb{Z}$

of $\mathbb{Z}$ (corresponding to the congruence $x \sim y$ iff $x \equiv y \mod k$) by C_k and its elements by $0, \dots, k-1$.

A *Boolean algebra* $\mathfrak{B}$ over a nonempty set S is a finite set of subsets of S containing $\emptyset, S$ and being closed under union, intersection and complement. Let $\mathfrak{B}, \mathfrak{C}$ be two Boolean algebras over sets S and T respectively. A *morphism* h from $\mathfrak{B}$ to $\mathfrak{C}$ is a mapping $h : \mathfrak{B} \to \mathfrak{C}$ such that $h(\emptyset) = \emptyset$, $h(S) = T$, $h(S \setminus B_1) = T \setminus h(B_1)$ and $h(B_1 \cup B_2) = h(B_1) \cup h(B_2)$ for all $B_1, B_2 \in \mathfrak{B}$.

If $\mathfrak{B}$ is a Boolean algebra over S and $\mathfrak{C} \subseteq \mathfrak{B}$ is again a Boolean algebra over S, then $\mathfrak{C}$ is a Boolean subalgebra of $\mathfrak{B}$.

Languages. An *alphabet* Σ is a finite non-empty set and its elements are called *letters*. The elements of the *free* monoid Σ^* are called *words* and a *language* is a subset of Σ^*. Let w be a word then we can uniquely write it as a product of letters $w = w_1 \dots w_n$, where n is called the *length* of the word, denoted by $|w|$. For a letter a and a word w we define $\#_a(w) = |\{i \mid w_i = a\}|$, i.e. the number of occurrences of the letter a in w. A morphism h from Σ^* to Δ^* is called length preserving if $h(\Sigma) \subseteq \Delta$.

A monoid M *recognizes* a language $L \subseteq \Sigma^*$ iff there is a morphism $h : \Sigma^* \to M$ and a subset $A \subseteq M$ such that $L = h^{-1}(A)$.

Given a language $L \subseteq \Sigma^*$ the *syntactic congruence* is defined on Σ^* by $x \equiv_L y$ iff for all $w, v \in \Sigma^*$ holds $wxv \in L \Leftrightarrow wyv \in L$. The *syntactic monoid* $M(L)$ of L is defined as the quotient monoid $\Sigma^* / \equiv_L$. The canonical epimorphism for this congruence is called *syntactic morphism* and denoted by η_L. It can be shown that the syntactic monoid is a minimal monoid recognizing L with respect to division.

3 Typed Monoids

In this section we introduce the notion of typed monoids. We start from the usual concept of language recognition by monoids. Instead of only considering the monoid M we want control over the possible morphisms, a concept already studied in [8,17]. We follow the approach of [8] and enhance the monoid with a subset, called units, and require morphisms to map units on units. Further, to reduce the power of the monoid we equip the structure with a finite Boolean algebra over the monoid, and require the accepting subsets to be elements of this Boolean algebra. This is in particular helpful when dealing with infinite monoids or non regular languages.

Definition 1 (Typed Monoid). *A* typed monoid *is a triple* $(S, \mathfrak{S}, \mathcal{E})$*, where* S *is a finitely generated monoid,* $\mathfrak{S}$ *is a finite Boolean algebra over* S*, and* $\mathcal{E} \subseteq S$ *is a finite set. The elements of* $\mathfrak{S}$ *are called* types *and the elements of* $\mathcal{E}$ *are called* units.

For a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ we will write $(S, \{\boldsymbol{\mathcal{S}}_1, \dots, \boldsymbol{\mathcal{S}}_d\}, \mathcal{E})$, where the $\boldsymbol{\mathcal{S}}_i$ are types generating $\mathfrak{S}$. If the Boolean algebra $\mathfrak{S}$ is generated by a single set we will drop the braces: $(S, \{\boldsymbol{\mathcal{S}}\}, \mathcal{E}) = (S, \boldsymbol{\mathcal{S}}, \mathcal{E})$. We call a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ *free*, if the underlying monoid S is free.

Please note, that we do not require the units to generate S.

Example 1. We give some examples of typed monoids.

(a) Let $S = \mathbb{Z}$, $\mathfrak{S} = \{\emptyset, \mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}_0^-\}$ (where $\mathbb{Z}^+$ are the positive numbers and $\mathbb{Z}_0^-$ are the non positive numbers), and $\mathcal{E} = \{-1, 1\}$. Then $(S, \mathfrak{S}, \mathcal{E})$ is a typed monoid. As stated above we use for $(S, \mathfrak{S}, \mathcal{E})$ the short hand notation $(\mathbb{Z}, \{\mathbb{Z}^+, \mathbb{Z}_0^-\}, \{-1, 1\})$ or even $(\mathbb{Z}, \mathbb{Z}^+, \{-1, 1\})$ to denote $(S, \mathfrak{S}, \mathcal{E})$.
(b) Let $S = \mathbb{Z}$, $\mathfrak{S} = \{\emptyset, 2\mathbb{Z}, 2\mathbb{Z}+1, \mathbb{Z}\}$, and $\mathcal{E} = \{0, 1\}$. Then $(S, \mathfrak{S}, \mathcal{E})$ is a typed monoid. Again, we use the short hand notation $(\mathbb{Z}, \{2\mathbb{Z}, 2\mathbb{Z}+1\}, \{0, 1\})$ or even $(\mathbb{Z}, 2\mathbb{Z}, \{0, 1\})$ to denote $(S, \mathfrak{S}, \mathcal{E})$.
(c) Let $S = C_4$, $\mathfrak{S} = \{\emptyset, \{0\}, \{1, 2, 3\}, \{0, 1, 2, 3\}\}$, and $\mathcal{E} = \{1\}$. Then $(S, \mathfrak{S}, \mathcal{E})$ is a finite typed monoid. As we will see later this monoid is more restricted than C_4 in the untyped world.
(d) Each finite monoid Scan be seen as a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ where the types are the subsets of S (that is, $\mathfrak{S}$ is the powerset of S) and the set of units is $\mathcal{E} = S$.
(e) There is a strong connection between a language $L \subseteq \Sigma^*$ and typed monoids. The structure $(\Sigma^*, \{\emptyset, L, \Sigma^* \setminus L, \Sigma^*\}, \Sigma) = (\Sigma^*, L, \Sigma)$ is a typed monoid. Note, that a language L and its complement $\bar{L}$ lead to the same typed monoid.

As mentioned before the units will limit the set of possible morphisms. On the free monoid Σ^* we usually pick Σ as the units; in this case every element has a unique representation as a product of the units and the "length" of an element is the number of units in its representation. Even on arbitrary monoids units give a kind of length property, this has been studied in [8].

We define the notion of a morphism for typed monoids.

Definition 2 (Typed Morphism). *We let a* (typed) morphism $h : (S, \mathfrak{S}, \mathcal{E}) \rightarrow (S', \mathfrak{S}', \mathcal{E}')$ of typed monoids *be specified by a triple $(h_S, h_{\mathfrak{S}}, h_{\mathcal{E}})$, where $h_S : S \rightarrow S'$ is a monoid morphism, $h_{\mathfrak{S}} : \mathfrak{S} \rightarrow \mathfrak{S}'$ is a morphism of Boolean algebras, $h_{\mathcal{E}} : \mathcal{E} \rightarrow \mathcal{E}'$ is a mapping of sets, and the triple fulfills the two compatibility requirements:*

1. *$\forall \boldsymbol{\mathcal{S}} \in \mathfrak{S}$ it holds $h_S(\boldsymbol{\mathcal{S}}) = h_{\mathfrak{S}}(\boldsymbol{\mathcal{S}}) \cap h_S(S)$,*
2. *$\forall u \in \mathcal{E}$ it holds $h_S(u) = h_{\mathcal{E}}(u)$.*

Because of the compatibility conditions of this definition we can omit the indices of the morphisms.

Condition 2 forces h_S to map units to units and to be compatible with $h_{\mathcal{E}}$, and thus the compatibility conditions imply that h_S induces $h_{\mathcal{E}}$ and - in case h_S is surjective - also $h_{\mathfrak{S}}$. Note further that 1 implies that a nonempty type cannot be mapped to the empty type, and thus $h_{\mathfrak{S}}$ needs to be injective, i.e. $|\mathfrak{S}| \leq |\mathfrak{S}'|$.

The definitions of injective and surjective morphisms are straightforward.

Definition 3 (Injective, Surjective, Typed Submonoid, Division) *Let $(S, \mathfrak{S}, \mathcal{E}), (T, \mathfrak{T}, \mathcal{F})$ be two typed monoids.*

– *A typed morphism $h : (S, \mathfrak{S}, \mathcal{E}) \to (T, \mathfrak{T}, \mathcal{F})$ with $h = (h_S, h_{\mathfrak{S}}, h_{\mathcal{E}})$ is* injective *(resp. surjective, bijective) if $h_S, h_{\mathfrak{S}}$, and $h_{\mathcal{E}}$ are all injective (resp. surjective, bijective).*

  If h is bijective we say that $(S, \mathfrak{S}, \mathcal{E})$ and $(T, \mathfrak{T}, \mathcal{F})$ are isomorphic *(denoted $\cong$).*
– *A typed monoid $(T, \mathfrak{T}, \mathcal{F})$ is a* typed submonoid *of $(S, \mathfrak{S}, \mathcal{E})$ (we write $(T, \mathfrak{T}, \mathcal{F}) \leq (S, \mathfrak{S}, \mathcal{E})$) if T is a submonoid of S and there is an injective morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$.*
– *A typed monoid $(T, \mathfrak{T}, \mathcal{F})$* divides *a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ (we write $(T, \mathfrak{T}, \mathcal{F}) \preceq (S, \mathfrak{S}, \mathcal{E})$) if $(T, \mathfrak{T}, \mathcal{F})$ is a morphic image of a typed submonoid of $(S, \mathfrak{S}, \mathcal{E})$.*

As one should expect, concatenation of two (injective/surjective) morphisms is again a (injective/surjective) morphism. And given a typed morphism $h : (S, \mathfrak{S}, \mathcal{E}) \to (T, \mathfrak{T}, \mathcal{F})$, the image (preimage) of h is a submonoid of $(T, \mathfrak{T}, \mathcal{F})$ ($(S, \mathfrak{S}, \mathcal{E})$).

To clarify the notion of a typed morphism consider the following typed monoids. We let

$$(S, \mathfrak{S}, \mathcal{E}) = (C_4, \{\{0\}, \{1\}, \{2\}, \{3\}\}, \{0, 1\}),$$

$$(T, \mathfrak{T}, \mathcal{F}) = (C_4, \{\{0, 2\}, \{1, 3\}\}, \{0, 1\}), \text{ and}$$

$$(U, \mathfrak{U}, \mathcal{G}) = (C_2, \{\{0\}, \{1\}\}, \{0, 1\}).$$

Since $|\mathfrak{S}| > |\mathfrak{T}|$, there is no typed morphism from $(S, \mathfrak{S}, \mathcal{E})$ to $(T, \mathfrak{T}, \mathcal{F})$. On the other hand the identity mapping yields an injective (but not surjective) typed morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$. Hence, $(T, \mathfrak{T}, \mathcal{F})$ is a typed submonoid of $(S, \mathfrak{S}, \mathcal{E})$.

There is no typed morphism from $(U, \mathfrak{U}, \mathcal{G})$ to $(S, \mathfrak{S}, \mathcal{E})$: assume h is such a morphism. Then $h_{\mathcal{G}}(0) = h_U(0) = 0$ and $h_{\mathcal{G}}(1)$ can be 0 or 1. In the first case we hurt condition 1. Because then the following equation should hold: $0 \in h_{\mathfrak{U}}(\{0\}) \cap h_{\mathfrak{U}}(\{1\}) = h_{\mathfrak{U}}(\emptyset) = \emptyset$; contradiction. If we let $h_U(1) = 1$ then h_U is not a (monoid) morphism; in particular $(U, \mathfrak{U}, \mathcal{G})$ is no submonoid of $(S, \mathfrak{S}, \mathcal{E})$. Conversely, $(U, \mathfrak{U}, \mathcal{G})$ is a factor of $(S, \mathfrak{S}, \mathcal{E})$, since the mapping $h_T : (T, \mathfrak{T}, \mathcal{F}) \to (U, \mathfrak{U}, \mathcal{G})$, $i \mapsto i \pmod 2$ defines a typed morphism from $(T, \mathfrak{T}, \mathcal{F})$ onto $(U, \mathfrak{U}, \mathcal{G})$, and $(T, \mathfrak{T}, \mathcal{F})$ is a submonoid of $(S, \mathfrak{S}, \mathcal{E})$.

Lemma 1. *If $(S, \mathfrak{S}, \mathcal{E}) \preceq (T, \mathfrak{T}, \mathcal{F})$ and $(T, \mathfrak{T}, \mathcal{F}) \preceq (U, \mathfrak{U}, \mathcal{G})$, then $(S, \mathfrak{S}, \mathcal{E}) \preceq (U, \mathfrak{U}, \mathcal{G})$.*

Proof. Suppose that $(S, \mathfrak{S}, \mathcal{E}) \preceq (T, \mathfrak{T}, \mathcal{F})$ and $(T, \mathfrak{T}, \mathcal{F}) \preceq (U, \mathfrak{U}, \mathcal{G})$. By definition of divisor we have submonoids $(T', \mathfrak{T}', \mathcal{F}') \leq (T, \mathfrak{T}, \mathcal{F})$ and $(U', \mathfrak{U}', \mathcal{G}') \leq (U, \mathfrak{U}, \mathcal{G})$ and are in the following situation:

$$\begin{tikzcd}
(S,\mathfrak{S},\mathcal{E}) & (T,\mathfrak{T},\mathcal{F}) & (U,\mathfrak{U},\mathcal{G}) \\
(T',\mathfrak{T}',\mathcal{F}') \arrow[u, two heads, "\alpha"] \arrow[ur, hook] & (U',\mathfrak{U}',\mathcal{G}') \arrow[u, two heads, "\beta"] \arrow[ur, hook] & \\
(U'',\mathfrak{U}'',\mathcal{G}'') \arrow[u, dotted, two heads] \arrow[ur, dotted, hook] & &
\end{tikzcd}$$

In the diagram above we define $(U'',\mathfrak{U}'',\mathcal{G}'') = \beta^{-1}((T',\mathfrak{T}',\mathcal{F}'))$, then $\alpha \circ \beta$ maps $(U'',\mathfrak{U}'',\mathcal{G}'')$ surjectively on $(S,\mathfrak{S},\mathcal{E})$ and is a submonoid of $(U,\mathfrak{U},\mathcal{G})$, hence $(S,\mathfrak{S},\mathcal{E}) \preceq (U,\mathfrak{U},\mathcal{G})$. $\square$

A concept strongly connected to morphisms is the concept of congruences. Each morphism induces a congruence and vice versa. Congruences on typed monoids need to be compatible with the set of types.

Definition 4 (Typed congruence). *Let $(S,\mathfrak{S},\mathcal{E})$ be a typed monoid and $\sim$ be a congruence on S. Then $\sim$ is a* typed congruence *if* $\forall \boldsymbol{\mathcal{S}} \in \mathfrak{S}, s_1, s_2 \in S : s_1 \sim s_2 \wedge s_1 \in \boldsymbol{\mathcal{S}} \Rightarrow s_2 \in \boldsymbol{\mathcal{S}}$.

In this case we say that $\sim$ is finer than $\mathfrak{S}$.

Example 2. The usual syntactic congruence $\equiv_L$ forms a typed congruence on (Σ^*, L, Σ). To see this we have to show that $\equiv_L$ respects the types, but these are only $\emptyset, L, \bar{L}, \Sigma^*$ and L is a union of congruence classes of $\equiv_L$.

Let $\sim$ be a typed congruence on a typed monoid $(S,\mathfrak{S},\mathcal{E})$. Let $\mathcal{E}/\sim = \{[x]_\sim \mid x \in \mathcal{E}\}$, $\boldsymbol{\mathcal{S}}/\sim = \{[x]_\sim \mid x \in \boldsymbol{\mathcal{S}}\}$ and $\mathfrak{S}/\sim = \{\boldsymbol{\mathcal{S}}/\sim \mid \boldsymbol{\mathcal{S}} \in \mathfrak{S}\}$. This is well defined, since $\sim$ is finer than $\mathfrak{S}$. We call $(S,\mathfrak{S},\mathcal{E})/\sim = (S/\sim, \mathfrak{S}/\sim, \mathcal{E}/\sim)$ the typed quotient monoid of $(S,\mathfrak{S},\mathcal{E})$ by $\sim$.

As in the classical case, a typed morphism h defines a typed congruence via $s_1 \sim_h s_2 \Leftrightarrow h_S(s_1) = h_S(s_2)$. Likewise, a typed congruence $\sim$ on a typed monoid $(S,\mathfrak{S},\mathcal{E})$ defines a morphism from $(S,\mathfrak{S},\mathcal{E})$ to $(S,\mathfrak{S},\mathcal{E})/\sim$.

4 Typed Syntactic Monoid

We turn now to the recognition of languages by typed monoids. Instead of considering all morphisms into a monoid we limit the allowed morphisms to those mapping letters to units, and the only accepting subsets allowed are types. We can reduce the definition of language recognition to the notion of a typed morphism:

Definition 5. *We say that $(S,\mathfrak{S},\mathcal{E})$* recognizes a language *$L \subseteq \Sigma^*$ if there is a typed morphism $h : (\Sigma^*, L, \Sigma) \to (S,\mathfrak{S},\mathcal{E})$.*

Since L and $\bar{L}$ are disjoint types they have to be mapped on two disjoint types. Hence, there is a type $\boldsymbol{\mathcal{S}} \in \mathfrak{S}$, such that $L = h_S^{-1}(\boldsymbol{\mathcal{S}})$. So the definition resembles the usual definition of language recognition via monoids. The standard notion of language recognition for finite monoids coincides with our notion in the following way: given a finite monoid S then S and the typed monoid $(S, \mathcal{P}(S), S)$ recognize exactly the same languages.

Example 3. Let $L \subseteq \Sigma^*$ be a language, $M(L)$ its usual syntactic monoid, η_L the syntactic morphism, and $A = \eta_L(L)$ the accepting subset. Then L is recognized by the typed monoid $(M(L), A, \eta_L(\Sigma)) = (\eta_L(\Sigma^*), \eta_L(L), \eta_L(\Sigma))$.

The typed syntactic monoid will be defined using the syntactic congruence which is a typed congruence on the typed monoid (Σ^*, L, Σ).

Definition 6 (Typed Syntactic Monoid). *Let $L \subseteq \Sigma^*$ be a language, then $syn(L) = (\Sigma^*, L, \Sigma)/\equiv_L$ is the* typed syntactic monoid *of L.*

It is easy to see that L is recognized by its typed syntactic monoid via the typed morphism induced by the syntactic morphism η_L. We call that morphism the *typed syntactic morphism.*

In the finite case it is known that the syntactic monoid is the unique (up to isomorphism) minimal monoid recognizing the language (minimal with respect to division). Since in general, division is only a preorder (see example 4), in the classical case we only have that $M(L)$ is a minimal element recognizing the language L, i.e.: if N is a monoid recognizing L, then $M(L)$ divides N.

Example 4. Let S and T be the free monoid with two and three generators, respectively. We define the typed monoids $(S, \mathfrak{S}, \mathcal{E}) = (\langle a, b\rangle, \{\langle a, b\rangle\}, \{1\})$ and $(T, \mathfrak{T}, \mathcal{F}) = (\langle a, b, c\rangle, \{\langle a, b, c\rangle\}, \{1\})$. Then S and T divide each other but are not isomorphic: $\langle a, b, c\rangle$ is isomorphic to the submonoid $\langle a, ab, abb\rangle$ (this remains true with the given types and units) and $\langle a, b\rangle$ is isomorphic to the submonoid $\langle a, b\rangle$ of $\langle a, b, c\rangle$ (this also remains true for the given types and units).

We want to show that syn(L) divides $(S, \mathfrak{S}, \mathcal{E})$ if $(S, \mathfrak{S}, \mathcal{E})$ recognizes L.

Lemma 2. *If there is a surjective morphism $\beta : (S, \mathfrak{S}, \mathcal{E}) \to (S', \mathfrak{S}', \mathcal{E}')$, then every morphism from the free monoid $(T, \mathfrak{T}, \mathcal{F})$, where $\mathcal{F}$ is the standard generator set for T, to $(S', \mathfrak{S}', \mathcal{E}')$ factors though β. That is for every morphism α, there is a morphism α' such that the following diagram commutes, i.e. $\beta \circ \alpha' = \alpha$:*

$$\begin{tikzcd}
 & (T, \mathfrak{T}, \mathcal{F}) \arrow[dl, dashed, "\alpha'"'] \arrow[dr, "\alpha"] & \\
(S, \mathfrak{S}, \mathcal{E}) \arrow[rr, two heads, "\beta"] & & (S', \mathfrak{S}', \mathcal{E}')
\end{tikzcd}$$

Proof. We need to define the typed morphism $\alpha' = (\alpha'_T, \alpha'_{\mathfrak{T}}, \alpha'_{\mathcal{F}})$. For each unit $s' \in \mathcal{E}'$ there is an element $s \in \mathcal{E}$ with $\beta(s) = s'$. For each s' pick such an element and call it $s_{s'}$. We define $\alpha' : (T, \mathfrak{T}, \mathcal{F}) \to (S, \mathfrak{S}, \mathcal{E})$ by setting $\alpha'_T(t) = s_{\alpha(t)}$ for all free generators t of T. Recall that this defines a morphism on T, and thus also the mapping $\alpha'_{\mathcal{F}}$. It remains to define the corresponding mapping for the types. Let $\boldsymbol{\mathcal{T}} \in \mathfrak{T}$ and set $\alpha'_{\mathfrak{T}}(\boldsymbol{\mathcal{T}}) = \max\{\boldsymbol{\mathcal{S}} \in \mathfrak{S} \ : \ \boldsymbol{\mathcal{S}} \cap \alpha'_T(T) = \alpha'_T(\boldsymbol{\mathcal{T}})\}$. Since every $\boldsymbol{\mathcal{S}} \in \beta^{-1}(\alpha(\boldsymbol{\mathcal{T}}))$ is in the above set, this is well defined. ☐

As in the finite case the syntactic monoid is minimal with respect to language recognition. We start by showing the following lemma which is a consequence of Lemma 2:

Lemma 3. *Let $(S, \mathfrak{S}, \mathcal{E})$, $(S', \mathfrak{S}', \mathcal{E}')$ be typed monoids, such that $(S', \mathfrak{S}', \mathcal{E}')$ divides $(S, \mathfrak{S}, \mathcal{E})$. Then every languages recognized by $(S', \mathfrak{S}', \mathcal{E}')$ is also recognized by $(S, \mathfrak{S}, \mathcal{E})$.*

In particular: if $\mathrm{syn}(L)$ *divides $(S, \mathfrak{S}, \mathcal{E})$, then $(S, \mathfrak{S}, \mathcal{E})$ recognizes L.*

Proof. By definition there is a submonoid $(U, \mathfrak{U}, \mathcal{G})$ of $(S, \mathfrak{S}, \mathcal{E})$ and a surjective morphism $\beta : (U, \mathfrak{U}, \mathcal{G}) \to (S', \mathfrak{S}', \mathcal{E}')$. Since $(S', \mathfrak{S}', \mathcal{E}')$ recognizes the language $L \subseteq \Sigma^*$ there is a morphism $\alpha : (\Sigma^*, L, \Sigma) \to (S', \mathfrak{S}', \mathcal{E}')$. Hence by Lemma 2 there is a morphism $\alpha' : (\Sigma^*, L, \Sigma) \to (U, \mathfrak{U}, \mathcal{G})$. Since $(U, \mathfrak{U}, \mathcal{G})$ is a submonoid of $(S, \mathfrak{S}, \mathcal{E})$ there is a morphism $i : (U, \mathfrak{U}, \mathcal{G}) \to (S, \mathfrak{S}, \mathcal{E})$, thus L is recognized by $(S, \mathfrak{S}, \mathcal{E})$ via the morphism $i \circ \alpha'$. ☐

Although in the theory of typed monoids division is not a partial order (see example 4) we can show that the typed syntactic monoid of L is in fact the unique minimal typed monoid recognizing L.

Lemma 4. *Let $L \subseteq \Sigma^*$ be a language and $(S, \mathfrak{S}, \mathcal{E})$ a typed monoid.*

(a) *$(S, \mathfrak{S}, \mathcal{E})$ recognizes L if and only if* $\mathrm{syn}(L)$ *divides $(S, \mathfrak{S}, \mathcal{E})$.*
(b) $\mathrm{syn}(L)$ *is the unique minimal element (with respect to division) recognizing L, i.e. if $(S, \mathfrak{S}, \mathcal{E})$ recognizes L and $(S, \mathfrak{S}, \mathcal{E})$ divides* $\mathrm{syn}(L)$, *then $(S, \mathfrak{S}, \mathcal{E}) \cong$* $\mathrm{syn}(L)$.

Proof. (a) : We only need to show that $\mathrm{syn}(L)$ divides every typed monoid that recognizes $L \subseteq \Sigma^*$. So let $h : (\Sigma^*, L, \Sigma) \to (S, \mathfrak{S}, \mathcal{E})$ be a morphism. Further let $(S', \mathfrak{S}', \mathcal{E}')$ be the image of (Σ^*, L, Σ). We show that there is a surjective morphism α from $(S', \mathfrak{S}', \mathcal{E}')$ to $\mathrm{syn}(L)$ (and hence $\mathrm{syn}(L)$ divides $(S, \mathfrak{S}, \mathcal{E})$).

So we need to show that $\alpha(s) = \eta(h^{-1}(s))$ is well defined. By contradiction assume that there are w_1, w_2 with $h(w_1) = h(w_2)$ and $\eta(w_1) \neq \eta(w_2)$, then there are $u, v \in \Sigma^*$ with $uw_1v \in L$ and $uw_2v \notin L$, but $h(uw_1v) = h(uw_2v)$ and hence L is not recognized by S. It is obvious that α respects units and types and therefore defines a typed morphism.

(b) : Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid recognizing a language $L \subseteq \Sigma^*$ via a morphism h and dividing $\mathrm{syn}(L) := (S_L, \mathfrak{S}_L, \mathcal{E}_L)$. Then there is a submonoid $(S'_L, \mathfrak{S}'_L, \mathcal{E}'_L)$ of $(S_L, \mathfrak{S}_L, \mathcal{E}_L)$, a submonoid $(S', \mathfrak{S}', \mathcal{E}')$ of $(S, \mathfrak{S}, \mathcal{E})$, and morphisms α and β as shown below:

$$\begin{tikzcd}
(\Sigma^*, L, \Sigma) \arrow[r, two heads, "h"] \arrow[d, two heads, "\eta"] & (S', \mathfrak{S}', \mathcal{E}') \arrow[r, tail] \arrow[dl, two heads, "\alpha"] & (S, \mathfrak{S}, \mathcal{E}) \\
(S_L, \mathfrak{S}_L, \mathcal{E}_L) & & \\
(S'_L, \mathfrak{S}'_L, \mathcal{E}'_L) \arrow[u, hook] \arrow[uurr, two heads, "\beta"] & &
\end{tikzcd}$$

Since Σ generates Σ^* and h, η is surjective, we know $\mathcal{E}'$ generates S', and S_L is generated by $\mathcal{E}_L$. Moreover $|\mathcal{E}'| \leq |\mathcal{E}| \leq |\mathcal{E}'_L| \leq |\mathcal{E}_L| \leq |\mathcal{E}'|$, so $|\mathcal{E}'| = |\mathcal{E}| = |\mathcal{E}'_L| = |\mathcal{E}_L|$. In particular $(S_L, \mathfrak{S}_L, \mathcal{E}_L) \cong (S'_L, \mathfrak{S}'_L, \mathcal{E}'_L)$, and thus $\mathcal{E}$ generates S, which again implies $(S, \mathfrak{S}, \mathcal{E}) \cong (S', \mathfrak{S}', \mathcal{E}')$.

Now we have a surjective morphism from $(S, \mathfrak{S}, \mathcal{E})$ to $(S_L, \mathfrak{S}_L, \mathcal{E}_L)$ and converse, but this does not imply that α or β are isomorphisms. But it is clear that $\alpha \circ \beta$ is a permutation of $\mathcal{E}$, hence there is a power of $\alpha \circ \beta$ that is the identity on $\mathcal{E}$. But then this power is also an identity on $(S, \mathfrak{S}, \mathcal{E})$ and we have $(S_L, \mathfrak{S}_L, \mathcal{E}_L) \cong (S, \mathfrak{S}, \mathcal{E})$. □

5 Weakly Closed Classes

Motivated by Eilenberg's notion of a weakly closed class of transformation semigroups we consider weakly closed classes of typed monoids and languages. Weakly closed classes of transformation semigroups are closed under division and direct product. For typed monoids we add an additional operation which allows to identify typed monoids recognizing the same languages.

Definition 7 (Reduced Monoid/Trivial Extension). *Let $(S, \mathfrak{S}, \mathcal{E})$ and $(T, \mathfrak{T}, \mathcal{F})$ be typed monoids such that there exists a surjective morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$. We call $(S, \mathfrak{S}, \mathcal{E})$ a* reduced monoid *of $(T, \mathfrak{T}, \mathcal{F})$ and conversely $(T, \mathfrak{T}, \mathcal{F})$ a* trivial extension *of $(S, \mathfrak{S}, \mathcal{E})$.*

The following lemma formalizes the idea that from a language recognition perspective, reduced monoids and trivial extensions are equivalent.

Lemma 5. *If $(S, \mathfrak{S}, \mathcal{E})$ is a reduced monoid of $(T, \mathfrak{T}, \mathcal{F})$, then they recognize exactly the same languages.*

The proof is straightforward using Lemma 3.

Similar to the case of the syntactic monoid, we can show that for each typed monoid there exists a unique minimal reduced monoid. This can be constructed - as the syntactic monoid - as a quotient monoid: given a typed monoid $(S, \mathfrak{S}, \mathcal{E})$, define the relation $\equiv_{(S,\mathfrak{S},\mathcal{E})}$ by letting $x \equiv_{(S,\mathfrak{S},\mathcal{E})} y$ $(x, y \in S)$ iff for all $z, z' \in S$ for all types $\boldsymbol{\mathcal{S}} \in \mathfrak{S}$ we have $zxz' \in \boldsymbol{\mathcal{S}} \Leftrightarrow zyz' \in \boldsymbol{\mathcal{S}}$. If we view a language as a typed monoid with two nontrivial types, the definition above coincides with the definition of the syntactic congruence. It is easy to verify that the relation above forms a type preserving congruence. This allows us to define the minimal reduced monoid:

Definition 8 (Minimal Reduced Monoid). *Given a typed monoid* $(S, \mathfrak{S}, \mathcal{E})$, the minimal reduced monoid *is defined by* $\widetilde{(S, \mathfrak{S}, \mathcal{E})} = (S, \mathfrak{S}, \mathcal{E})/ \equiv_{(S,\mathfrak{S},\mathcal{E})}$.

Lemma 6. *Let* $(S, \mathfrak{S}, \mathcal{E})$ *be a typed monoid and* $(T, \mathfrak{T}, \mathcal{F})$ *be a reduced monoid of* $(S, \mathfrak{S}, \mathcal{E})$ *then* $\widetilde{(S, \mathfrak{S}, \mathcal{E})}$ *is a reduced monoid of* $(T, \mathfrak{T}, \mathcal{F})$.

Proof. Let $h : (S, \mathfrak{S}, \mathcal{E}) \to (T, \mathfrak{T}, \mathcal{F})$ be a surjective morphism. Since $h_S(s_1) = h_S(s_2)$ implies that $s_1 \equiv_{(S,\mathfrak{S},\mathcal{E})} s_2$ the mapping $T \to \widetilde{(S, \mathfrak{S}, \mathcal{E})}$ which maps every t to the congruence class of an inverse image of t is well defined and gives the desired surjective morphism. □

A standard operation on monoids is the direct product which corresponds to the Boolean closure on the language side. We will get the same equivalence in the typed world. The definition of the direct product of two typed monoids is straightforward and sound in the category theory sense.

Definition 9 (Direct Product). *The* direct product *of two monoids* $(S, \mathfrak{S}, \mathcal{E})$, $(S', \mathfrak{S}', \mathcal{E}')$, *denoted by* $(S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}')$, *is defined as* $(S \times S', \mathfrak{S} \times \mathfrak{S}', \mathcal{E} \times \mathcal{E}')$.

The direct product for typed monoids can express Boolean operations on the language side as in the case of conventional monoids:

Lemma 7. *Let* $L_1, L_2 \subseteq \Sigma^*$ *be languages recognized by typed monoids* $(S, \mathfrak{S}, \mathcal{E})$ *and* $(S', \mathfrak{S}', \mathcal{E}')$ *respectively. Then* $L_1 \cap L_2$ *and* $L_1 \cup L_2$ *are recognized by* $(S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}')$.

Motivated by the definition of a weakly closed class of transformation semigroups ([7, Chapter III]) we define:

Definition 10 (Weakly closed class of languages). *A* weakly closed class *of languages is a function* $\mathcal{V}$ *which associates to each alphabet* A *a nonempty set* $A^*\mathcal{V}$ *of languages over* A *such that*

1. $A^*\mathcal{V}$ *is closed under Boolean combinations, and*
2. $\varphi^{-1}(L) \in B^*\mathcal{V}$ *for every* $L \in A^*\mathcal{V}$ *and every length preserving morphisms* $\varphi : B^* \to A^*$.

Many sets of languages defined in other contexts, e.g. descriptive complexity or circuit complexity, do not form varieties (see Section 6) but weakly closed classes.

Example 5. The languages described by the logic class FO$[<, \mathrm{mod}]$ form a weakly closed class. One can easily verify that $\mathcal{L}(\mathrm{FO}[<, \mathrm{mod}])$ is closed under length preserving morphisms and it is closed under Boolean operations. But since FO$[<, \mathrm{mod}]$ cannot recognize the language L_{parity}, it is not closed under non length preserving morphisms.

We will now define sets of typed monoids:

Definition 11 (Weakly Closed Class). *We define a* weakly closed class *of typed monoids as a nonempty set of typed monoids that is closed under trivial extensions, division and finite direct products.*

Given a nonempty set $\mathbf{V}$ of typed monoids, we let $\mathcal{L}(\mathbf{V})$ be a mapping which associates with every alphabet A the nonempty set of all languages over A that can be recognized by a typed monoid of $\mathbf{V}$.

Obviously we have

Lemma 8. *Let $\mathbf{V}$, $\mathbf{W}$ be sets of typed monoids.*

(a) If $\mathbf{V}$ is closed under division, then $A^*\mathcal{L}(\mathbf{V})$ is the set of all languages $L \subseteq A^*$ with $syn(L) \in \mathbf{V}$.
(b) For two classes $\mathbf{V} \subseteq \mathbf{W}$ we have $A^*\mathcal{L}(\mathbf{V}) \subseteq A^*\mathcal{L}(\mathbf{W})$ for every alphabet A.

Our aim is a correspondence between weakly closed classes of languages and weakly closed classes of typed monoids, where the correspondence is given by the function $\mathcal{L}$.

Proposition 1. *If $\mathbf{V}$ is a weakly closed class of typed monoids, then $\mathcal{L}(\mathbf{V})$ is a weakly closed class of languages.*

Proof. We have to show that $\mathcal{V} = \mathcal{L}(\mathbf{V})$ forms a weakly closed class of languages. We first show $\mathcal{V}$ to fulfill the closure under inverse length preserving morphisms: let $L \subseteq \Sigma^*$ be a language in $\Sigma^*\mathcal{V}$ and $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid recognizing L via the typed morphism h. Assume that $L' \subseteq \Pi^*$ is a language such that $L' = \varphi^{-1}(L)$ where $\varphi : \Pi^* \to \Sigma^*$ is a length preserving morphism. Since φ is length preserving it can be seen as typed morphism from (Π^*, L', Π) to (Σ^*, L, Σ), thus $h \circ \varphi$ is a typed morphism from (Π^*, L', Π) to $(S, \mathfrak{S}, \mathcal{E})$, and therefore $L' \in \Pi^*\mathcal{V}$.

The other closure properties follow with Lemma 7. □

The next proposition ensures that every weakly closed class of languages can be characterized by a weakly closed class of typed monoids.

Proposition 2. *If $\mathcal{V}$ is a weakly closed class of languages, then there is a weakly closed class of typed monoids $\mathbf{V}$ with $\mathcal{L}(\mathbf{V}) = \mathcal{V}$.*

Proof. Let $\mathbf{V}$ be the smallest weakly closed class that contains all syntactic monoids of $\mathcal{V}$. We have to show that $\mathcal{L}(\mathbf{V}) \subseteq \mathcal{V}$, i.e. if $L \in \Sigma^*\mathcal{L}(\mathbf{V})$ then $L \in \Sigma^*\mathcal{V}$. The other inclusion is obvious.

The outline of the proof is as follows: we start with a language $L \in \Sigma^*\mathcal{L}(\mathbf{V})$ and want to show that it is also in $\Sigma^*\mathcal{V}$. We do this by constructing a language $L' \in \Pi^*$ as a Boolean combination of languages $L_i \in \Sigma_i^*\mathcal{V}$, where the L_i raise immediately from the typed monoid recognizing L, and constructing a length preserving morphism $\varphi : \Sigma^* \to \Pi^*$, such that $L = \varphi^{-1}(L')$.

L is recognized by a monoid in $\mathbf{V}$. We may assume that L is recognized via a typed morphism $h : (\Sigma^*, L, \Sigma) \to \bigtimes_{i=1}^{n}(S_i, \mathfrak{S}_i, \mathcal{E}_i)$, where $(S_i, \mathfrak{S}_i, \mathcal{E}_i)$ are syntactic monoids of some languages $L_i \in \Sigma_i^*\mathcal{V}$, in particular $(S_i, \mathfrak{S}_i, \mathcal{E}_i) \in \mathbf{V}$ (we can ignore the closure under division and trivial extension by Lemma 5). Further, the languages $L_i \subseteq \Sigma_i^*$ are recognized via surjective morphisms $\eta_i : (\Sigma_i^*, L_i, \Sigma_i) \to (S_i, \mathfrak{S}_i, \mathcal{E}_i)$.

We now construct L' and φ. The following diagram depicts the situation:

$$\begin{tikzcd}
(\Sigma^*, L, \Sigma) \arrow[r, "h"] \arrow[dr, dotted, "\tilde{h}"] & \bigtimes_{i=1}^{n}(S_i, \mathfrak{S}_i, \mathcal{E}_i) \in \mathbf{V} \\
& \bigtimes_{i=1}^{n}(\Sigma_i^*, L_i, \Sigma_i) \in \mathbf{V} \arrow[u, two heads]
\end{tikzcd}$$

The typed monoids $(\Sigma_i^*, L_i, \Sigma_i)$ are trivial extensions of $(S_i, \mathfrak{S}_i, \mathcal{E}_i)$ and therefore there exists a typed morphism $\tilde{h} : (\Sigma^*, L, \Sigma) \to \bigtimes_{i=1}^{n}(\Sigma_i^*, L_i, \Sigma_i)$. So $L = \tilde{h}^{-1}(\boldsymbol{\mathcal{S}})$ for some type $\boldsymbol{\mathcal{S}} = \bigtimes_{i=1}^{n} \boldsymbol{\mathcal{S}}_i$, where $\boldsymbol{\mathcal{S}}_i \in \{\emptyset, \Sigma_i^*, L_i, \Sigma_i^* \setminus L_i\}$. Note that $\tilde{h}_{\Sigma^*}(\Sigma^*) \subseteq (\bigtimes_{i=1}^{n} \Sigma_i)^*$ (since $\tilde{h}_{\Sigma}(\Sigma) \subseteq \bigtimes_{i=1}^{n} \Sigma_i$ and by the compatibility conditions of typed morphisms), thus $\tilde{h}_{\Sigma^*} : \Sigma^* \to (\bigtimes_{i=1}^{n} \Sigma_i)^*$ is a length preserving morphism. The assertion follows by setting $\Pi = (\bigtimes_{i=1}^{n} \Sigma_i)$, $L' = \boldsymbol{\mathcal{S}}$ and $\varphi = \tilde{h}_{\Sigma^*}$. $\square$

Thus, by Proposition 1 and Proposition 2 we get:

Theorem 1. *Let* $\mathbf{V}$ *be a weakly closed class of typed monoids, then* $\mathcal{L}(\mathbf{V})$ *is a weakly closed class of languages.*

Moreover, if $\mathcal{V}$ *is a weakly closed class of languages, then there is a weakly closed class* $\mathbf{V}$ *of typed monoids such that* $\mathcal{L}(\mathbf{V}) = \mathcal{V}$*.*

Note that this theorem does not guarantee a 1-1 correspondence: for a given weakly closed class of languages, there could be multiple weakly closed classes of typed monoids.

6 Varieties

In this section we prove our analogon to Eilenberg's theorem. Our notion of a language variety is the same that is found as $*$-variety in the literature [7,15].

The right quotient of a language $L \subseteq \Sigma^*$ by $w \in \Sigma^*$ is defined by $Lw^{-1} = \{x \in \Sigma^* \mid xw \in L\}$. The left quotient $w^{-1}L$ is defined analogously. A *variety* of languages is a weakly closed class $\mathcal{V}$ of languages such that for all alphabets A and B holds:

1. If $L \in A^*\mathcal{V}$, then $a^{-1}L, La^{-1} \in A^*\mathcal{V}$ for all $a \in A$ and $\varphi^{-1}(L) \in B^*\mathcal{V}$ where $\varphi : B^* \to A^*$ is a morphism.

To adapt this concept to the theory of typed monoids, we need the notion of shifts and unit relaxations. The following definition models the fact that varieties are closed under left and right quotients.

Definition 12 (Shifting). *Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid. Then $(S, \mathfrak{S}', \mathcal{E})$ is a* shift *of $(S, \mathfrak{S}, \mathcal{E})$, if there are $\lambda, \varrho \in S$ with $\mathfrak{S}' = \{\lambda^{-1}\boldsymbol{\mathcal{S}}\varrho^{-1} \mid \boldsymbol{\mathcal{S}} \in \mathfrak{S}\}$, where $\{\lambda^{-1}\boldsymbol{\mathcal{S}}\varrho^{-1}\}$=$\{s \in S \mid \lambda s \varrho \in \boldsymbol{\mathcal{S}}\}$.*

The use of units let typed morphism correspond to length preserving morphisms for languages. In order to a notion for non-length-preserving morphism we allow arbitrary finite subsets as units:

Definition 13 (Unit Relaxation). *Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid. Then for any finite set $\mathcal{E}' \subseteq S$ we say $(S, \mathfrak{S}, \mathcal{E}')$ is a* unit relaxation *of $(S, \mathfrak{S}, \mathcal{E})$.*

Adding these two closure properties to the requirement of a weakly closed class of typed monoids we obtain a variety of typed monoids.

Definition 14 (Variety of Typed Monoids). *A* variety *of typed monoids is a weakly closed class that is closed under shifting and unit relaxation.*

Note that a variety of finite monoids in the sense of [15] does not form a variety of typed monoids if we consider every finite monoid S as the typed monoid $(S, \mathcal{P}(S), S)$, since it is not closed under trivial extension leading to infinite typed monoids.

Proposition 3. *If $\mathbf{V}$ is a variety of typed monoids, then $\mathcal{L}(\mathbf{V})$ is a variety of languages.*

Proof. Let $\mathcal{V} = \mathcal{L}(\mathbf{V})$. By Proposition 1 it remains to show that $\mathcal{V}$ is closed under quotients and under inverse morphism. The closure under quotients is obviously given, since $\mathbf{V}$ is closed under shifting.

Assume that $\varphi : \Pi^* \to \Sigma^*$ is a morphism and $L \subseteq \Sigma^*$ is a language recognized by a typed monoid $(S, \mathfrak{S}, \mathcal{E}) \in \mathbf{V}$ (thus there is a typed morphism $h : (\Sigma^*, L, \Sigma) \to (S, \mathfrak{S}, \mathcal{E})$). We need to show that $L' = \varphi^{-1}(L)$ is also recognized by a monoid in $\mathbf{V}$. But this follows by unit relaxation, since we can consider φ as a typed morphism from (Π^*, L', Π) to $(\Sigma^*, L, \varphi(\Pi))$ and thus $h \circ \varphi : (\Pi^*, L', \Pi) \to (S, \mathfrak{S}, h(\varphi(\Pi)))$ is a typed morphism to a monoid in $\mathbf{V}$. □

Proposition 4. *For two varieties $\mathbf{V} \subseteq \mathbf{W}$ we have $\mathcal{L}(\mathbf{V}) \subseteq \mathcal{L}(\mathbf{W})$, where equality occurs only if $\mathbf{V} = \mathbf{W}$.*

Proof. Let $\mathcal{L}(\mathbf{V}) = \mathcal{V}$ and $\mathcal{L}(\mathbf{W}) = \mathcal{W}$. By Lemma 8 we have $\mathcal{V} \subseteq \mathcal{W}$, so we need to show the equality statement for varieties.

Let $(S, \mathfrak{S}, \mathcal{E})$ be a monoid in $\mathbf{W}$ and assume $\mathcal{V} = \mathcal{W}$, we show $(S, \mathfrak{S}, \mathcal{E})$ is in $\mathbf{V}$. We denote the types in $\mathfrak{S}$ by $\boldsymbol{\mathcal{S}}_i$. Let G be a generating set of S containing $\mathcal{E}$, then $(S, \boldsymbol{\mathcal{S}}_i, G)$ is a typed monoid in $\mathbf{W}$ for all i. The set G generates S, thus there is a language $L_i \subseteq G^*$ recognized by $(S, \boldsymbol{\mathcal{S}}_i, G)$, moreover $\mathrm{syn}(L_i) \cong \widetilde{(S, \boldsymbol{\mathcal{S}}_i, G)} \in \mathbf{W}$. Thus L_i is in $G^*\mathcal{W} = G^*\mathcal{V}$ and consequently $\widetilde{(S, \boldsymbol{\mathcal{S}}_i, G)}$ and therefore $(S, \boldsymbol{\mathcal{S}}_i, G) \in \mathbf{V}$ for all i. Since $(S, \mathfrak{S}, \mathcal{E})$ divides $\times_i (S, \boldsymbol{\mathcal{S}}_i, G)$ we conclude $(S, \mathfrak{S}, \mathcal{E}) \in \mathbf{V}$ and hence $\mathbf{V} = \mathbf{W}$. □

Proposition 5. *For every variety of languages $\mathcal{V}$ there is a corresponding variety of typed monoids* $\mathbf{V}$*, such that* $\mathcal{V} = \mathcal{L}(\mathbf{V})$*.*

Proof. The proof is similar to the proof of Proposition 2. We let $\mathbf{V}$ be the smallest variety that contains all syntactic monoids of $\mathcal{V}$. We need to show that $L \in \Sigma^*\mathcal{L}(\mathbf{V})$ implies $L \in \Sigma^*\mathcal{V}$. We construct a language $L' \in \Pi^*\mathcal{V}$ and a morphism $\varphi : \Sigma^* \to \Pi^*$ such that $L = \varphi^{-1}(L')$.

L is recognized by a monoid in $\mathbf{V}$. We may assume that L is recognized by $\times(S_i, \mathfrak{S}_i, \widetilde{\mathcal{E}_i})$ via some morphism $h : (\Sigma^*, L, \Sigma) \to \times(S_i, \mathfrak{S}_i, \widetilde{\mathcal{E}_i})$, where $(S_i, \mathfrak{S}_i, \mathcal{E}_i)$ are syntactic monoids of some languages $L_i \in \Sigma_i^*\mathcal{V}$ and $\widetilde{\mathcal{E}_i}$ are arbitrary finite subsets of S_i. Further, the languages $L_i \subseteq \Sigma_i^*$ are recognized via surjective morphisms $\eta_i : (\Sigma_i^*, L_i, \Sigma_i) \to (S_i, \mathfrak{S}_i, \mathcal{E}_i)$. Note that we may ignore the closure under shifting since $\mathcal{V}$ is closed under quotients.

The typed monoids $(\Sigma_i^*, L_i, \eta_i^{-1}(\widetilde{\mathcal{E}_i}))$ are trivial extensions of $(S_i, \mathfrak{S}_i, \widetilde{\mathcal{E}_i})$ and therefore exists a typed morphism $\tilde{h} : (\Sigma^*, L, \Sigma) \to \times_{i=1}^{n}(\Sigma_i^*, L_i, \eta_i^{-1}(\widetilde{\mathcal{E}_i}))$. So $L = \tilde{h}^{-1}(\boldsymbol{\mathcal{S}})$ for some type $\boldsymbol{\mathcal{S}} = \times_{i=1}^{n}\boldsymbol{\mathcal{S}}_i$, where $\boldsymbol{\mathcal{S}}_i \in \{\emptyset, \Sigma_i^*, L_i, \Sigma_i^* \setminus L_i\}$.

In contrast to the proof of Proposition 2 we can not immediately conclude that $\tilde{h}_{\Sigma^*}(\Sigma^*) \subseteq (\times\, \Sigma_i)^*$, since every entry in a tuple $\tilde{h}_{\Sigma^*}(w) \in (\times\, \Sigma_i^*)$ could have different length. Nevertheless, since we have the closure under unit relaxation we may assume that every language L_i has a neutral letter, i.e. a letter that gets mapped to the neutral elements by the morphism η_i. Thus we can identify $\tilde{h}_{\Sigma^*}(w) = (w_1, \ldots w_n)$ with a word $(v_1, \ldots, v_n) \in (\times\, \Sigma_i)^*$ where v_i is w_i padded with the neutral letter.

Now the assertion follows with φ is the morphism induced by $\tilde{h}_{\Sigma^*}$, $\Pi = (\times_{i=1}^{n}\Sigma_i)$ and $L' = \boldsymbol{\mathcal{S}}$. □

Summing up these results we obtain a one-to-one correspondence for varieties as in the finite case:

Theorem 2. *Varieties of typed monoids and varieties of languages are in a one-to-one correspondence:*

– *Let $\mathcal{V}$ be a variety of languages and* $\mathbf{V}$ *the smallest variety of typed monoids that recognizes all languages in $\mathcal{V}$, then* $\mathcal{L}(\mathbf{V}) = \mathcal{V}$*.*
– *Let* $\mathbf{V}$ *be a variety of typed monoids and* $\mathbf{W}$ *be the smallest variety that recognizes all languages of* $\mathcal{L}(\mathbf{V})$*, then* $\mathbf{V} = \mathbf{W}$*.*

As shown above, a variety of typed monoids always contains infinite typed monoids because of the closure under trivial extensions. But without this closure property there would be no one-to-one correspondence in the previous theorem. It is easy to construct a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ such that any language recognized by $(S, \mathfrak{S}, \mathcal{E})$ is aperiodic but S contains a group.

7 Discussion

We have introduced typed monoids and shown that they can be used to describe languages, weakly closed classes, and varieties of languages. Typed monoids allow us to obtain a more precise description of language classes than with the usual approach with monoids or as mg-pairs or stamps as in [8] and [17].

In this paper we presented basic results about typed monoids. We have shown the existence and uniqueness of a typed syntactic monoid and the equivalent of Eilenberg's theorem for typed monoids as well as a weaker version for weakly closed classes. A number of questions are open when studying typed monoids. Green's relation are a useful tool in the study of monoids. Many important varieties within the regular languages can be defined via some properties of the Green's relations of the monoids involved. This opens the question whether there is a useful notion of similar relations for typed monoids.

It usually makes a difference if one considers language recognition via monoids or semigroups. In the latter case languages are subsets of Σ^+ instead of Σ^* and the term +-varieties is used. The difference between these two approaches diminishes for typed monoids. The empty word must be mapped onto the neutral element of the typed monoid. But if the neutral element is not contained in the units, the morphism on $\Sigma^* \setminus \{\lambda\} = \Sigma^+$ behaves like a semigroup morphism. Conversely, if one considers a language $L \subseteq \Sigma^+$ recognized by a semigroup S with accepting set A, this language is recognized by the typed monoid $(S^1, A, \eta_L(\Sigma))$. Here, S^1 denotes S with an additional neutral element added. If for example a language L without a neutral letter is recognized by a semigroup S then $(S^1, A, \eta_L(\Sigma))$ recognizes L but not L with an additional neutral letter.

The motivation to study typed monoids stems from an interest in algebraic characterizations of logic and circuit classes. The program linking finite monoids [10,11,5,4,3] to first order logic relied heavily on the block product [19] (or the bilateral semidirect product). It is possible to define the block product for typed monoids and, for example, obtain characterizations for first order logic with the majority quantifier [13] as well as subclasses [6]. To do so it was essential to limit the set of possible accepting subsets. The limitation of the acceptance sets (types) was used in [20] to obtain monoids for context free languages. By using a Boolean algebra we loose the ability to characterize language classes that are not closed under Boolean operations. We choose a Boolean algebra because circuit classes are usually closed under Boolean operations. Even more, the use of a Boolean algebra lead to a clean definition of a block product (or bilateral semidirect product) and eases the characterizations of logic classes with (arbitrary) Lindström quantifiers.

The approach of ordered monoids ([16]) allows to examine language classes not closed under complement. A possible research area is to introduce "positive" and "negative" types, closed under union to characterize context free languages and one might be even able to define a block product for these kind of objects.

Acknowledgments. The authors want to thank Pierre McKenzie for his help to improve the paper and the anonymous referees for their useful comments on the paper.

References

1. Ajtai, M.: First-order definability on finite structures. Ann. Pure Appl. Logic 45(3), 211–225 (1989)
2. Almeida, J.: Finite Semigroups and Universal Algebra. World Scientific, Singapore (1995)
3. Barrington, D.A.M., Compton, K.J., Straubing, H., Thérien, D.: Regular languages in NC^1. J. Comput. Syst. Sci. 44(3), 478–499 (1992)
4. Barrington, D.A.M., Immerman, N., Straubing, H.: On Uniformity within NC^1. J. Comput. Syst. Sci. 41(3), 274–306 (1990)
5. Barrington, D.A.M., Straubing, H., Thérien, D.: Non-uniform automata over groups. Inf. Comput. 89(2), 109–132 (1990)
6. Behle, C., Krebs, A., Mercer, M.: Linear circuits, two-variable logic and weakly blocked monoids. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 147–158. Springer, Heidelberg (2007)
7. Eilenberg, S.: Automata, Languages and Machines, vol. A+B. Academic Press, London (1976)
8. Ésik, Z., Larsen, K.G.: Regular languages definable by Lindström quantifiers. ITA 37(3), 179–241 (2003)
9. Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. In: FOCS, pp. 260–270 (1981)
10. Gurevich, Y., Lewis, H.R.: A logic for constant-depth circuits. Information and Control 61(1), 65–74 (1984)
11. Immerman, N.: Languages that capture complexity classes. SIAM J. Comput. 16(4), 760–778 (1987)
12. Krebs, A.: Typed Semigroups, Majority Logic, and Threshold Circuits. Ph.D. thesis, Universität Tübingen (2008)
13. Krebs, A., Lange, K.J., Reifferscheid, S.: Characterizing TC^0 in terms of infinite groups. Theory Comput. Syst. 40(4), 303–325 (2007)
14. McNaughton, R., Papert, S.: Counter-free automata. With an appendix by William Henneman. Research Monograph, vol. 65, XIX, 163 p. The M.I.T. Press, Cambridge (1971)
15. Pin, J.E.: Varieties of formal languages. Plenum, London (1986)
16. Pin, J.E.: A variety theorem without complementation. Izvestiya VUZ Matematika 39, 80–90 (1995); english version: Russian Mathem. (Iz. VUZ) 39, 74–83 (1995)
17. Pin, J.E., Straubing, H.: Some results on C-varieties. ITA 39(1), 239–262 (2005)
18. Rhodes, J., Weil, P.: Algebraic and topological theory of languages. ITA 29(1), 1–44 (1995)
19. Rhodes, J.L., Tilson, B.: The kernel of monoid morphisms. J. Pure Applied Alg. 62, 27–268 (1989)
20. Sakarovitch, J.: An algebraic framework for the study of the syntactic monoids application to the group languages. In: Mazurkiewicz, A. (ed.) MFCS 1976. LNCS, vol. 45, pp. 510–516. Springer, Heidelberg (1976)
21. Schützenberger, M.P.: On finite monoids having only trivial subgroups. Information and Control 8(2), 190–194 (1965)
22. Straubing, H.: Finite Automata, Formal Logic, and Circuit Complexity. Birkhäuser, Boston (1994)
23. Tesson, P., Thérien, D.: Logic meets algebra: the case of regular languages. Logical Methods in Computer Science 3(1) (2007)

Codes and Combinatorial Structures from Circular Planar Nearrings

Anna Benini[1], Achille Frigeri[2], and Fiorenza Morini[3]

[1] Università di Brescia
anna.benini@ing.unibs.it
[2] Politecnico di Milano
achille.frigeri@polimi.it
[3] Università di Parma
fiorenza.morini@unipr.it

Abstract. Nearrings are generalized rings in which addition is not in general abelian and only one distributive law holds. Some interesting combinatorial structures, as tactical configurations and balanced incomplete block designs (BIBDs) naturally arise when considering the class of planar and circular nearrings. In [12] the authors define the concept of disk and prove that in the case of field-generated planar circular nearrings it yields a BIBD, called *disk-design*. In this paper we present a method for the construction of an association scheme which makes the disk-design, in some interesting cases, an union of partially incomplete block designs (PBIBDs). Such designs can be used in the construction of some classes of codes for which we are able to calculate the parameters and to prove that in some cases they are also cyclic.

Keywords: Binary codes, Planar circular nearring, PBIB design.

1 Introduction

A *binary* (n, m, d)*-code* is a subset $\mathcal{C}$ of $\mathbb{Z}_2^n$ such that $|\mathcal{C}| = m$ and the number of coordinates in which any two elements of $\mathcal{C}$, called *codewords*, differ is at least d. A code which is also a vector subspace of $\mathbb{Z}_2^n$ is a *linear* code. Binary codes are widely used in information theory, as they allow the correction of errors in the delivery of digital data over unreliable communication channels. Moreover (error-correcting) codes are strongly related with other areas of theoretical computer science. They are used in [15] to show an example of a general provably secure steganographic protocol, while in [18] it is shown that the **IP** theorem, stating that any set in **PSPACE** has an interactive proof, can be proved by using (linear) error-correcting codes.

The construction of codes from planar nearrings was at first pointed out in [14], where the codes arising from *balancing incomplete block designs* (BIBDs) constructed from planar nearrings are investigated. These codes turn out to be "good" w.r.t. several properties, although it is well known that there cannot exist a universal criterion that decides the efficiency of a code. Such "nearring codes" were further investigated in [10] and decoding methods are presented in [13]. Moreover, with the implementation of the GAP package SONATA [2,1] which provides methods for the construction and the analysis of finite nearrings, algorithms for coding/decoding of such codes can be effectively realized.

A left (resp. right) nearring is an algebraic structure $(N, +, \cdot)$ extending the concept of ring. In particular the addition is not necessarily abelian and only the left (resp. right) distributive law holds. A standard example is the collection of all mappings from a group into itself w.r.t. addition and composition of functions, which gives a right nearring. In a left nearring one can define "line" by means of the equation $y = a \cdot x + c$. A nearring is *planar* if any two non-parallel lines has exactly one intersection and if there exists at least three non-equivalent slopes (note that lines $y = a \cdot x$ and $y = b \cdot x$ can be equal also if $a \neq b$, Definition 2). A well-known example for a planar nearring is a normed vector space V over $\mathbb{R}$ with the multiplication $x \star y = \|x\|y$. Then two slopes a and b are equivalent if and only if $\|a\| = \|b\|$. Thus, $(V, +, \star)$ is a planar nearring which is not a ring. Circular planar nearrings are special planar nearrings in which any two distinct sets of the form $N^*a + b$, called *circles*, intersect in at most two points. For example, $(\mathbb{C}, +, \circ)$, where $a \circ b = \frac{a}{|a|} \cdot b$ if $a \neq 0$ and 0 otherwise, is a planar circular nearring. While incidence structures that arise considering lines and circles have been widely studied, little is known about the concept of *disk*, i.e., the union of the circle and its interior part. Various definitions of disk have been proposed in [6,7] and in [11] the author proves some properties of this kind of disks, but no special combinatorial structure seems to arise from this approach. Recently, in [12], a new and more natural definition was proposed from which a new class of BIBDs can be constructed.

2 Preliminaries and Notations

2.1 Circular Planar Nearrings

Definition 1. *A* (left) nearring *is an algebraic structure* $(N, +, \cdot)$ *on a nonempty set* N *with two inner operations,* $+$ *and* $\cdot$*, such that:*

1. $(N, +)$ *is a group;*
2. $(N, \cdot)$ *is a semigroup;*
3. *the left distributive law holds, i.e.,*

$$\forall x, y, z \in N, \quad x \cdot (y + z) = (x \cdot y) + (x \cdot z).$$

Moreover, if $(N \setminus \{0\}, \cdot)$ *is a group, then* $(N, +, \cdot)$ *is a* (left) nearfield.

We remind now some definitions and simple properties of nearrings, for details see [6].

Let $(N, +, \cdot)$ be a nearring, then for any $x, y \in N$, $x \cdot (-y) = -(x \cdot y)$ and $x \cdot 0 = 0$; moreover if for any $x \in N$ we have $x \cdot 0 = 0 \cdot x = 0$, N is said to be *0-symmetric.*

We say that $a, b \in N$ are *equivalent multipliers* if and only if for all $n \in N$, $a \cdot n = b \cdot n$. It is easy to see that to be equivalent multipliers is an equivalence relation and we denote it by $\equiv_m$. We have the following fundamental definition.

Definition 2. *A nearring* $(N, +, \cdot)$ *is said to be* planar *if:*

1. $|N/\equiv_m| \geq 3$*;*
2. $\forall a, b, c \in N$*, with* $a \not\equiv_m b$*, the equation*
$$a \cdot x = b \cdot x + c$$
has a unique solution in N*.*

For planar nearrings we consider the set $A = \{n \in N \mid n \equiv_m \ 0 \}$, called *annihilator* of N, and we write

$$N^0 = N \setminus \{0\} \quad \text{and} \quad N^* = N \setminus A.$$

It is well known that planar nearrings are 0-symmetric; moreover planar nearrings with identity are also planar nearfield. Vice-versa a finite nearfield with at least three elements is planar and 0-symmetric and its additive group is the additive group of a field.

Definition 3. *Let* $(N, +)$ *be a group. A subgroup of automorphisms* Φ*,* $\{1\} \neq \Phi < \mathrm{Aut}N$*, is said to be* regular *if for any* $\varphi \in \Phi \setminus \{1\}$*,* φ *is a fixed point free (f.p.f.) automorphism, i.e.,* $\varphi(x) = x \Leftrightarrow x = 0$*. Moreover, if for any* $\varphi \in \Phi \setminus \{1\}$*,* $-\varphi + 1$ *is surjective, the pair* (N, Φ) *is called a* Ferrero pair*. Then an* orbit *of* Φ *is the set*
$$\Phi(a) = \{\varphi(a) \mid \varphi \in \Phi\},$$
for some $a \in N$ *(*$\Phi(0) = \{0\}$ *is the trivial orbit).*

The concept of Ferrero pair is central in this framework since every planar nearring can be constructed from such pair by the so called *Ferrero Planar Nearring Factory* (for details, see [6],§4.1). Moreover we remember that, even if non-isomorphic nearrings can be generated by the same pair, the knowledge of the generating pair suffices in the study of the geometrical properties of the nearring.

Theorem 1 (Ferrero, Clay). *Let* (N, Φ) *be a finite Ferrero pair. Then*

1. $\forall a \in N^0,\ |\Phi(a)| = |\Phi|$*;*
2. $\forall a \in N,\ \forall b \in \Phi(a),\ \Phi(a) = \Phi(b)$*;*
3. $\{\Phi(a) \mid a \in N\}$ *is a partition of* N*;*
4. $|\Phi|$ *divides* $|N| - 1$*.*

Definition 4. *A planar nearring is said to be* field-generated *if it is generated from a Ferrero pair* (N, Φ)*, where* $(N, +, \cdot)$ *is a field and* Φ *is isomorphic to a subgroup of* $(N^0, \cdot)$*.*

It follows immediately that a field-generated planar nearring has a very simple structure, in fact every non trivial orbit is isomorphic to a multiplicative subgroup of a field.

Now, on a planar nearring $(N, +, \cdot)$ we consider the incidence structure $(N, \mathcal{B}^*)$ where $\mathcal{B}^* \subset \mathcal{P}(N)$ is defined by

$$\mathcal{B}^* = \{N^* \cdot a + b \mid a, b \in N,\ a \neq 0\}.$$

As usual in this context, we call the elements of N points and those of $\mathcal{B}^*$ blocks. It is immediate to observe that the structure $(N, \mathcal{B}^*)$ depends only on the pair (N, Φ) in the sense that if $(N, +, *)$ and $(N, +, \circ)$ are two planar nearrings constructed from the same Ferrero pair, then they yield identical $(N, \mathcal{B}^*)$; it makes no difference to assume that the nearring is integral planar. This allows the following definition.

Definition 5. *Let $(N, +, \cdot)$ be a finite planar nearring generated by the Ferrero pair (N, Φ) and let $(N, \mathcal{B}^*)$ be the incidence structure with $\mathcal{B}^* = \{N^* \cdot a + b \mid a, b \in N,\ a \neq 0\}$. Then $(N, +, \cdot)$ is said* circular *if every three distinct points of N belong to at most one block of $\mathcal{B}^*$ and if every two distinct points belong to at least two distinct blocks. In this case, also the incidence structure $(N, \mathcal{B}^*)$ and the pair (N, Φ) are called circular and the block $N^* \cdot a + b = \Phi(a) + b$ is called* circle *with* center *b and* radius *a (see [6],§5.1).*

2.2 PBIBDs and Association Scheme

Definition 6. *Let $(X, \mathcal{B})$ be an incidence structure with $|X| = v$ and $|\mathcal{B}| = b$. The pair $(X, \mathcal{B})$ is called* tactical configuration *with parameters v, b, k, r if for any $B \in \mathcal{B}$, $|B| = k$, and every $x \in X$ belongs to exactly r distinct blocks $B_1, ..., B_r \in \mathcal{B}$. Moreover, if an integer λ exists so that every pair of points belongs to exactly λ distinct blocks, then $(X, \mathcal{B})$ is called* balanced incomplete block design (BIBD) *with parameters v, b, k, r, λ.*

Proposition 1 ([6],§5). *Let $(X, \mathcal{B})$ be a BIBD with parameters v, b, k, r, λ. Then $vr = bk$ and $\lambda(v-1) = r(k-1)$.*

If $(N, +, \cdot)$ is a finite planar nearring, then $(N, \mathcal{B}^*)$ is a BIBD with parameters $v = |N|$, $k = |N^*/\equiv_m|$, $b = v(v-1)/k$, $r = v-1$, $\lambda = k-1$. Moreover if the BIBD is circular, then $k \leq (3 + \sqrt{4v-7})/2$ and this limit is effective.

For more information on design theory we refer to [5].

Definition 7. *An* association scheme with m associate classes *on a finite set X is a family of m symmetric and antireflexive binary relations $R_1, \ldots, R_m$ on X such that:*

1. *any two distinct elements of X are ith associates for exactly one value of $i = 1, \ldots, m$;*
2. *for all $i = 1, \ldots, m$ and $x \in X$, there are exactly n_i distinct elements $y \in X$ so that $(x, y) \in R_i$;*
3. *for all $i, j, k = 1, \ldots, m$, if $(x, y) \in R_k$, the number p_{ij}^k of $z \in X$ so that $(x, z) \in R_i$ and $(y, z) \in R_j$ is a constant depending on i, j, k but not on the particular choice of x and y.*

For further information about association schemes see [4].

Definition 8. *A tactical configuration* $(X, \mathcal{B})$ *with an association scheme on* X *is called* partially balanced incomplete block design *(PBIBD) if there are positive integers* λ_i, $i = 1, \ldots, m$, *such that, if* $x, y \in X$ *are any two ith associate elements, then* x, y *occur together in exactly* λ_i *blocks of* $\mathcal{B}$.

Thus a $PBIBD$ has the tactical configuration parameters v, b, r and k, the association scheme parameters n_i and p_{ij}^k, and the partial balance parameters λ_i, in addition. We refer to [19] for more information on $PBIBD$s.

3 Disks

In [12] the authors introduced the following definition.

Definition 9. *Let* (N, Φ) *be a circular Ferrero pair and* $\Phi(a) + b \in \mathcal{B}^*$, *then we define* $D(a; b)$, *the* disk of center b and radius a, *as*

$$D(a; b) = \{x \in \Phi(r) + c \mid r \neq 0,\ b \in \Phi(r) + c,\ |(\Phi(r) + c) \cap (\Phi(a) + b)| = 1\}.$$

The idea is to construct a disk of center b and radius a joining together all the circles tangent to the circle $\Phi(a) + b$ and containing its center.

In [12] the geometrical structure of the disks is studied and in particular is presented, at least in the more interesting cases, a very fast way to construct them and to check the membership of a point. Now we summarize some results proved in [12].

Lemma 1. *Let* (N, Φ) *be a Ferrero pair and* $a, b \in N$ *with* $a \neq 0$, *then*

$$D(a; b) = D(a; 0) + b$$

Lemma 2. *Let* (N, Φ) *be a field-generated Ferrero pair and let* $(N, +, \cdot)$ *be the generating field. If* $a \in N^0$, *then*

$$D(a; 0) = a \cdot D(1; 0)$$

By the two previous lemmas we immediately have the following corollary.

Corollary 1. *With the same hypothesis of the previous lemma, we have*

$$D(a; b) = a \cdot D(1; 0) + b = a \cdot D(1; a^{-1} \cdot b)$$

Theorem 2. *In the same hypothesis of the previous lemma, if* $c \in D(a; b)$, *then* $\Phi(c) + b \subseteq D(a; b)$. *More precisely, if* $|\Phi| = 2n$, *then every disk is union of* $n + 1$ *circles (one of which degenerates), that is there are* $c_1, ..., c_{n-1} \in N^0 \setminus \Phi(a)$, *with* $\Phi(c_i) \neq \Phi(c_j)$ *if* $i \neq j$, *so that*

$$D(a; b) = \{\Phi(0) \cup \Phi(c_1) \cup ... \cup \Phi(c_{n-1}) \cup \Phi(a)\} + b.$$

It follows that every disk has exactly $2n^2 + 1$ *points.*

In [12] the authors prove, in a special case, an important combinatorial property of the incidence structure obtained considering the sets of all disks, namely

$$\mathcal{B}^{\mathcal{D}} = \{D(a;b) \mid a,b \in N,\ a \neq 0\}$$

Theorem 3. *Let (N,Φ) be a finite circular field-generated Ferrero pair with $|N| = p$, p prime, and $|\Phi| = 2n$. Then $(N,\mathcal{B}^{\mathcal{D}})$ is a BIBD.*

Moreover, if $D(1;0)\backslash\{0\}$ is not a group, then $(N,\mathcal{B}^{\mathcal{D}})$ is a BIBD of parameters $v = p$, $b = p(p-1)/2n$, $k = 2n^2+1$, $r = (p-1)(2n^2+1)/2n$, $\lambda = n(2n^2+1)$.

3.1 Construction of PBIBDs

Now, in a special case, we describe the steps of the construction of an association scheme which makes $\mathcal{D}_a = (N, \mathcal{B}_a = \{D(a;b) \mid b \in N\})$, for $a \in N^0$, a PBIBD.

First step. Let now N be a circular field-generated integral nearring obtained by the Ferrero pair $(\mathbb{Z}_p,\Phi)$, where $\mathbb{Z}_p$ denote the finite field of order p prime and $|\Phi| = 2n < p-1$. For convenience, in what follows each element of $\mathbb{Z}_p$ will be denoted via his smallest non negative representative, that is $\mathbb{Z}_p = \{0,1,\ldots,p-1\}$. Moreover, we will denote by $D_{a,b}$ the disk $D(a;b) = D(a;0)+b$ defined in Def. 9. From [12] we know that $D_{a_1,0} = D_{a_2,0}$ if and only if a_1 and a_2 belong to the same Φ-orbit. Let $E = \{e_1,e_2,\ldots,e_m\}$ be a set of representatives of the non trivial Φ-orbits, so there are $m = (p-1)/2n$ different disks with center 0: those with radius $e_i \in E$. From Theorem 2 we learn that $D_{e_i,0}$ is an union of Φ-orbits and $D_{e_i,b_1} = D_{e_i,b_2}$ implies $b_1 = b_2$. So the development of $D_{e_i,0}$ in N, results in a tactical configuration whose blocks are of the form $D_{e_i,0}+b = D_{e_i,b}, b \in N$, for each $i = 1,\ldots,m$. From Theorem 3 we know that the union of previous designs, for $e_i \in E$, results in a tactical configuration, too.

Second step. Consider the orbit of e_i and denote it by Φ_i. Define x and y to be ith associates if $x-y$ belongs to Φ_i, that is the ith associates of y are the elements of the circle of center y and radius e_i. In this way *an association scheme is given on* $\mathbb{Z}_p$ and it fits in with all the previously constructed block designs. Precisely, the tactical configuration $\mathcal{D}_{e_i} = (N,\mathcal{B}_{e_i})$ with this association scheme results a PBIBD, called *orbital disk-design*. In this way we obtain $m = (p-1)/2n$ orbital disk-designs, isomorphic to each other. Moreover, the *union disk-design* $\mathcal{D} = \bigcup_{e_i \in E}\mathcal{D}_{e_i}$, where the blocks have the form $D_{e_i,b}$, with $e_i \in E$ and $b \in N$, turns out to be even a BIB-design (Theorem 3).

3.2 Notations and Parameters

To remember easily and quickly the main definitions and properties, we give in Table 1, 2 and 3 some useful keys about notation and parameters of designs.

Remark 1. It is easy to see that for any $h \in N$,

$$f_{a_1,a_2,h} = f_{a_1,a_2,\phi(h)}\ \forall\phi \in \Phi, \quad \forall a_1,a_2 \in N^0 \quad \text{and} \quad f_{a,h} = f_{1,ha^{-1}},\ \forall a \in N^0$$

Table 1. Notation for nearrings and disk-designs

$\varPhi$	automorphism group of $(\mathbb{Z}_p, +)$	$1 < \lvert\varPhi\rvert = 2n < p-1$ vspace-2mm
$N = (\mathbb{Z}_p, +, *)$	circular planar integral nearring	
m	number of the non trivial $\varPhi$-orbits	$m = (p-1)/2n$
$E = \{e_j\}$	representatives of the non trivial $\varPhi$-orbits	
$\varPhi_l = \varPhi(e_l)$	lth non trivial $\varPhi$-orbit with representatives e_l	$l = 1, \ldots, m$
$\varPhi(a) + b$	circle of center $b \in N$ and radius $a \in N^0$	
$D_{a,b}$	disk (block) of center $b \in N$ and radius $a \in N^0$	
$[D_{a_1,b_1} - D_{a_2,b_2}]$	list of differences between the elements of D_{a_1,b_1} and those of D_{a_2,b_2}	
$f_{a_1,a_2,h}$	frequency of h in $[D_{a_1,0} - D_{a_2,0}]$	note: $f_{a,a,h} = f_{a,h}$
$\mathcal{D}_a = (N, \mathcal{B}_a)$	orbital disk-design: development of $D_{a,0}$	$\mathcal{B}_a = \{\mathcal{D}(a;b) \mid b \in N\}$
$\mathcal{D} = (N, \mathcal{B}^{\mathcal{D}})$	(union) disk-design	$\mathcal{B}^{\mathcal{D}} = \bigcup_{a \in N^0} \mathcal{B}_a = \{D_{a,b} \mid a \in N^0, b \in N\}$

Table 2. Parameters for association schemes

$x, y \in \mathcal{R}_i$	ith associate elements	if $x \in \varPhi_i + y$
m	number of the associate classes	$m = (p-1)/2n$
n_i	number of the ith associates elements of an element	$n_i = \lvert\varPhi\rvert = 2n$
p_{ij}^k	number of the ith associates of x and jth associates of y when x and y are kth associates	

Thus, to know the frequency of h in the list $[D_{a_1,0} - D_{a_2,0}]$ it is sufficient to know the frequency of any element of its orbit $\varPhi(h)$ in the same list. Moreover, for any $a \in N^0$, the frequency of h in $[D_{a,0} - D_{a,0}]$ equals the frequency of ha^{-1} in $[D_{1,0} - D_{1,0}]$.

Proposition 2. *Let D_{a_1,b_1} and D_{a_2,b_2} be two blocks of the design $\mathcal{D}$. Set $a = a_2a_1^{-1}$ and $h = (b_2 - b_1)a_1^{-1}$. Then*

$$|D_{a_1,b_1} \cap D_{a_2,b_2}| = f_{1,a,h}.$$

Proof. Let $y \in D_{a_1,b_1} \cap D_{a_2,b_2}$. From Corollary 1 there are $x, \overline{x} \in D_{1,0}$ so that $y = a_1x + b_1 = a_2\overline{x} + b_2$. Setting $a = a_2a_1^{-1}$ and $h = (b_2 - b_1)a_1^{-1}$, we obtain $x - a\overline{x} = h$, thus $h \in [D_{1,0} - D_{a,0}]$. Suppose that $y' = a_1x' + b_1 = a_2\overline{x}' + b_2$ belongs to $D_{a_1,b_1} \cap D_{a_2,b_2}$, then $x' - a\overline{x}' = h$, also. If $y' \neq y$ we have $a_1x \neq a_1x'$, and this implies $x \neq x'$, as well as $a_2\overline{x} \neq a_2\overline{x}'$ implies $a\overline{x} \neq a\overline{x}'$. So, two different elements in $D_{a_1,b_1} \cap D_{a_2,b_2}$ produce two different occurrences of h in $[D_{1,0} - D_{a,0}]$. Conversely, if h occurs in $[D_{1,0} - D_{a,0}]$, we have $h = x'' - a\overline{x}''$ for $a = a_2a_1^{-1}$, $h = (b_2 - b_1)a_1^{-1}$ and for some $x'', \overline{x}'' \in D_{1,0}$. Thus, there exists $y'' = a_1x'' + b_1 = a_2\overline{x}'' + b_2$ belonging to $D_{a_1,b_1} \cap D_{a_2,b_2}$. Obviously, two different occurrences of h in $[D_{1,0} - D_{a,0}]$ imply two different elements in $D_{a_1,b_1} \cap D_{a_2,b_2}$.

Remark 2. To compute the p_{ij}^k as well, we can observe that p_{ij}^k equals the frequency of e_k, the representative of $\varPhi_k$, in the list $[\varPhi_i - \varPhi_j]$ of the differences between the elements of $\varPhi_i$ and those of $\varPhi_j$.

4 Incidence Matrix of a Disk-Design

If $\mathcal{B} = \{B_1, \dots, B_b\}$ is a block design on $X = \{x_1, \dots, x_v\}$ of parameters (v, b, r, k), the $v \times b$ matrix $A = (a_{y,z})$ is called *incidence matrix* of the block design when $a_{y,z} = 1$ if $x_y \in B_z$ and $a_{y,z} = 0$ if $x_y \notin B_z$.

Table 3. Parameters for tactical configurations and PBIBDs

	tactical configuration				partial balance
design	v	b	k	r	λ_i
$\mathcal{D}_{e_s} = (N, \mathcal{B}_{e_s})$	p	p	$2n^2 + 1$	k	$(\lambda_i)_s = f_{e_s,e_i} = f_{1,e_i e_s^{-1}}$
$\mathcal{D} = (N, \mathcal{B}^{\mathcal{D}})$	p	$p(p-1)/2n$	$2n^2 + 1$	$k(p-1)/2n$	$\lambda = n(2n^2 + 1)$

Here we remember that a matrix of the form

$$\begin{pmatrix} c_0 & c_1 & \dots & c_{n-1} \\ c_{n-1} & c_0 & \dots & c_{n-2} \\ \vdots & \vdots & & \vdots \\ c_1 & c_2 & \dots & c_0 \end{pmatrix}$$

is called *circulant matrix* and, for a given circulant matrix C, we have

$$\det(C) = \prod_{j=1}^{n} f(\epsilon_j) \qquad \text{where} \qquad f(x) = \sum_{z=0}^{n-1} c_z x^z \tag{1}$$

is the defining polinomial of C and ϵ_1, ϵ_2, $\dots$, ϵ_n denote the distinct nth roots of unity, (see [9,16]).

Let $A_s = (a_{y,z})$ be the $p \times p$ incidence matrix of $\mathcal{D}_{e_s}$, where $a_{y,z} = 1$ if, and only if, $y - 1 \in D_{e_s,z-1}$ and $a_{y,z} = 0$ otherwise, for $y, z = 1, \dots, p$. The incidence matrix A of the union disk-design $\mathcal{D} = \bigcup_{e_s \in E} \mathcal{D}_{e_s}$, is a $p \times mp$ matrix obtained by the juxtaposition of the A_ss, that is $A = (A_1 A_2 \dots A_m)$.

Theorem 4. *Let $\mathcal{D}_{e_s}$ be the orbital disk-design generated by the development of the disk of center* 0 *and a fixed radius $e_s \in E$. Then, for $s = 1, \dots, m$, the following results hold:*

1. *A_s is a symmetric circulant matrix;*
2. *$det(A_s) = k \prod_{j=1}^{p-1} \left(\sum_{z \in D_{1,0}} \epsilon^{jz} \right)$;*
3. *$H_s = A_s \cdot A_s^T$ is a symmetric circulant matrix and*

$$det(H_s) = \prod_{j=1}^{p} \left(k + \sum_{i=1}^{m} (\lambda_i)_s \sum_{z \in \Phi_i} \epsilon^{jz} \right)$$

where ϵ is a primitive p^{th} root of unity and $(\lambda_1)_s, \dots, (\lambda_m)_s$ are the partial balance parameters of the orbital design $\mathcal{D}_{e_s}$.

Proof. 1. $A_s = (a_{y,z})$ is circulant, because $y-1 \in D_{e_s,z-1}$ if, and only if, $y \in D_{e_s,z}$, for $y,z = 1,\dots,p$. This implies $a_{y,z} = a_{y+1,z+1}$, where the subscripts are considered modulo p for all $y,z = 1,\dots,p$.

Moreover, $A_s = (a_{y,z})$ is symmetric, because all the orbits are self paired being $|\Phi|$ even. Thus, if $a_{y,1} = 1$, that means $y-1 \in D_{e_s,0}$, it results in $-(y-1) \in D_{e_s,0}$. Hence $0 \in D_{e_s,y-1}$, that is $a_{1,y} = 1$. The converse is analogous, so $a_{y,1} = a_{1,y}$ for $y = 1,\dots,p$. Using this last statement in addition to the circulant definition, when $y > z$ we have $a_{y,z} = a_{y-(z-1),z-(z-1)} = a_{y-z+1,1} = a_{1,y-z+1} = a_{1+(z-1),y-z+1+(z-1)} = a_{z,y}$.

2. Setting $a_{1,z} = c_{z-1}$ and applying (1), for $s = 1,\dots,m$ we have

$$det(A_s) = \prod_{j=1}^{p}\left(\sum_{z=0}^{p-1} c_z \epsilon_j^z\right) = \prod_{j=1}^{p}\left(\sum_{z=0}^{p-1} c_z \epsilon^{jz}\right) \tag{2}$$

where ϵ is a primitive p^{th} root of unity and we set $\epsilon_j = \epsilon^j$. We know that $c_z = a_{1,z+1} = 1$ if, and only if, $0 \in D_{e_s,z}$ and this happens if, and only if, $-z \in D_{e_s,0}$, for $z = 0,\dots,p-1$. To be $|\Phi|$ even implies $-z \in D_{e_s,0}$ if, and only if, z itself belongs to $D_{e_s,0}$, because the Φ-orbits are self paired. So $c_z = 1 \iff z \in D_{e_s,0}$.

Thus, applying (2), for $s = 1,\dots,m$ we have

$$det(A_s) = \prod_{j=1}^{p}\left(\sum_{z\in D_{e_s,0}} \epsilon^{jz}\right) = \prod_{j=1}^{p}\left(\sum_{z\in jD_{e_s,0}} \epsilon^{z}\right).$$

Finally, for every $s = 1,\dots,m$, we can see that $(D_{e_s,0}, 2D_{e_s,0},\dots,pD_{e_s,0})$ becomes $\big(e_s(D_{1,0}), 2e_s(D_{1,0}),\dots,pe_s(D_{1,0})\big)$, being $D_{e_s,0} = e_s(D_{1,0})$.

Since $\{e_s, 2e_s,\dots,pe_s\} = \mathbb{Z}_p$, rearranging the previous sequence we obtain

$$\big((D_{1,0}), 2(D_{1,0}),\dots,p(D_{1,0})\big).$$

Thus, $\forall s = 1,\dots,m$,

$$det(A_s) = \prod_{j=1}^{p}\left(\sum_{z\in D_{1,0}} \epsilon^{jz}\right) = k\prod_{j=1}^{p-1}\left(\sum_{z\in D_{1,0}} \epsilon^{jz}\right)$$

being $\sum_{z\in D_{1,0}} \epsilon^{pz} = |D_{1,0}| = k$.

3. The matrix $H_s = A_s \cdot A_s^T = (h_{y,z})$ is obviously symmetric and circulant. The element $h_{y,z}$ gives us the number of the blocks of $\mathcal{D}_{e_s}$ containing both the elements $y-1$ and $z-1$, hence $h_{y,y} = k$, for $y = 1,\dots,p$. When $y \neq z$, $y-1$ and $z-1$ are i-associates if, and only if, their difference belongs to Φ_i. So, $z \in \Phi_i$ implies $h_{1,z+1} = (\lambda_i)_s$, for $i = 1,\dots,m$. Applying again (1), where $c_z = h_{1,z+1}$, we can write

$$det(H_s) = \prod_{j=1}^{p}\left(\sum_{z=0}^{p-1} c_z \epsilon^{jz}\right) = \prod_{j=1}^{p}\left(k + \sum_{i=1}^{m}(\lambda_i)_s \sum_{z\in\Phi_i} \epsilon^{jz}\right)$$

being $c_0 = k$ and $c_z = (\lambda_i)_s$ when $z \in \Phi_i$, for $i = 1,\dots,m$.

Remark 3. Let $\mathcal{D} = \bigcup_{e_s \in E} \mathcal{D}_{e_s}$ be the union disk-design and $A = (A_1 A_2 \dots A_m)$ its $p \times mp$ incidence matrix. Since $\mathcal{D}$ is BIBD, we know that $H = A \cdot\ A^T$ is the $p \times p$ matrix

$$H = \begin{pmatrix} r & \lambda & \dots & \lambda \\ \lambda & r & \dots & \lambda \\ \vdots & \vdots & & \vdots \\ \lambda & \lambda & \dots & r \end{pmatrix}$$

and $det(H) = (r - \lambda)^{p-1}[r + \lambda(p-1)]$.

5 Row and Column Codes from a Disk-Design

A *binary code* of *length* n is a subset $\mathcal{C}$ of $\mathbb{Z}_2^n$. The *weight* of a codeword is the number of its non zero coordinate places. The *Hamming distance* between two codewords is the number of coordinate places in which they differ. The smallest of the distances between distinct codewords is called *minimum distance* of $\mathcal{C}$ and denoted by $d(\mathcal{C})$. A binary code of length n having m codewords and minimum distance d is called a *binary* (n, m, d)*-code.* If $\mathcal{C}$ is a vector subspace of $\mathbb{Z}_2^n$ and $dim(\mathcal{C}) = k$, then it is called a binary *linear* (n, k)*-code.*

It is well known that there is a link between block designs and codes via the design incidence matrix A: the set of all the columns of A, the set of all the rows of A, as well as their linear hulls, can be regarded as binary codes and A itself can be regarded as the parity check matrix of a linear code. For more information on codes see [3,17].

In what follows we are interested in the set $\mathcal{C}_c$ of all the columns of A and the set $\mathcal{C}_r$ of all the rows of A, the so called *column code* and *row code*, respectively. The parameters characterizing $\mathcal{C}_c$ and $\mathcal{C}_r$ depend on the design parameters, as we summarize in the following proposition.

Proposition 3. *Let* $(N, \mathcal{B})$ *be a* $PBIBD$ *with parameters* $(v, b, r, k, \lambda_1, \dots, \lambda_m)$, *$A$ its incidence matrix, $\mathcal{C}_r$ and $\mathcal{C}_c$ the related row and column codes. Set* $\lambda = \max\{\lambda_1, \dots, \lambda_m\}$ *and* $\mu = \max\{|B_i \cap B_j|,\ B_i, B_j \in \mathcal{B}, i \neq j\}$*.Then*

1. *the cardinality of $\mathcal{C}_r$ is v, the codeword length is b, each codeword has the same weight r and the minimum distance is $d(\mathcal{C}_r) = 2(r - \lambda)$;*
2. *the cardinality of $\mathcal{C}_c$ is b, the codewords length is v, each codeword has the same weight k and the minimum distance is $d(\mathcal{C}_c) = 2(k - \mu)$.*

Now, we come back to the $PBIBD$s constructed before, our *orbital disk-designs*, and we consider their incidence matrices. Working on $\mathbb{Z}_p$ with $|\Phi| = 2n$, each orbital disk-design has a $p \times p$ matrix A_i, for $i = 1, \dots, m$, and $A = (A_1 \dots A_m)$ is the incidence matrix of the union disk-design. Applying previous proposition and using the keys of Paragraph 3.2, we are able to compute all the disk-design parameters as well as those of the row (and column) codes related to their incidence matrices. Results are summarized in Table 4.

Table 4. Parameters for row and column codes where $k = 2n^2 + 1$, $m = (p-1)/2n$, $\lambda_{e_s} = \max\{f_{1,e_i e_s^{-1}} \mid e_i \in E\}$, $\lambda = n(2n^2+1)$, $\mu = \max\{f_{1,e_i,e_j} \mid (e_i, e_j) \in E \times E\}$. `orbCr` and `orbCc` are the row and column codes obtained from the orbital disk-design, respectively. `diskCr` and `diskCc` are the row and column codes obtained from the union disk-design, respectively.

design	code	words	length	d	weight
$\mathcal{D}_{e_s}$	`orbCr/orbCc`	p	p	$2(k-\lambda_{e_s})$	k
$\mathcal{D}$	`diskCr`	p	mp	$2(mk-\lambda)$	mk
$\mathcal{D}$	`diskCc`	mp	p	$2(k-\mu)$	k

Table 5. Parameters of some row and column codes generated by the circular Ferrero pair (N, Φ). For codes `orbCr`, `orbCc`, `diskCc` and `diskCr` the parameters (n, m, d); for linear codes `linCr` the parameters (n, k) and the minimum distance d are reported. All designs were constructed via the SONATA [1] GAP package and the parameters checked via the GUAVA [8] GAP package.

(N, Φ)	$\lvert\Phi\rvert$	`orbCr/orbCc`	`diskCc`	`diskCr`	`linCr`
$(\mathbb{Z}_{13}, \langle 5 \rangle)$	4	$(13, 13, 4)$	$(13, 39, 4)$	$(39, 13, 18)$	$(39, 13)$, d=9
$(\mathbb{Z}_{17}, \langle 4 \rangle)$	4	$(17, 17, 6)$	$(17, 68, 6)$	$(68, 17, 36)$	$(68, 17)$, d=20
$(\mathbb{Z}_{29}, \langle 12 \rangle)$	4	$(29, 29, 6)$	$(29, 203, 6)$	$(203, 29, 90)$	$(203, 29)$, d=63
$(\mathbb{Z}_{31}, \langle 6 \rangle)$	6	$(31, 31, 10)$	$(31, 155, 10)$	$(155, 31, 76)$	$(155, 31)$, d=35
$(\mathbb{Z}_{37}, \langle 6 \rangle)$	4	$(37, 37, 6)$	$(37, 333, 6)$	$(333, 37, 126)$	$(333, 37)$, d =81
$(\mathbb{Z}_{37}, \langle 11 \rangle)$	6	$(37, 37, 10)$	$(37, 222, 10)$	$(222, 37, 114)$	$(222, 37)$, d=66
$(\mathbb{F}_{25}, \langle 4x+3 \rangle)$	6	$(25, 25, 8)$	$(25, 100, 4)$	$(100, 25, 32)$	$(100, 25)$, d=20

Although no general criterion exists to decide the efficiency of a code, we can observe that these codes turn out to have some useful properties. For example all of them have a small redundancy R, which is the length of the codewords$-\log|\mathcal{C}|$ (see [17], observe that for linear codes redundancy is sometimes defined as the difference between dimension and word length) and the row code from $\mathcal{D}$ has a high minimum distance. An incidence matrix of a design generates a row (resp. column) linear code whose codewords are the linear combinations of its rows (resp. columns). As the incidence matrix of an orbital disk-design has always maximum rank, it generates a trivial linear code (all possible words are codewords). The same holds for the column linear code generated by a matrix of a union disk-design. On the other hand, the row linear code generated by $\mathcal{D}$ (called `linCr`) proves to be quite interesting. All `linCr`s, we obtain, have a high minimum distance, e.g., the `linCr` obtained from the Ferrero pair $(\mathbb{Z}_{29}, \langle 12 \rangle)$ is a linear 31-error-correcting code. Moreover, we observe that the generated `linCr`s of Table 5, result to be self-complementary, i.e., for any codeword c, also $\underline{1} - c$, where $\underline{1}$ is the all-one word, is a codeword. Finally, in the last entry of Table 5 we consider codes obtained from the union disk-design generated from $(\mathbb{F}_{25}, \langle 4x+3 \rangle)$, with $\mathbb{F}_{25} \simeq \mathbb{Z}_5[x]/\langle x^2+2 \rangle$ but also that ones obtained from the orbital disk-design. So we observe that the construction of the PBIBDs shown above could be carried out also for general field-generated Ferrero pairs.

6 Future Work

We are interested in studying further properties of the codes we obtained, in particular to establish when the `linCr` code results self-complementary. We are also implementing some GAP functions in order to simplify the construction of the disk-designs and the calculation of the parameters of the associated code. Efficient decoding strategies can be envisaged for such codes, as in the non-linear cases codewords have the property they are cyclic permutation of a word of type $1ww^r$, where w^r is the reversed of the word w.

References

1. Aichinger, E., Binder, F., Ecker, J., Mayr, P., Nöbauer, C.: SONATA - system of near-rings and their applications, GAP package, Version 2 (2003), http://www.algebra.uni-linz.ac.at/Sonata/
2. Aichinger, E., Ecker, J., Nöbauer, C.: The use of computers in near-rings theory. In: Fong, Y., et al. (eds.) Near-Rings and Near-Fields, pp. 35–41 (2001)
3. Assmus Jr., E., Key, J.: Designs and their codes. Cambridge Tracts in Mathematics, vol. 103. Cambridge University Press, Cambridge (1992)
4. Bannai, E.: An introduction to association schemes. Quad. Mat. 5, 1–70 (1999)
5. Beth, T., Jungnickel, D., Lenz, H.: Design Theory. Cambridge University Press, Cambridge (1999)
6. Clay, J.: Nearrings: Geneses and Applications. Oxford University Press, Oxford (1992)
7. Clay, J.: Geometry in fields. Algebra Colloq. 1(4), 289–306 (1994)
8. Cramwinckel, J., Roijackers, E., Baart, R., Minkes, E., Ruscio, L., Joyner, D.: GUAVA, a GAP Package for computing with error-correcting codes (2009), http://www.opensourcemath.org/guava/
9. Davis, P.: Circulant Matrices. Chelsea Publishing, New York (1994)
10. Eggetsberger, R.: On extending codes from finite Ferrero pairs. Contributions to General Algebra 9, 151–162 (1994)
11. Eggetsberger, R.: Circles and their interior points from field generated Ferrero pairs. In: Saad, G., Thomsen, M. (eds.) Nearrings, Nearfields and K-Loops, pp. 237–246 (1997)
12. Frigeri, A., Morini, F.: Circular planar nearrings: geometrical and combinatorial aspects (submitted), http://arxiv.org/
13. Fuchs, P.: A decoding method for planar near-ring codes. Riv. Mat. Univ. Parma 4(17), 325–331 (1991)
14. Fuchs, P., Hofer, G., Pilz, G.: Codes from planar near-rings. IEEE Trans. Inform. Theory 36, 647–651 (1990)
15. Hopper, N., von Ahn, L., Langford, J.: Provably secure steganography. IEEE Transactions on Computers 58(5), 662–676 (2009)
16. Lancaster, P., Timenetsky, M.: The Theory of Matrices with Applications. Academic Press, New York (1985)
17. McWilliams, F., Sloane, N.: The Theory of Error-correcting Codes. North-Holland, Amsterdam (1977)
18. Meir, O.: IP=PSPACE using Error Correcting Codes. Electronic Colloquium on Computational Complexity 17(137) (2010)
19. Street, A., Street, D.: Combinatorics of Experimental Design. Oxford University Press, New York (1987)

Independence of Hyperlogarithms over Function Fields via Algebraic Combinatorics

Matthieu Deneufchâtel[1], Gérard H.E. Duchamp[1], Vincel Hoang Ngoc Minh[1], and Allan I. Solomon[2]

[1] Institut Galilée, LIPN - UMR 7030 CNRS - Université Paris 13 F-93430 Villetaneuse, France
[2] Department of Physics and Astronomy, The Open University, Milton Keynes MK7 6AA, UK and LPTMC - UMR 7600 CNRS - Université Paris 6 F-75252 Paris, France

Abstract. We obtain a necessary and sufficient condition for the linear independence of solutions of differential equations for hyperlogarithms. The key fact is that the multiplier (i.e. the factor M in the differential equation $dS = MS$) has only singularities of first order (Fuchsian-type equations) and this implies that they freely span a space which contains no primitive. We give direct applications where we extend the property of linear independence to the largest known ring of coefficients.

1 Introduction

In his 1928 study of the solutions of linear differential equations following Poincaré, Lappo-Danilevski introduced the so-called *hyperlogarithmic functions of order n*, functions of iterated integrals of the following form with logarithmic poles [1] :

$$L(a_0, \ldots, a_n|z, z_0) = \int_{z_0}^{z} \int_{z_0}^{s_n} \ldots \int_{z_0}^{s_1} \frac{ds_0}{s_0 - a_0} \ldots \frac{ds_n}{s_n - a_n}, \tag{1}$$

where z_0 is a fixed point. It suffices that $z_0 \neq a_0$ for this iterated integral to converge. The classical polylogarithm Li_n is a particular case of these integrals [2] :

$$\mathrm{Li}_n(z) = \int_{0}^{z} \int_{0}^{s_n} \ldots \int_{0}^{s_2} \frac{ds_1}{1 - s_1} \frac{ds_2}{s_2} \ldots \frac{ds_n}{s_n} = -L(1, \underbrace{0, \ldots, 0}_{n-1 \text{ times}} |z, 0). \tag{2}$$

These iterated integrals also appear in quantum electrodynamics (see [3,4] for example). Chen [5] studied them systematically and provided a noncommutative algebraic context in which to treat them. Fliess [6,7] encoded these iterated integrals by words over a finite alphabet and extended them to a symbolic calculus[1] for nonlinear differential equations of the following form, in the context of noncommutative formal power series:

[1] A kind of Feynman like operator calculus [8].

$$\begin{cases} y(z) &= f(q(z)),\\ \dot{q}(z) &= \displaystyle\sum_{i=0}^{m} \frac{A_i(q)}{z-a_i},\\ q(z_0) = q_0, \end{cases} \tag{3}$$

where the state $q = (q_1, \ldots, q_n)$ belongs to a complex analytic manifold of dimension N, q_0 denotes the initial state, the observable f belongs to $\mathbb{C}^{\mathrm{cv}}[\![q_1, \ldots, q_N]\!]$, and $\{A_i\}_{i=0,n}$ is the polysystem defined as follows

$$A_i(q) = \sum_{j=1}^{n} A_i^j(q) \frac{\partial}{\partial q_j}, \tag{4}$$

with, for any $j = 1, \ldots, n$, $A_i^j(q) \in \mathbb{C}^{\mathrm{cv}}[\![q_1, \ldots, q_N]\!]$.

By introducing the encoding alphabet $X = \{x_0, \ldots, x_m\}$, the method of Fliess consists in exhibiting two formal power series over the monoid X^* :

$$F := \sum_{w \in X^*} \mathcal{A}(w) \circ f_{|_{q_0}}\ w \text{ and } C := \sum_{w \in X^*} \alpha_{z_0}^{z}(w)\ w \tag{5}$$

in order to compute the output y. These series are subjected to convergence conditions (precisely speaking, the convergence of a duality pairing), as follows:

$$y(z) = \langle F || C \rangle := \sum_{w \in X^*} \mathcal{A}(w) \circ f_{|_{q_0}}\ \alpha_{z_0}^{z}(w), \tag{6}$$

where

- $\mathcal{A}$ is a morphism of algebras from $\mathbb{C}\langle\langle X \rangle\rangle$ to the algebra generated by the polysystem $\{A_i\}_{i=0,n}$:

$$\mathcal{A}(1_{X^*}) = \text{identity}, \tag{7}$$

$$\forall w = v x_i, x_i \in X, v \in X^*, \quad \mathcal{A}(w) = \mathcal{A}(v) A_i \tag{8}$$

- $\alpha_{z_0}^{z}$ is a shuffle algebra morphism from $(\mathbb{C}\langle\langle X \rangle\rangle, ш)$ to some differential ring $\mathcal{C}$:

$$\alpha_{z_0}^{z}(1_{X^*}) = 1, \tag{9}$$

$$\forall w = v x_i, x_i \in X, v \in X^*, \quad \alpha_{z_0}^{z}(w) = \int_{z_0}^{z} \frac{\alpha_{z_0}^{s}(v)}{s - a_i}. \tag{10}$$

Formula (6) also states that the iterated integrals over the rational functions

$$u_i(z) = \frac{1}{z - a_i}, \ \ i = 0, .., n, \tag{11}$$

span the vector space $\mathcal{A}$.

As for the linear differential equations, the essential difficulty is to construct the fundamental system of solutions, or the Picard-Vessiot extension, to describe the space of solutions of the differential system (3) algorithmically [9]. For that, one needs to prove the linear independence of the iterated integrals in order to obtain the *universal* Picard-Vessiot extension. The $\mathbb{C}$-linear independence was already shown by Wechsung [10]. His method uses a recurrence based on the total degree. However this method cannot be used with variable coefficients. Another proof, based on monodromy, was given in [11] but also leads to strong restrictions on the coefficients. In this note we describe a general theorem in differential computational algebra and show that, at the cost of using variable domains (which is the realm of germ spaces), and replacing the recurrence on total degree by a recursion on the words (with graded lexicographic ordering), one can encompass the previous results mentioned above and obtain much larger rings of coefficients and configuration alphabets (even infinite of continuum cardinality).

2 Non Commutative Differential Equations

We recall the Dirac-Schützenberger notation, as in [12,13,14]. Let X be an alphabet and R be a commutative ring with unit. The algebra of noncommutative polynomials is the algebra $R[X^*]$ of the free monoid X^*. As an R-module, $R^{(X^*)}$ is the set of finitely supported R-valued function on X^* and, as such, it is in natural duality with the algebra of all functions on X^* (the large algebra of X^* [15]), $R^{X^*} = R\langle\langle X\rangle\rangle$, the duality being given, for $f \in R\langle\langle X\rangle\rangle$ and $g \in R[X^*]$, by

$$\langle f|g\rangle = \sum_{w\in X^*} f(w)g(w)\ . \tag{12}$$

The rôle of the ring is played here by a commutative differential k-algebra $(\mathcal{A}, d)$; that is, a k-algebra $\mathcal{A}$ (associative and commutative with unit) endowed with a distinguished derivation $d \in \mathfrak{Der}(\mathcal{A})$ (the ground field k is supposed commutative and of characteristic zero). We assume that the ring of constants $ker(d)$ is precisely k.

An alphabet X being given, one can at once extend the derivation d to a derivation of the algebra $\mathcal{A}\langle\langle X\rangle\rangle$ by

$$\mathbf{d}(S) = \sum_{w\in X^*} d(\langle S|w\rangle)w\ . \tag{13}$$

We now act with this derivation $\mathbf{d}$ on the power series C given in (5). We then get :

$$\mathbf{d}(C) = \left(\sum_{i=1}^{m} u_i x_i\right)C\ . \tag{14}$$

We are now in a position to state the main theorem which resolves many important questions, some of which we shall see in the applications.

Theorem 1. *Let $(\mathcal{A}, d)$ be a k-commutative associative differential algebra with unit ($ch(k) = 0$) and $\mathcal{C}$ be a differential subfield of $\mathcal{A}$ (i.e. $d(\mathcal{C}) \subset \mathcal{C}$). We suppose that $S \in \mathcal{A}\langle\langle X\rangle\rangle$ is a solution of the differential equation*

$$\mathbf{d}(S) = MS\ ;\ \langle S|1\rangle = 1 \tag{15}$$

where the multiplier M is a homogeneous series (a polynomial in the case of finite X) of degree 1, i.e.

$$M = \sum_{x\in X} u_x x \in \mathcal{C}\langle\langle X\rangle\rangle \ . \tag{16}$$

The following conditions are equivalent :

i) The family $(\langle S|w\rangle)_{w\in X^*}$ of coefficients of S is free over $\mathcal{C}$.
ii) The family of coefficients $(\langle S|y\rangle)_{y\in X\cup\{1_{X^*}\}}$ is free over $\mathcal{C}$.
iii) The family $(u_x)_{x\in X}$ is such that, for $f\in\mathcal{C}$ and $\alpha_x\in k$

$$d(f) = \sum_{x\in X} \alpha_x u_x \Longrightarrow (\forall x\in X)(\alpha_x = 0) \ . \tag{17}$$

iv) The family $(u_x)_{x\in X}$ is free over k and

$$d(\mathcal{C}) \cap \operatorname{span}_k \Big((u_x)_{x\in X}\Big) = \{0\} \ . \tag{18}$$

Proof — (i)$\Longrightarrow$(ii) Obvious.

(ii)$\Longrightarrow$(iii)
Suppose that the family $(\langle S|y\rangle)_{y\in X\cup\{1_{X^*}\}}$ (coefficients taken at letters and the empty word) of coefficients of S is free over $\mathcal{C}$ and let us consider the relation as in eq. (17)

$$d(f) = \sum_{x\in X} \alpha_x u_x \ . \tag{19}$$

We form the polynomial $P = -f1_{X^*} + \sum_{x\in X}\alpha_x x$. One has $\mathbf{d}(P) = -d(f)1_{X^*}$ and

$$d(\langle S|P\rangle) = \langle \mathbf{d}(S)|P\rangle + \langle S|\mathbf{d}(P)\rangle = \langle MS|P\rangle - d(f)\langle S|1_{X^*}\rangle = (\sum_{x\in X}\alpha_x u_x) - d(f) = 0 \tag{20}$$

whence $\langle S|P\rangle$ must be a constant, say $\lambda\in k$. For $Q = P - \lambda.1_{X^*}$, we have

$$\operatorname{supp}(Q) \subset X\cup\{1_{X^*}\} \text{ and } \langle S|Q\rangle = \langle S|P\rangle - \lambda\langle S|1_{X^*}\rangle = \langle S|P\rangle - \lambda = 0 \ .$$

This implies that $Q = 0$ and, as $Q = -(f+\lambda)1_{X^*} + \sum_{x\in X}\alpha_x x$, one has, in particular, all the $\alpha_x = 0$.

(iii)$\Longleftrightarrow$(iv)
Obvious, (iv) being a geometric reformulation of (iii).
(iii)$\Longrightarrow$(i)
Let $\mathcal{K}$ be the kernel of $P\mapsto\langle S|P\rangle$ (a linear form $\mathcal{C}\langle X\rangle\to\mathcal{C}$) i.e.

$$\mathcal{K} = \{P\in\mathcal{C}\langle X\rangle | \langle S|P\rangle = 0\} \ . \tag{21}$$

If $\mathcal{K} = \{0\}$, we are done. Otherwise, let us adopt the following strategy.

First, we order X by some well-ordering $<$ ([16] III.2.1) and X^* by the graded lexicographic ordering $\prec$ defined by

$$u \prec v \Longleftrightarrow |u| < |v| \text{ or } (u = pxs_1\ ,\ v = pys_2 \text{ and } x < y). \tag{22}$$

It is easy to check that $\prec$ is also a well-ordering relation. For each nonzero polynomial P, we denote by $lead(P)$ its leading monomial; i.e. the greatest element of its support $\mathrm{supp}(P)$ (for $\prec$).

Now, as $\mathcal{R} = \mathcal{K} - \{0\}$ is not empty, let w_0 be the minimal element of $lead(\mathcal{R})$ and choose a $P \in \mathcal{R}$ such that $lead(P) = w_0$. We write

$$P = fw_0 + \sum_{u \prec w_0} \langle P|u\rangle u\ ;\ f \in \mathcal{C} - \{0\}\ . \tag{23}$$

The polynomial $Q = \frac{1}{f}P$ is also in $\mathcal{R}$ with the same leading monomial, but the leading coefficient is now 1; and so Q is given by

$$Q = w_0 + \sum_{u \prec w_0} \langle Q|u\rangle u\ . \tag{24}$$

Differentiating $\langle S|Q\rangle = 0$, one gets

$$\begin{aligned} 0 = \langle \mathbf{d}(S)|Q\rangle + \langle S|\mathbf{d}(Q)\rangle = \langle MS|Q\rangle + \langle S|\mathbf{d}(Q)\rangle = \\ \langle S|M^\dagger Q\rangle + \langle S|\mathbf{d}(Q)\rangle = \langle S|M^\dagger Q + \mathbf{d}(Q)\rangle \end{aligned} \tag{25}$$

with

$$M^\dagger Q + \mathbf{d}(Q) = \sum_{x \in X} u_x (x^\dagger Q) + \sum_{u \prec w_0} d(\langle Q|u\rangle)u \in \mathcal{C}\langle X\rangle\ . \tag{26}$$

It is impossible that $M^\dagger Q + \mathbf{d}(Q) \in \mathcal{R}$ because it would be of leading monomial strictly less than w_0, hence $M^\dagger Q + \mathbf{d}(Q) = 0$. This is equivalent to the recursion

$$d(\langle Q|u\rangle) = -\sum_{x \in X} u_x \langle Q|xu\rangle\ ;\ \text{for } x \in X\ ,\ u \in X^*. \tag{27}$$

From this last relation, we deduce that $\langle Q|w\rangle \in k$ for every w of length $deg(Q)$ and, because $\langle S|1\rangle = 1$, one must have $deg(Q) > 0$. Then, we write $w_0 = x_0 v$ and compute the coefficient at v

$$d(\langle Q|v\rangle) = -\sum_{x \in X} u_x \langle Q|xv\rangle = \sum_{x \in X} \alpha_x u_x \tag{28}$$

with coefficients $\alpha_x = -\langle Q|xv\rangle \in k$ as $|xv| = \deg(Q)$ for all $x \in X$. Condition (17) implies that all coefficients $\langle Q|xu\rangle$ are zero; in particular, as $\langle Q|x_0u\rangle = 1$, we get a contradiction. This proves that $\mathcal{K} = \{0\}$. $\square$

3 Applications

Let V be a connected and simply connected analytic variety (for example, the doubly cut plane $\mathbb{C} - (]-\infty, 0[\cup]1, +\infty[)$, the Riemann sphere or the universal covering of $\mathbb{C} - \{0, 1\}$), and let $\mathcal{H} = C^\omega(V, \mathbb{C})$ be the space of analytic functions on V.

It is possible to enlarge the range of scalars to coefficients that are analytic functions with variable domains $f : \ dom(f) \to \mathbb{C}$.

Definition 1. *We define a differential field of germs as the data of a filter basis $\mathcal{B}$ [18] of open connected subsets of V, and a map $\mathcal{C}$ defined on $\mathcal{B}$ such that for every $U \in \mathcal{B}$, $\mathcal{C}[U]$ is a subring of $C^\omega(U, \mathbb{C})$ and*

1. *$\mathcal{C}$ is compatible with restrictions i.e. if $U, V \in \mathcal{B}$ and $V \subset U$, one has*
$$res_{VU}(\mathcal{C}[U]) \subset \mathcal{C}[V]$$
2. *if $f \in \mathcal{C}[U] \setminus \{0\}$ then there exists $V \in \mathcal{B}$ s.t. $V \subset U - \mathcal{O}_f$ and $\frac{1}{f}$ (defined on V) is in $\mathcal{C}[V]$.*

There are important cases where the conditions (2) are satisfied as shown by the following theorem.

Theorem 2. *Let V be a simply connected non-void open subset of $\mathbb{C} - \{a_1, \cdots a_n\}$ ($\{a_1, \cdots a_n\}$ are distinct points), $M = \sum_{i=1}^n \frac{\lambda_i x_i}{z - a_i}$ be a multiplier on $X = \{x_1, \cdots x_n\}$ with all $\lambda_i \neq 0$ and S be any regular solution of*

$$\frac{d}{dz} S = MS \ . \tag{29}$$

Then, let $\mathcal{C}$ be a differential field of functions defined on V which does not contain linear combinations of logarithms on any domain but which contains z and the constants (as, for example the rational functions).

If $U \in \mathcal{B}$ (i.e. U is a domain of $\mathcal{C}$) and $P \in \mathcal{C}[U]\langle X \rangle$, one has

$$\langle S|P \rangle = 0 \Longrightarrow P = 0 \tag{30}$$

Proof — Let $U \in \mathcal{B}$. For every non-zero $Q \in \mathcal{C}[U]\langle X \rangle$, we denote by $lead(Q)$ the greatest word in the support of Q for the graded lexicographic ordering $\prec$. We endow X with an arbitrary linear ordering, and call Q monic if the leading coefficient $\langle Q|lead(Q) \rangle$ is 1. A monic polynomial is then given by

$$Q = w + \sum_{u \prec w} \langle Q|u \rangle u \ . \tag{31}$$

Now suppose that it is possible to find U and $P \in \mathcal{C}[U]\langle X \rangle$ (not necessarily monic) such that $\langle S|P \rangle = 0$; we choose P with $lead(P)$ minimal for $\prec$.

Then

$$P = f(z)w + \sum_{u \prec w} \langle P|u\rangle u \tag{32}$$

with $f \not\equiv 0$. Thus $U_1 = U \setminus \mathcal{O}_f \in \mathcal{B}$ and $Q = \frac{1}{f(z)} P \in \mathcal{C}[U_1]\langle X\rangle$ is monic and satisfies

$$\langle S|Q\rangle = 0 \ . \tag{33}$$

Differentiating eq. (33), we get

$$0 = \langle S'|Q\rangle + \langle S|Q'\rangle = \langle MS|Q\rangle + \langle S|Q'\rangle = \langle S|Q' + M^\dagger Q\rangle \ . \tag{34}$$

Remark that one has

$$Q' + M^\dagger Q \in \mathcal{C}[U_1]\langle X\rangle \tag{35}$$

If $Q' + M^\dagger Q \neq 0$, one has $lead(Q' + M^\dagger Q) \prec lead(Q)$ and this is not possible because of the minimality hypothesis of $lead(Q) = lead(P)$. Hence, one must have $R = Q' + M^\dagger Q = 0$. With $|w| = n$, we now write

$$Q = Q_n + \sum_{|u|<n} \langle Q|u\rangle u \tag{36}$$

where $Q_n = \sum_{|u|=n} \langle Q|u\rangle u$ is the dominant homogeneous component of Q. For every $|u| = n$ we have

$$(\langle Q|u\rangle)' = -\langle M^\dagger Q|u\rangle = -\langle Q|Mu\rangle = 0 \tag{37}$$

thus all the coefficients of Q_n are constant.

If $n = 0$, $Q \neq 0$ is constant which is impossible by eq. (33) and because S is regular. If $n > 0$, for any word $|v| = n - 1$, we have

$$(\langle Q|v\rangle)' = -\langle M^\dagger Q|v\rangle = -\langle Q|Mv\rangle = -\sum_{i=0}^{n} \frac{\lambda_i}{z - a_i} \langle Q|x_i v\rangle = -\sum_{i=0}^{n} \frac{\lambda_i}{z - a_i} \langle Q_n|x_i v\rangle \tag{38}$$

because all $x_i v$ are of length n.

Then

$$\langle Q|v\rangle = -\sum_{i=0}^{n} \langle Q_n|x_i v\rangle \int_\alpha^z \frac{\lambda_i}{s - a_i} ds + const \tag{39}$$

But all the functions $\int_\alpha^z \frac{\lambda_i}{s-a_i} ds$ are linearly independent over $\mathbb{C}$ and not all the scalars $\langle Q_n|x_i v\rangle$ are zero (write $w = x_k v$ and choose v accordingly). This contradicts the fact that $Q \in \mathcal{C}[U_1]\langle X\rangle$ as $\mathcal{C}$ contains no linear combination of logarithms. □

Corollary 1. *Let V be as above and R be the ring of functions which can be analytically extended to some $V \cup U_{a_1} \cup U_{a_2} \cup \cdots U_{a_n}$ where U_{a_i} are open neighborhoods of $a_i, i = 1 \cdots n$ and have non-essential singularities at these points. Then, the family of hyperlogarithms $(\langle S|w\rangle)_{w \in X^*}$ is linearly independent over R.*

Remark 1. i) If a series $S = \sum_{w \in X^*} \langle S|w \rangle w$ is a regular solution of (29) and satisfies the equivalent conditions of the theorem (1), then so too does every Se^C (with $C \in \mathrm{Lie}_{\mathbb{C}}\langle\langle X \rangle\rangle$) .

ii) Series such as that of polylogarithms and all the exponential solutions of equation

$$\frac{d}{dz}(S) = (\frac{x_0}{z} + \frac{x_1}{1-z})S \tag{40}$$

satisfy the conditions of the theorem (1) as shown by theorem (2).

iii) Call $\mathcal{F}(S)$ the vector space generated by the coefficients of the series S. One may ask what happens when the conditions for independence are not satisfied.

In fact, the set of Lie series $C \in \mathrm{Lie}_{\mathbb{C}}\langle\langle \mathbb{X} \rangle\rangle$ such that there exists a $\phi \in \mathrm{End}(\mathcal{F}(S))$ (thus a derivation) such that $SC = \phi(S)$, is a closed Lie subalgebra of $\mathrm{Lie}_{\mathbb{C}}\langle\langle \mathbb{X} \rangle\rangle$ which we will denote by Lie_S. For example

- for $X = \{x_0, x_1\}$ and $S = e^{zx_0}$ one has $x_0 \in Lie_S$; $x_1 \notin Lie_S$
- for $X = \{x_0, x_1\}$ and $S = e^{z(x_0+x_1)}$, one has $x_0, x_1 \notin Lie_S$ but $(x_0 + x_1) \in Lie_S$.

iv) Theorem (2) holds *mutatis mutandis* when the multiplier is infinite i.e.

$$M = \sum_{i \in I} \frac{\lambda_i x_i}{z - a_i}$$

even if I is continuum infinite (say $I = \mathbb{R}$, singularities being all the reals).

v) Theorem (2) no longer holds with singularities of higher order (i.e. not Fuchsian). For example, with

$$M = \frac{x_0}{z^2} + \frac{x_1}{(1-z)^2} \ . \tag{41}$$

Firstly, the differential field $\mathcal{C}$ generated by

$$u_0 = \frac{1}{z^2}, \ u_1 = \frac{1}{(1-z)^2} \tag{42}$$

contains

$$\frac{d}{dz}(\frac{1}{2u_0}) = z \tag{43}$$

and hence $\mathcal{C} = \mathbb{C}(z)$, the field of rational functions over $\mathbb{C}$. Condition (ii) of theorem (1) is not satisfied (as $z^2u_0 - (1-z)^2u_1 = 0$). Moreover, one has $\mathbb{Z}$-dependence relations such as

$$\langle S|x_1x_0 \rangle + \langle S|x_0x_1 \rangle + \langle S|x_1 \rangle - \langle S|x_0 \rangle = 0 \ . \tag{44}$$

4 Through the Looking Glass: Passing from Right to Left

We are still in the context of analytic functions as above. A series $S \in \mathcal{H}\langle\langle X\rangle\rangle$ is said to be group-like if

$$\Delta(S) = S \otimes S \tag{45}$$

where Δ is the dual of the shuffle product [14] defined on series by $\Delta(S) = \sum_{w\in X^*}\langle S|w\rangle\Delta(w)$ and on the words by the recursion ($x \in X, u \in X^*$)

$$\Delta(1_{X^*}) = 1_{X^*} \otimes 1_{X^*} \ ; \ \Delta(xu) = (x \otimes 1_{X^*} + 1_{X^*} \otimes x)\Delta(u) \tag{46}$$

Let $S \in \mathcal{H}\langle\langle X\rangle\rangle$. We call $\mathcal{F}(S)$ the $\mathbb{C}$-vector space generated by the coefficients of S. One has

$$\mathcal{F}(S) = \{\langle S|P\rangle\}_{P\in\mathbb{C}\langle X\rangle} \ . \tag{47}$$

We recall that, for $a \in X$ and $w \in X^*$, the partial degree $|w|_a$ is the number of occurrences of a in w, it is defined by the recursion

$$|1_{X^*}|_a = 0 \ ; \ |bu|_a = \delta_{b,a} + |u|_a \ . \tag{48}$$

Of course the lenght of the word is the sum of the partial degrees i.e. $|w| = \sum_{x\in X}|w|_x$. The function $a \mapsto |w|_a$ belongs to $\mathbb{N}^{(X)}$ (finitely supported functions from X to $\mathbb{N}$). For $\alpha \in \mathbb{N}^{(X)}$, we note $\mathbb{C}_{\leq\alpha}\langle X\rangle$, the set of polynomials $Q \in \mathbb{C}\langle X\rangle$ such that $supp(Q) \subset X^{\leq\alpha}$ i.e.

$$\langle Q|w\rangle \neq 0 \Longrightarrow (\forall x \in X)(|w|_x \leq \alpha(x)) \tag{49}$$

In the same way, we consider the filtration by total degree (length)

$$\mathbb{C}_{\leq n}\langle X\rangle = \sum_{|\alpha|\leq n}\mathbb{C}_{\leq\alpha}\langle X\rangle \ . \tag{50}$$

We use the following increasing filtrations

$$\mathcal{F}_{\leq\alpha}(S) = \{\langle S|P\rangle\}_{P\in\mathbb{C}_{\leq\alpha}\langle X\rangle} \ . \tag{51}$$

or

$$\mathcal{F}_{\leq n}(S) = \{\langle S|P\rangle\}_{P\in\mathbb{C}_{\leq n}\langle X\rangle} \ . \tag{52}$$

Proposition 1. *We have the following properties :*

i) If $T \in \mathbb{C}\langle\langle X\rangle\rangle$ then $\mathcal{F}(ST) \subset \mathcal{F}(S)$ and one has equality if T is invertible.

ii) If S is group-like, then $\mathcal{F}(S)$ is a unital sub-algebra of $\mathcal{H}$, which is filtered w.r.t. (51) and (52) i.e.

$$\mathcal{F}_{\leq\alpha}(S)\mathcal{F}_{\leq\beta}(S) \subset \mathcal{F}_{\leq\alpha+\beta}(S) \tag{53}$$

Proof — (i) The space $\mathcal{F}(ST)$ is spanned by the

$$\langle ST|w\rangle = \sum_{uv=w} \langle S|u\rangle\langle T|v\rangle \in \mathcal{F}(S)$$

and if T is invertible one has $\mathcal{F}(S) = \mathcal{F}(STT^{-1}) \subset \mathcal{F}(ST)$ which proves the equality.
ii) If S is group-like, one has

$$\langle S|u\rangle\langle S|v\rangle = \langle S \otimes S|u \otimes v\rangle = \langle \Delta(S)|u \otimes v\rangle = \langle S|u \sqcup\!\sqcup v\rangle \tag{54}$$

In the case when all the functions $\langle S|w\rangle$ are $\mathbb{C}$-linearly independent, one has a correspondence between the differential Galois group (acting on the right) of a differential equation of type (40) (acting on the right) and the group of automorphisms of $\mathcal{F}(S)$ compatible with the preceding filtration (they turn out to be unipotent).

Proposition 2. *Let S be a group-like series. The following conditions are equivalent:*

i) For every $x \in X$, $ker_{\mathbb{C}}(S) \subset ker_{\mathbb{C}}(Sx)$.

ii) For every $x \in X$, there is a derivation $\delta_x \in \mathfrak{Der}(\mathcal{F}(S))$ such that

$$\delta_x(S) = Sx \tag{55}$$

iii) For every $x \in X$, there is a one-parameter group of automorphisms $\phi_x^t \in Aut(\mathcal{F}(S))$; $t \in \mathbb{R}$ such that

$$\phi_x^t(S) = Se^{tx} \tag{56}$$

iv) For every $C \in \mathrm{Lie}_C\langle\langle X\rangle\rangle$, there is $\delta \in \mathfrak{Der}(\mathcal{F}(S))$ such that

$$\delta(S) = SC \tag{57}$$

v) For every $C \in \mathrm{Lie}_C\langle\langle X\rangle\rangle$, there is $\phi \in Aut(\mathcal{F}(S))$ such that

$$\phi(S) = Se^C \tag{58}$$

vi) The functions $(\langle S|w\rangle)_{w\in X^*}$ are $\mathbb{C}$-linearly independent.

Proof — i) $\Longrightarrow$ ii) From the inclusion, we deduce that, for all $x \in X$ there exists a $\mathbb{C}$-linear mapping $\phi \in \mathrm{End}(\mathcal{F}(S))$ such that for all $w \in \mathcal{M}$, $\phi(\langle S|w\rangle) = \langle Sx|w\rangle$. It must be a derivation of $\mathcal{F}(S)$ as

$$\begin{aligned}\phi(\langle S|u\rangle\langle S|v\rangle) = \phi(\langle S|u \sqcup\!\sqcup v\rangle) = \langle Sx|u \sqcup\!\sqcup v\rangle = \langle S|(u \sqcup\!\sqcup v)x^{-1}\rangle = \\ \langle S|(ux^{-1} \sqcup\!\sqcup v) + (u \sqcup\!\sqcup vx^{-1})\rangle = \langle S|(ux^{-1} \sqcup\!\sqcup v)\rangle\langle S|(u \sqcup\!\sqcup vx^{-1})\rangle = \\ \langle Sx|u\rangle\langle S|v\rangle + \langle S|u\rangle\langle Sx|v\rangle = \phi(\langle S|u\rangle)\langle S|v\rangle + \langle S|u\rangle\phi(\langle S|v\rangle)\end{aligned} \tag{59}$$

from the fact that $(\langle S|w\rangle)_{w\in X^*}$ spans $\mathcal{F}(S)$.

ii) $\Longrightarrow$ *iv)* As $(\langle S|w\rangle)_{w\in X^*}$ spans $\mathcal{F}(S)$, the derivation ϕ is uniquely defined. We denote it by δ_x, and notice that, in so doing, we have constructed a mapping $\Phi : X \to \mathfrak{Der}(\mathcal{F}(S))$, which is a Lie algebra. Therefore, there is a unique extension of this mapping as a morphism $\mathrm{Lie}_{\mathbb{C}}\langle X\rangle \to \mathfrak{Der}(\mathcal{F}(S))$. This correspondence, which we denote by $P \to \delta(P)$, is (uniquely) recursively defined by

$$\delta(x) = \delta_x \ ; \ \delta([P,Q]) = [\delta(P), \delta(Q)] \ . \tag{60}$$

For $C = \sum_{n\geq 0} C_n \in \mathrm{Lie}_{\mathbb{C}}\langle\langle X\rangle\rangle$ with $C_n \in \mathrm{Lie}_{\mathbb{C}}\langle X\rangle_n$, we remark that the sequence $\langle S\sum_{0\leq n\leq N} C_n|w\rangle$ is stable (for large N). Set $\delta_{\leq N} := \delta(\sum_{0\leq n\leq N} C_n)$. We see that $\delta_{\leq N}$ is stable (for large N) on every $\mathcal{F}_\alpha$; we call its limit $\delta(C)$. It is clear that this limit is a derivation and that it corresponds to C.

iv) $\Longrightarrow$ *v)* For every $C = \sum_{n\geq 0} C_n \in \mathrm{Lie}_{\mathbb{C}}\langle\langle X\rangle\rangle$, the exponential e^C defines a mapping $\phi \in \mathrm{End}(\mathcal{F}(S))$ as indeed $e^{\delta_{\leq N}}$ is stationnary. It is easily checked that this mapping is an automorphism of algebra of $\mathcal{F}(S)$.

v) $\Longrightarrow$ *iii)* For $C_i \in \mathrm{Lie}_{\mathbb{C}}\langle\langle X\rangle\rangle; i = 1, 2$ which commute we have

$$Se^{C_1}e^{C_2} = \phi_{C_1}(S)e^{C_2} = \phi_{C_1}(Se^{C_2}) = \phi_{C_1}\phi_{C_2}(S). \tag{61}$$

This proves the existence, for a $C \in \mathrm{Lie}_{\mathbb{C}}\langle\langle X\rangle\rangle$, of a one-parameter (rational) group ϕ_C^t in $\mathrm{Aut}(\mathcal{F}(S))$ such that $Se^{tC} = \phi_C^t(S)$. This one-parameter (rational) group can be extended to $\mathbb{R}$ as continuity is easily checked by taking the scalar products $\langle\phi_C^t(S)|w\rangle = \langle Se^{tC}|w\rangle$ and it suffices to specialize the result to $C = x$.

iii) $\Longrightarrow$ *ii)* By stationary limits one has

$$\langle Sx|w\rangle = \lim_{t\to 0}\frac{1}{t}(\langle Se^{tx}|w\rangle - \langle S|w\rangle) = \lim_{t\to 0}\frac{1}{t}(\langle\phi_x^t(S)|w\rangle - \langle S|w\rangle) \tag{62}$$

v) $\Longrightarrow$ *i)* Let $x \in X, t \in \mathbb{R}$, we take $C = tx$ and $\phi_t \in \mathrm{Aut}(\mathcal{F}(S))$ s.t. $\phi_t(S) = Se^{tx}$. It there is $P \in \mathbb{C}\langle X\rangle$ such that $\langle S|P\rangle = 0$ one has

$$0 = \langle S|P\rangle = \phi_t(\langle S|P\rangle) = \langle\phi_t(S)|P\rangle = \langle Se^{tx}|P\rangle = \sum_{n=0}^{\deg(P)}\frac{t^n}{n!}\langle Sx^n|P\rangle \tag{63}$$

and then, for all $z \in V$, the polynomial

$$\sum_{n=0}^{\deg(P)}\frac{t^n}{n!}\langle S(z)x^n|P\rangle \tag{64}$$

is identically zero over $\mathbb{R}$ hence so are all of its coefficients in particular $\langle S(z)x|P\rangle$ for all $z \in V$. This proves the claim.

$i) \Longrightarrow vi)$ Let $P \in ker_{\mathbb{C}}(S)$ if $P \neq 0$ take it of minimal degree with this property. For all $x \in X$, one has $P \in ker_{\mathbb{C}}(Sx)$ which means $\langle Sx|P\rangle = 0$ and then $Px^{\dagger} = 0$ as $\deg(Px^{\dagger}) = \deg(P) - 1$. The reconstruction lemma implies that

$$P = \langle P|1\rangle + \sum_{x\in X}(Px^{\dagger})x = \langle P|1\rangle \tag{65}$$

Then, one has $0 = \langle S|P\rangle = \langle S|1\rangle\langle P|1\rangle = \langle P|1\rangle$ which shows that $ker_{\mathbb{C}}(S) = \{0\}$. This is equivalent to the statement (vi).

$vi) \Longrightarrow i)$ Is obvious as $ker_{\mathbb{C}}(S) = \{0\}$. $\square$

Remark 2. The derivations δ_x cannot in general be expressed as restrictions of derivations of $\mathcal{H}$. For example, with equation (40), one has $\delta_{x_0}(\frac{\log(z)^{n+1}}{(n+1)!}) = \frac{\log(z)^n}{n!}$ but $\delta_{x_0}(\langle S|ux_1\rangle) = 0$.

5 Conclusion

In this paper, we showed that by using fields of germs, some difficult results can be considerably simplified and extended. For instance, polylogarithms were known to be independant over either $\mathbb{C}[z, 1/z, 1/(1-z)]$ or, presumably, over "functions which do not involve monodromy"; these two results are now encompassed by Theorem (1). We believe that this procedure is not only of theoretical importance, but can be taken into account at the computational level because every formula (especially analytic) carries with it its domain of validity. As a matter of fact, having at hand the linear independence of coordinate functions over large rings allows one to express uniquely solutions of systems like (3) in the basis of hyperlogarithms.

A valuable prospect would be to determine the asymptotic expansion at infinity of the Taylor coefficients of the $y(z)$ as given in (6) for the general case. This has been done already for the case of singularities at $\{0, 1\}$ and for different purposes (see arXiv:1011.0523v2 and http://fr.arxiv.org/abs/0910.1932).

Acknowledgements. The authors are pleased to acknowledge the hospitality of institutions in Paris and UK. We take advantage of these lines to acknowledge support from the French Ministry of Science and Higher Education under Grant ANR PhysComb and local support from the Project "Polyzetas".

References

1. Lappo-Danilevsky, J.A.: Théorie des systèmes des équations différentielles linéaires. Chelsea, New York (1953)
2. Lewin, L.: Polylogarithms and associated functions. North Holland, New York (1981)

3. Dyson, F.J.: The radiation theories of Tomonaga, Schwinger and Feynman. Physical Rev. 75, 486–502 (1949)
4. Magnus, W.: On the Exponential Solution of Differential Equation for a Linear Operator. Comm. Pure Appl. Math. 7, 649–673 (1954)
5. Chen, K.T.: Iterated path integrals. Bull. Amer. Math. Soc. 83, 831–879 (1977)
6. Fliess, M.: Fonctionnelles causales non linéaires et indéterminées non commutatives. Bull. Soc. Math. 109, 3–40 (1981)
7. Fliess, M.: Réalisation locale des systèmes non linéaires, algèbres de Lie filtrées transitives et séries génératrices. Invent. Math. 71, 521–537 (1983)
8. Feynman, R.: An Operator Calculus Having Applications in Quantum Electrodynamics. Phys. Rev. 84, 108–128 (1951)
9. van der Put, M., Singer, M.F.: Galois Theory of Linear Differential Equation. Comprehensive Studies in Mathematics, vol. 328. Springer, Berlin (2003)
10. Wechsung, G.: Functional Equations of Hyperlogarithms. In: [17]
11. Minh, H.N., Petitot, M., Van der Hoeven, J.: Polylogarithms and Shuffle Algebra. In: Proceedings of FPSAC 1998 (1998)
12. Berstel, J., Reutenauer, C.: Rational series and their languages. EATCS Monographs on Theoretical Computer Science. Springer, Heidelberg (1988)
13. Duchamp, G., Reutenauer, C.: Un critère de rationalité provenant de la géométrie noncommutative. Invent. Math. 128, 613–622 (1997)
14. Reutenauer, C.: Free Lie algebras. Oxford University Press, Oxford (1993)
15. Bourbaki, N.: Algebra, ch. III. Springer, Heidelberg (1970)
16. Bourbaki, N.: Theory of sets. Springer, Heidelberg (2004)
17. Lewin, L.: Structural properties of polylogarithms. Mathematical survey and monographs, Amer. Math. Soc. 37 (1992)
18. Bourbaki, N.: General Topology, vol. 1. Springer, Heidelberg (1970)

Quantifier Elimination over Finite Fields Using Gröbner Bases[*]

Sicun Gao, André Platzer, and Edmund M. Clarke

Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. We give an algebraic quantifier elimination algorithm for the first-order theory over any given finite field using Gröbner basis methods. The algorithm relies on the strong Nullstellensatz and properties of elimination ideals over finite fields. We analyze the theoretical complexity of the algorithm and show its application in the formal analysis of a biological controller model.

1 Introduction

We consider the problem of quantifier elimination of first-order logic formulas in the theory T_q of arithmetic in any given finite field F_q. Namely, given a quantified formula $\varphi(\boldsymbol{x}; \boldsymbol{y})$ in the language, where $\boldsymbol{x}$ is a vector of quantified variables and $\boldsymbol{y}$ a vector of free variables, we describe a procedure that outputs a quantifier-free formula $\psi(\boldsymbol{y})$, such that φ and ψ are equivalent in T_q.

Clearly, T_q admits quantifier elimination. A naive algorithm is to enumerate the exponentially many assignments to the free variables $\boldsymbol{y}$, and for each assignment $\boldsymbol{a} \in F^{|\boldsymbol{y}|}$, evaluate the truth value of the closed formula $\varphi(\boldsymbol{x}; \boldsymbol{a})$ (with a decision procedure). Then the quantifier-free formula equivalent to $\varphi(\boldsymbol{x}; \boldsymbol{y})$ is $\bigvee_{\boldsymbol{a} \in A}(\boldsymbol{y} = \boldsymbol{a})$, where $A = \{\boldsymbol{a} \in F^{|\boldsymbol{y}|} : \varphi(\boldsymbol{x}; \boldsymbol{a}) \text{ is true.}\}$. This naive algorithm *always* requires exponential time and space, and cannot be used in practice. Note that a quantifier elimination procedure is more general and complex than a decision procedure: Quantifier elimination yields an equivalent quantifier-free formula while a decision procedure outputs a yes/no answer. For instance, fully quantified formulas over finite fields can be "bit-blasted" and encoded as Quantified Boolean Formulas (QBF), whose truth value can, in principle, be determined by QBF decision procedures. However, for formulas with free variables, the use of decision procedures can only serve as an intermediate step in the naive algorithm mentioned above, and does not avoid the exponential enumeration of values for the free variables. We believe there has been no investigation into quantifier elimination procedures that can be practically used for this theory.

[*] This research was sponsored by National Science Foundation under contracts no. CNS0926181, no. CCF0541245, and no. CNS0931985, the SRC under contract no. 2005TJ1366, General Motors under contract no. GMCMUCRLNV301, Air Force (Vanderbilt University) under contract no. 18727S3, the GSRC under contract no. 1041377 (Princeton University), the Office of Naval Research under award no. N000141010188, and DARPA under contract FA8650-10-C-7077.

Such procedures are needed, for instance, in the formal verification of cipher programs involving finite field arithmetic [16,8] and polynomial dynamical systems over finite fields that arise in systems biology [11,12,4]. Take the S2VD virus competition model [11] as an example, which we study in detail in Section 6: The dynamics of the system is given by a set of polynomial equations over the field F_4. We can encode image computation and invariant analysis problems as quantified formulas, which are solvable using quantifier elimination. As is mentioned in [11], there exists no verification method suitable for such systems over general finite fields so far.

In this paper we give an algebraic quantifier elimination algorithm for T_q. The algorithm relies on strong Nullstellensatz and Gröbner basis methods. We analyze its theoretical complexity, and show its practical application.

In Section 3, we exploit the strong Nullstellensatz over finite fields and properties of elimination ideals, to show that Gröbner basis computation gives a way of eliminating quantifiers in formulas of the form $\exists \boldsymbol{x}(\bigwedge_i \alpha_i)$, where the α_is are atomic formulas and $\exists \boldsymbol{x}$ is a quantifier block. We then show, in Section 4, that the DNF-expansion of formulas can be avoided by using standard ideal operations to "flatten" the formulas. Any quantifier-free formula can be transformed into conjunctions of atomic formulas at the cost of introducing existentially quantified variables. This transformation is linear in the size of the formula, and can be seen as a generalization of the *Tseitin transformation*. Combining the techniques, we obtain a complete quantifier elimination algorithm.

In Section 5, we analyze the complexity of our algorithm, which depends on the complexity of Gröbner basis computation over finite fields. For ideals in $F_q[\boldsymbol{x}]$ that contain $x_i^q - x_i$ for each x_i, Buchberger's Algorithm computes Gröbner bases within exponential time and space [13]. Using this result, the worst-case time/space complexity of our algorithm is bounded by $q^{O(|\varphi|)}$ when φ contains no more than two *alternating blocks* of quantifiers, and $q^{q^{O(|\varphi|)}}$ for more alternations. Recently a polynomial-space algorithm for Gröbner basis computation over finite fields has been proposed in [17], but it remains theoretical so far. If the new algorithm can be practically used, the worst-case complexity of quantifier elimination is $q^{O(|\varphi|)}$ for arbitrary alternations.

Note that this seemingly high worst-case complexity, as is common for Gröbner basis methods, does not prevent the algorithm from being useful on practical problems. This is crucially different from the naive algorithm, which always requires exponential cost, not just in worst cases. In Section 6, we show how the algorithm is successfully applied in the analysis of a controller design in the S2VD virus competition model [11], which is a polynomial dynamical system over finite fields. The authors developed control strategies to ensure a safety property in the model, and used simulations to conclude that the controller is effective. However, using the quantifier elimination algorithm, we found bugs that show inconsistency between specifications of the system and its formal model. This shows how our algorithm can provide a practical way of extending formal verification techniques to models over finite fields.

Throughout the paper, omitted proofs are provided in the Appendix.

2 Preliminaries

2.1 Ideals, Varieties, Nullstellensatz, and Gröbner Bases

Let k be any field and $k[x_1, ..., x_n]$ the polynomial ring over k with indeterminates $x_1, ..., x_n$. An *ideal* generated by $f_1, ..., f_m \in k[x_1, ..., x_n]$ is $\langle f_1, ..., f_m \rangle = \{h : h = \sum_{i=1}^{m} g_i f_i,\ g_i \in k[x_1, ..., x_n]\}$. Let $\boldsymbol{a} \in k^n$ be an arbitrary point, and $f \in k[x_1, ..., x_n]$ be a polynomial. We say that f *vanishes* on $\boldsymbol{a}$ if $f(\boldsymbol{a}) = 0$.

Definition 2.1. *For any subset J of $k[x_1, ..., x_n]$, the* ***affine variety*** *of J over k is $V_n(J) = \{\boldsymbol{a} \in k^n : \forall f \in J, f(\boldsymbol{a}) = 0\}$.*

Definition 2.2. *For any subset V of k^n, the* ***vanishing ideal*** *of V is defined as $I(V) = \{f \in k[x_1, ..., x_n] : \forall \boldsymbol{a} \in V, f(\boldsymbol{a}) = 0\}$.*

Definition 2.3. *Let J be any ideal in $k[x_1, ..., x_n]$, the* ***radical*** *of J is defined as $\sqrt{J} = \{f \in k[x_1, ..., x_n] : \exists m \in \mathbb{N}, f^m \in J\}$.*

When $J = \sqrt{J}$, we say J is a radical ideal. The celebrated Hilbert Nullstellensatz established the correspondence between radical ideals and varieties:

Theorem 2.1 (Strong Nullstellensatz [14]). *For an arbitrary field k, let J be an ideal in $k[x_1, ..., x_n]$. We have $I(V^a(J)) = \sqrt{J}$, where k^a is the algebraic closure of k and $V^a(J) = \{\boldsymbol{a} \in (k^a)^n : \forall f \in J, f(\boldsymbol{a}) = 0\}$.*

The method of Gröbner bases was introduced by Buchberger [6] for the algorithmic solution of various fundamental problems in commutative algebra. For an ideal $\langle f_1, ..., f_m \rangle$ in a polynomial ring, Gröbner basis computation transforms $f_1, ..., f_m$ to a canonical representation $\langle g_1, ..., g_s \rangle = \langle f_1, ..., f_m \rangle$ that has many useful properties. Detailed treatment of the theory can be found in [3].

Definition 2.4. *Let $T = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} : \alpha_i \in N\}$ be the set of monomials in $k[x_1, ..., x_n]$. A* ***monomial ordering*** *$\prec$ on T is a well-ordering on T satisfying*
(1) For any $t \in T$, $1 \prec t$
(2) For all $t_1, t_2, s \in T$, $t_1 \prec t_2$ then $t_1 \cdot s \prec t_2 \cdot s$.

We order the monomials appearing in any single polynomial $f \in k[x_1, ..., x_n]$ with respect to $\prec$. We write $LM(f)$ to denote the *leading monomial* in f (the maximal monomial under $\prec$), and $LT(f)$ to denote the *leading term* of f ($LM(f)$ multiplied by its coefficient). We write $LM(S) = \{LM(f) : f \in S\}$ where S is a set of polynomials.

Let J be an ideal in $k[x_1, ..., x_n]$. Fix any monomial order on T. The ideal of leading monomials of J, $\langle LM(J) \rangle$, is the ideal generated by the leading monomials of all polynomials in J. Now we are ready to define:

Definition 2.5 (Gröbner Basis [3]). *A* ***Gröbner basis*** *for J is a set $GB(J) = \{g_1, ..., g_s\} \subseteq J$ satisfying $\langle LM(GB(J)) \rangle = \langle LM(J) \rangle$.*

2.2 The First-Order Theory over a Finite Field

Let F_q be an arbitrary finite field of size q, where q is a prime power. We fix the structure to be $M_q = \langle F_q, 0, 1, +, \times\rangle$ and the signature $\mathcal{L}_q = \langle 0, 1, +, \times\rangle$ ("=" is a logical predicate). For quantified formulas, we write $\varphi(\boldsymbol{x}; \boldsymbol{y})$ to emphasize that the $\boldsymbol{x}$ is a vector of quantified variables and $\boldsymbol{y}$ is a vector of free variables.

The standard first-order theory for each M_q consists of the usual axioms for fields [15] plus $\exists x_1 \cdots \exists x_q((\bigwedge_{1\le i<j\le q} x_i \neq x_j) \wedge \forall y(\bigvee_i y = x_i))$, which fixes the size of the domain. We write this theory as T_q. In $\mathcal{L}_q$, we consider all the atomic formulas as polynomial equations $f = 0$. The *realization* of a formula is the set of assignments to its free variables that makes the formula true over M_q. Formally:

Definition 2.6 (Realization). *Let $\varphi(x_1, ..., x_n)$ be a formula with free variables $\boldsymbol{x} = (x_1, ..., x_n)$. The realization of φ, written as $[\![\varphi]\!] \subseteq F_q^n$, is inductively defined as:*

- $[\![p = 0]\!] =_{df} V(\langle p\rangle) \subseteq F_q^n$ *(in particular, $[\![\top]\!] = F_q^n$)*
- $[\![\neg\psi]\!] = F_q^n \setminus [\![\psi]\!]$
- $[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]$
- $[\![\exists x_0.\psi(x_0, \boldsymbol{x})]\!] = \{\langle a_1, ..., a_n\rangle \in F_q^n : \exists a_0 \in F_q,$ *such that* $\langle a_0, ..., a_n\rangle \in [\![\psi]\!]\}$

Proposition 2.1 (Fermat's Little Theorem). *Let F_q be a finite field. For any $a \in F_q$, we have $a^q - a = 0$. Conversely, $V(x^q - x) = [\![x^q - x]\!] = F_q$.*

Definition 2.7 (Quantifier Elimination). *T_q admits quantifier elimination if for any formula $\varphi(\boldsymbol{x}; \boldsymbol{y})$, where the $\boldsymbol{x}$ variables are quantified and the $\boldsymbol{y}$ variables free, there exists a quantifier-free formula $\psi(\boldsymbol{y})$ such that $[\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = [\![\psi(\boldsymbol{y})]\!]$.*

2.3 Nullstellensatz in Finite Fields

The strong Nullstellensatz admits a special form over finite fields. This was proved for prime fields in [10] and used in [4,5]. Here we give a short proof that the special form holds over arbitrary finite fields, as a corollary of Theorem 2.1.

Lemma 2.1. *For any ideal $J \subseteq F_q[x_1, ..., x_n]$, $J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$ is radical.*

Theorem 2.2 (Strong Nullstellensatz in Finite Fields). *For an arbitrary finite field F_q, let $J \subseteq F_q[x_1, ..., x_n]$ be an ideal, then*

$$I(V(J)) = J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle.$$

Proof. Apply Theorem 2.1 to $J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$ and use Lemma 2.1. We have $I(V^a(J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle)) = J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$. But since $V^a(\langle x_1^q - x_1, ..., x_n^q - x_n\rangle) = F_q^n$, it follows that

$$V^a(J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle) = V^a(J) \cap F_q^n = V(J).$$

Thus we obtain $I(V(J)) = J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$. $\square$

3 Quantifier Elimination Using Gröbner Bases

In this section, we show that the key step in quantifier elimination can be realized by Gröbner basis computation. Namely, for any formula φ of the form $\exists \boldsymbol{x} \bigwedge_{i=1}^{r} f_i(\boldsymbol{x}, \boldsymbol{y}) = 0$, we can compute a quantifier-free formula $\psi(\boldsymbol{y})$ such that $[\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = [\![\psi(\boldsymbol{y})]\!]$. We use the following notational conventions:

- $|\boldsymbol{x}| = n$ is the number of quantified variables and $|\boldsymbol{y}| = m$ the number of free variables. We write $\boldsymbol{x}^q - \boldsymbol{x} =_{df} \{x_1^q - x_1, ..., x_n^q - x_n\}$ and $\boldsymbol{y}^q - \boldsymbol{y} =_{df} \{y_1^q - y_1, ..., y_m^q - y_m\}$, and call them *field polynomials* (following [10]).
- We use $\boldsymbol{a} = (a_1, ..., a_n) \in F_q^n$ to denote the assignment for the $\boldsymbol{x}$ variables, and $\boldsymbol{b} = (b_1, ..., b_m) \in F_q^m$ for the $\boldsymbol{y}$ variables. $(\boldsymbol{a}, \boldsymbol{b}) \in F_q^{n+m}$ is a complete assignment for all the variables in φ.
- When we write $J \subseteq F_q[\boldsymbol{x}, \boldsymbol{y}]$ or a formula $\varphi(\boldsymbol{x}; \boldsymbol{y})$, we assume that all the $\boldsymbol{x}, \boldsymbol{y}$ variables do occur in J or φ. We assume that the $\boldsymbol{x}$ variables always rank higher than the $\boldsymbol{y}$ variables in the lexicographic order.

3.1 Existential Quantification and Elimination Ideals

First, we show that eliminating the $\boldsymbol{x}$ variables is equivalent to *projecting* the variety $V(\langle f_1, ..., f_r\rangle)$ from F_q^{n+m} to F_q^m.

Lemma 3.1. *For $f_1, ..., f_r \in F_q[\boldsymbol{x}, \boldsymbol{y}]$, we have $[\![\bigwedge_{i=1}^{r} f_i = 0]\!] = V(\langle f_1, ..., f_r\rangle)$.*

Definition 3.1 (Projection). *The l-th projection mapping is defined as:*

$$\pi_l : F_q^N \to F_q^{N-l}, \pi_l((c_1, ..., c_N)) = (c_{l+1}, ..., c_N)$$

where $l < N$. For any set $A \subseteq F_q^N$, we write $\pi_l(A) = \{\pi_i(\boldsymbol{c}) : \boldsymbol{c} \in A\} \subseteq F_q^{N-l}$.

Lemma 3.2. $[\![\exists \boldsymbol{x} \varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = \pi_n([\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!])$.

Next, we show that the projection π_n of the variety $V_{n+m}(\langle f_1, ..., f_r\rangle)$ from F_q^{n+m} to F_q^m, is exactly the variety $V_m(\langle f_1, ..., f_r\rangle \cap F_q[\boldsymbol{y}])$.

Definition 3.2 (Elimination Ideal [7]). *Let $J \subseteq F_q[x_1, ..., x_n]$ be an ideal. The l-th **elimination ideal** J_l, for $1 \leq l \leq N$, is the ideal of $F_q[x_{l+1}, ..., x_N]$ defined by $J_l = J \cap F_q[x_{l+1}, ..., x_N]$.*

The following lemma shows that adding field polynomials does not change the realization. For $f_1, ..., f_r \in F_q[\boldsymbol{x}, \boldsymbol{y}]$, we have:

Lemma 3.3. $[\![\bigwedge_{i=1}^{r} f_i = 0]\!] = [\![\bigwedge_{i=1}^{r} f_i = 0 \wedge \bigwedge(x_i^q - x_i = 0) \wedge \bigwedge(y_i^q - y_i = 0)]\!]$.

Now we can prove the key equivalence between projection operations and elimination ideals. This requires the use of Nullstellensatz for finite fields.

Theorem 3.1. *Let $J \subseteq F_q[\boldsymbol{x}, \boldsymbol{y}]$ be an ideal which contains the field polynomials for all the variables in J. We have $\pi_n(V(J)) = V(J_n)$.*

Proof. We show inclusion in both directions.

– $\pi_n(V(J)) \subseteq V(J_n)$:
For any $\boldsymbol{b} \in \pi_n(V(J))$, there exists $\boldsymbol{a} \in F_q^n$ such that $(\boldsymbol{a}, \boldsymbol{b}) \in V(J)$. That is, $(\boldsymbol{a}, \boldsymbol{b})$ satisfies all polynomials in J; in particular, $\boldsymbol{b}$ satisfies all polynomials in J that only contain the $\boldsymbol{y}$ variables ($\boldsymbol{a}$ is not assigned to variables). Thus, $\boldsymbol{b} \in V(J \cap F_q[\boldsymbol{y}]) = V(J_n)$.
– $V(J_n) \subseteq \pi_n(V(J))$:
Let $\boldsymbol{b}$ be a point in F_q^m such that $\boldsymbol{b} \notin \pi_n(V(J))$. Consider the polynomial

$$f_{\boldsymbol{b}} = \prod_{i=1}^{m} (\prod_{c \in F_q \setminus \{b_i\}} (y_i - c)).$$

$f_{\boldsymbol{b}}$ vanishes on all the points in F_q^n, except $\boldsymbol{b} = (b_1, ..., b_m)$, since $(y_i - b_i)$ is excluded in the product for all i. In particular, $f_{\boldsymbol{b}}$ vanishes on all the points in $V(J)$, because for each $(\boldsymbol{a}, \boldsymbol{b}') \in V(J)$, $\boldsymbol{b}'$ must be different from $\boldsymbol{b}$, and $f_{\boldsymbol{b}}(\boldsymbol{a}, \boldsymbol{b}') = f_{\boldsymbol{b}}(\boldsymbol{b}') = 0$ (since there are no $\boldsymbol{x}$ variables). Therefore, $f_{\boldsymbol{b}}$ is contained in the vanishing ideal of $V(J)$, i.e., $f_{\boldsymbol{b}} \in I(V(J))$.

Now, Theorem 2.2 shows $I(V(J)) = J + \langle \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle$. Since J already contains the field polynomials, we know $J + \langle \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle = J$, and consequently $I(V(J)) = J$. Since $f_{\boldsymbol{b}} \in I(V(J))$, we must have $f_{\boldsymbol{b}} \in J$. But on the other hand, $f_{\boldsymbol{b}} \in F_q[\boldsymbol{y}]$. Hence $f_{\boldsymbol{b}} \in J \cap F_q[\boldsymbol{y}] = J_n$. But since $f_{\boldsymbol{b}}(\boldsymbol{b}) \neq 0$, we know $\boldsymbol{b} \notin V(J_n)$. □

3.2 Quantifier Elimination Using Elimination Ideals

Theorem 3.1 shows that to obtain the projection of a variety over F_q, we only need to take the variety of the corresponding elimination ideal. In fact, this can be easily done using the Gröbner basis of the original ideal:

Proposition 3.1 (cf. [7]). *Let $J \subseteq F_q[x_1, ..., x_N]$ be an ideal and let G be the Gröbner basis of J with respect to the lexicographic order $x_1 \succ \cdots \succ x_N$. Then for every $1 \leq l \leq N$, $G \cap F_q[x_{l+1}, ..., x_N]$ is a Gröbner basis of the l-th elimination ideal J_l. That is, $J_l = \langle G \rangle \cap F_q[x_{l+1}, ..., x_N] = \langle G \cap F_q[x_{l+1}, ..., x_N] \rangle$.*

Now, putting all the lemmas together, we arrive at the following theorem:

Theorem 3.2. *Let $\varphi(\boldsymbol{x}; \boldsymbol{y})$ be $\exists \boldsymbol{x}.(\bigwedge_{i=1}^{r} f_i = 0)$ be a formula in $\mathcal{L}_q$, with $f_i \in F_q[\boldsymbol{x}, \boldsymbol{y}]$. Let G be the Gröbner basis of $\langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle$. Suppose $G \cap F_q[\boldsymbol{y}] = \{g_1, ..., g_s\}$, then we have $[\![\varphi]\!] = [\![\bigwedge_{i=1}^{s} (g_i = 0)]\!]$.*

Proof. We write $J = \langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle$ for convenience. First, by Lemma 3.3, adding the polynomials $\boldsymbol{x}^q - \boldsymbol{x}$ and $\boldsymbol{y}^q - \boldsymbol{y}$ does not change the realization:

$$[\![\varphi]\!] = [\![\exists \boldsymbol{x}.(\bigwedge_{i=1}^{r} f_i = 0)]\!] = [\![\exists \boldsymbol{x}.(\bigwedge_{i=1}^{r} f_i = 0 \wedge \bigwedge_{i=1}^{n} (x_i^q - x_i = 0) \wedge \bigwedge_{i=1}^{m} (y_i^q - y_i = 0))]\!]$$

Next, by Lemma 3.2, the quantification on $\boldsymbol{x}$ corresponds to projecting a variety:

$$[\![\exists \boldsymbol{x}.(\bigwedge_{i=1}^{r} f_i = 0 \wedge \bigwedge_{i=1}^{n}(x_i^q - x_i = 0) \wedge \bigwedge_{i=1}^{m}(y_i^q - y_i = 0))]\!] = \pi_n(V(J)).$$

Using Theorem 3.1, we know that the projection of a variety is equivalent to the variety of the corresponding elimination ideal, i.e., $\pi_n(V(J)) = V(J \cap F_q[\boldsymbol{y}])$. Now, using the property of Gröbner bases in Proposition 3.1, we know the elimination ideal $\langle G \rangle \cap F_q[\boldsymbol{y}]$ is generated by $G \cap F_q[\boldsymbol{y}]$:

$$V(J \cap F_q[\boldsymbol{y}]) = V(\langle G \rangle \cap F_q[\boldsymbol{y}]) = V(\langle G \cap F_q[\boldsymbol{y}] \rangle) = V(\langle g_1, ..., g_s \rangle)$$

Finally, by Lemma 3.1, an ideal is equivalent to the conjunction of atomic formulas given by the generators of the ideal: $V(\langle g_1, ..., g_s \rangle) = [\![\bigwedge_{i=1}^{s} g_i = 0]\!]$.

Connecting all the equations above, we have shown $[\![\varphi]\!] = [\![\bigwedge_{i=1}^{s} g_i = 0]\!]$. Note that $g_1, ..., g_s \in F_q[\boldsymbol{y}]$ (they do not contain $\boldsymbol{x}$ variables). □

4 Formula Flattening with Ideal Operations

If negations on atomic formulas can be eliminated (to be shown in Lemma 4.1), Theorem 3.2 already gives a direct quantifier elimination algorithm. That is, we can always use duality to make the innermost quantifier block an existential one, and expand the quantifier-free part to DNF. Then the existential block can be distributed over the disjuncts and Theorem 3.2 is applied. However, this direct algorithm *always* requires exponential blow-up in expanding formulas into DNF.

We show that the DNF-expansion can be avoided: Any quantifier-free formula can be transformed into an equivalent formula of the form $\exists \boldsymbol{z}.(\bigwedge_{i=1}^{r} f_i = 0)$, where $\boldsymbol{z}$ are new variables and f_is are polynomials. The key is that Boolean conjunctions and disjunctions can both be turned into additions of ideals; in the latter case new variables need be introduced. This transformation can be done in linear time and space, and is a generalization of the *Tseitin transformation* from F_2 to general finite fields.

We use the usual definition of ideal addition and multiplication. Let $J_1 = \langle f_1, ..., f_r \rangle$ and $J_2 = \langle g_1, ..., g_s \rangle$ be ideals, and h be a polynomial. Then $J_1 + J_2 = \langle f_1, ..., f, g_1, ..., g_s \rangle$ and $J_1 \cdot h = \langle f_1 \cdot h, ..., f_r \cdot h \rangle$.

Lemma 4.1 (Elimination of Negations). *Suppose φ is a quantifier free formula in $\mathcal{L}_q$ in NNF and contains k negative atomic formulas. Then there is a formula $\exists \boldsymbol{z}.\psi$, where ψ contains new variables $\boldsymbol{z}$ but no negative atoms, such that $[\![\varphi]\!] = [\![\exists \boldsymbol{z}.\psi]\!]$.*

Lemma 4.2 (Elimination of Disjunctions). *Suppose ψ_1 and ψ_2 are two formulas in variables $x_1, ..., x_n$, and J_1 and J_2 are ideals in $F_q[x_1, ..., x_n]$ satisfying $[\![\psi_1]\!] = V(J_1)$ and $[\![\psi_2]\!] = V(J_2)$. Then, using x_0 as a new variable, we have $[\![\psi_1 \vee \psi_2]\!] = V(J_1) \cup V(J_2) = \pi_0(V(x_0 J_1 + (1 - x_0)J_2))$.*

Theorem 4.1. *For any quantifier-free formula $\varphi(\boldsymbol{x})$ given in NNF, there exists a formula ψ of the form $\exists \boldsymbol{u}, \boldsymbol{v}(\bigwedge_i (f_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = 0))$ such that $[\![\varphi]\!] = [\![\psi]\!]$. Furthermore, ψ can be generated in time $O(|\varphi|)$, and also $|\boldsymbol{u}| + |\boldsymbol{v}| = O(|\varphi|)$.*

Proof. Since $\varphi(\boldsymbol{x})$ is in NNF, all the negations occur in front of atomic formulas. We first use Lemma 4.1 to eliminate the negations. Suppose there are k negative atomic formulas in φ, we obtain $[\![\varphi]\!] = [\![\exists u_1, ..., u_k.\varphi']\!]$. Now φ' does not contain negations.

We then prove that there exists an ideal $J_{\varphi'}$ for φ' satisfying $\pi_{|\boldsymbol{v}|}(V(J_{\varphi'})) = [\![\varphi']\!]$, where $\boldsymbol{v}$ are the introduced variables (which rank higher than the existing variables in the variable ordering, so that the projection $\pi_{|\boldsymbol{v}|}$ truncates assignments on the $\boldsymbol{v}$ variables).

- If φ' is an atomic formula $f = 0$, then $J_{\varphi'} = \langle f \rangle$;
- If φ' is of the form $\theta_1 \wedge \theta_2$, then $J_{\varphi'} = J_{\theta_1} + J_{\theta_2}$;
- If φ' is of the form $\theta_1 \vee \theta_2$, then $J_{\varphi'} = v_i \cdot J_{\theta_1} + (1 - v_i) \cdot J_{\theta_2}$, where v_i is new.

Note that the new variables are only introduced in the disjunction case, and therefore the number of $\boldsymbol{v}$ variables equals the number of disjunctions. Following Lemma 3.1 and 4.2, the transformation preserves the realization of the formula in each case. Hence, we have $\pi_{\boldsymbol{v}}(V(J_{\varphi'})) = [\![\varphi']\!]$. Writing $J_{\varphi'} = \langle f_1, ..., f_r \rangle$, we know $[\![\varphi]\!] = [\![\exists \boldsymbol{u}.\varphi']\!] = [\![\exists \boldsymbol{u} \exists \boldsymbol{v}. \bigwedge_{i=1}^r f_i]\!]$. Notice that the number of rewriting steps is bounded by the number of logical symbols appearing in φ. Hence the transformation is done in time linear in the size of the formula. The number of new variables is equal to the number of negations and disjunctions. □

5 Algorithm Description and Complexity Analysis

We now describe the full algorithm using the following notations:

- The input formula is given by $\varphi = Q_1 \boldsymbol{x}_1 \cdots Q_m \boldsymbol{x}_m \psi$. Each $Q_i \boldsymbol{x}_i$ represents a *quantifier block*, where Q_i is either $\exists$ or $\forall$. Q_i and Q_{i+1} are different quantifiers. We write $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$. ψ is a quantifier-free formula in $\boldsymbol{x}$ and $\boldsymbol{y}$ given in NNF, where $\boldsymbol{y}$ are free variables.
- We assume the innermost quantifier is existential, $Q_m = \exists$. (Otherwise we apply quantifier elimination on the negation of the formula.)

5.1 Algorithm Description

Section 3 shows how to eliminate existential quantifiers over conjunctions of positive atomic formulas. Section 4 shows how formulas can be put into conjunctions of positive atoms with new quantified variables. It follows that we can always eliminate the innermost existential quantifiers, and iterate the process by flipping the universal quantifiers into existential ones. We first emphasize some special features of the algorithm:

Algorithm 1. Quantifier Elimination for $\varphi = Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m.\psi$

```
1: Input: $\varphi = Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m.\psi(\boldsymbol{x}_1, ..., \boldsymbol{x}_m, \boldsymbol{y})$ where $m$ is the number of quantifier alternations, $Q_m x_m$ is an existential block ($Q_m = \exists$), and $\psi$ is in negation normal form.
2: Output: A quantifier-free equivalent formula of $\varphi$
3: Procedure QE($\varphi$)
4: while $m \geq 1$ do
5:    $\exists \boldsymbol{u}.\psi' \leftarrow$ Eliminate_Negations($\psi$)
6:    $\exists \boldsymbol{v}.(f_1 = 0 \wedge \cdots \wedge f_r = 0) \leftarrow$ Formula_Flattening($\psi'$)
7:    $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m \exists \boldsymbol{u} \exists \boldsymbol{v}.(f_1 = 0 \wedge \cdots \wedge f_r = 0)$
8:    $\{g_1, ..., g_s\} =$ Gröbner_Basis($\langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{u}^q - \boldsymbol{u}, \boldsymbol{v}^q - \boldsymbol{v}\rangle$)
9:    if $m = 1$ then
10:      $\varphi \leftarrow g_1 = 0 \wedge \cdots \wedge g_s = 0$
11:      break
12:   end if
13:   $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2} Q_{m-1}\boldsymbol{x}_{m-1}.(\bigwedge_{i=1}^{s} g_i = 0)$ where $Q_{m-1} = \forall$
14:   $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}.(\bigwedge_{i=1}^{s} \neg\exists \boldsymbol{x}_{m-1}(g_i \neq 0))$
15:   for $i = 1$ to $s$ do
16:      $\bigwedge_{j=1}^{t_i} h_{ij} = 0 \leftarrow$ QE($\exists \boldsymbol{x}_{m-1}(g_i \neq 0)$)
17:   end for
18:   $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2} \bigwedge_{i=1}^{s}(\bigvee_{j=1}^{t_i} h_{ij} \neq 0)$
19:   $m \leftarrow m - 2$
20: end while
21: return $\varphi$
```

– In each elimination step, a full *quantifier block* is eliminated. This is desirable in practical problems, which usually contain many variables but few alternating quantifier blocks. For instance, many verification problems are expressible using two blocks of quantifiers ($\forall\exists$-formulas).
– The quantifier elimination step essentially transforms an ideal to another ideal. This corresponds to transforming conjunctions of atomic formulas to conjunctions of new atomic formulas. Therefore, the quantifier elimination steps do not introduce new nesting of Boolean operators.
– The algorithm always directly outputs CNF formulas.

A formal description of the full algorithm is given in Algorithm 1. The main steps in the algorithm are explained below. Each loop of the algorithm contains three main steps. In Step 1, φ is flattened; in Step 2, the innermost existential quantifier block is eliminated; in Step 3, the next (universal) quantifier block is eliminated and the process loops back to Step 1. The algorithm terminates either after Step 2 or Step 3, when there are no remaining quantifiers to be eliminated.

• Step 1: (Line 5-7)
First, since ψ is in NNF, we use Theorem 4.1 to eliminate the negations and disjunctions in ψ to get $[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m \exists \boldsymbol{u} \exists \boldsymbol{v}.(\bigwedge_{i=1}^{r} f_i = 0)]\!]$, where $\boldsymbol{u}$

are the variables introduced for eliminating negations (Lemma 4.1), and $\boldsymbol{v}$ are the variables introduced for eliminating disjunctions (Lemma 4.2).

• **Step 2: (Line 8-12)** Since $Q_m = \exists$, using Theorem 4.1, we can eliminate the variables $\boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{v}$ simultaneously by computing

$$\{g_1, ..., g_{r_1}\} = GB(\langle f_1, ..., f_r, \boldsymbol{x}_m^q - \boldsymbol{x}_m, \boldsymbol{u}^q - \boldsymbol{u}, \boldsymbol{v}^q - \boldsymbol{v}, \boldsymbol{y}^q - \boldsymbol{y}\rangle) \cap F_q[\boldsymbol{x}_1, ..., \boldsymbol{x}_{m-1}, \boldsymbol{y}].$$

Now we have $[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_{m-1}\boldsymbol{x}_{m-1}.(\bigwedge_{i=1}^s (g_i = 0))]\!]$.
If there are no more quantifiers, the output is $\bigwedge_{i=1}^s (g_i = 0)$, which is in CNF.

• **Step 3: (Line 13-18)** Since $Q_{m-1} = \forall$, we distribute the block $Q_{m-1}\boldsymbol{x}_{m-1}$ over the conjuncts:

$$[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}(\bigwedge_{i=1}^{s}(\neg\exists \boldsymbol{x}_{m-1}\neg(g_i = 0)))]\!]$$

Now we do elimination recursively on $\exists \boldsymbol{x}_{m-1}(\neg g_i = 0)$ for each $i \in \{1, ..., s\}$, which can be done using only Step 1 and Step 2. We obtain:

$$[\![\exists \boldsymbol{x}_{m-1}(\neg g_i = 0)]\!] = [\![\exists \boldsymbol{x}_{m-1}\exists u'.(g_i \cdot u' - 1 = 0)]\!] = [\![\bigwedge_{j=1}^{t_i} h_{ij} = 0]\!] \tag{1}$$

and the formula becomes (note that the extra negation is distributed)

$$[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}.(\bigwedge_{i=1}^{s}(\bigvee_{j=1}^{t_i} h_{ij} \neq 0))]\!]. \tag{2}$$

If there are no more quantifiers left, the output formula is $\bigwedge_{i=1}^s(\bigvee_{j=1}^{t_i} h_{ij} \neq 0)$, which is in CNF. Otherwise, $Q_{m-2} = \exists$, and we return to Step 1.

Theorem 5.1 (Correctness). *Let $\varphi(\boldsymbol{x};\boldsymbol{y})$ be a formula $Q_1\boldsymbol{x}_i \cdots Q_m\boldsymbol{x}_m.\psi$ where $Q_m = \exists$ and ψ is in NNF. Algorithm 1 computes a quantifier-free formula $\varphi'(\boldsymbol{y})$, such that $[\![\varphi(\boldsymbol{x};\boldsymbol{y})]\!] = [\![\varphi'(\boldsymbol{y})]\!]$ and φ' is in CNF.*

5.2 Complexity Analysis

The worst-case complexity of Gröbner basis computation on ideals in $F_q[\boldsymbol{x}]$ that contain $x_i^q - x_i$ for each variable x_i is known to be single exponential in the number of variables in time and space. This follows from the complexity result for Gröbner basis computation of zero-dimensional radical ideals [13] (a direct proof can be found in [9]).

Proposition 5.1. *Let $J = \langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}\rangle \subseteq F_q[x_1, ..., x_n]$ be an ideal. The time and space complexity of Buchberger's Algorithm is bounded by $q^{O(n)}$, assuming that the length of input $(f_1, ..., f_r)$ is dominated by $q^{O(n)}$.*

Now we are ready to estimate the complexity of our algorithm.

Theorem 5.2 (Complexity). *Let φ be the input formula with m quantifier blocks. When $m \leq 2$, the time/space complexity of Algorithm 1 is bounded by $q^{O(|\varphi|)}$. Otherwise, it is bounded by $q^{q^{O(|\varphi|)}}$.*

Proof. The complexity is dominated by Gröbner basis computation, whose complexity is determined by the number of variables occurring in the ideal. When $m \leq 2$, the main loop is executed once, and the number of newly introduced variables is bounded by the original length of the input formula. Therefore, Gröbner basis computations can be done in single exponential time/space. When $m > 2$, the number of newly introduced variables is bounded by the length of the formula obtained from the previous run of the main loop, which can itself be exponential in the number of the remaining variables. In that case, Gröbner basis computation can take double exponential time/space.

• **Case $m \leq 2$:**
In Step 1, the number of the introduced $\boldsymbol{u}$ and $\boldsymbol{v}$ variables equals to the number of negations and disjunctions that appear in the φ. Hence the total number of variables is bounded by the length of φ. The flattening takes linear time and space, $O(|\varphi|)$, as proved in Theorem 4.1.

In Step 2, by Proposition 5.1, Gröbner basis computation takes time/space $q^{O(|\varphi|)}$.

In Step 3, the variables $\boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{v}$ have all been eliminated. The length of each $g_i u' - 1$ (see Formula (1) in Step 3) is bounded by the number of monomials consisting of the remaining variables, which is $O(q^{(|\boldsymbol{y}|+\sum_{i=1}^{m-1}|\boldsymbol{x}_i|)})$ (because the degree on each variable is lower than q). Following Proposition 5.1, Gröbner basis computation on each $g_i u' - 1$ takes time and space $q^{O(|\boldsymbol{y}|+\sum_{i=1}^{m-1}|\boldsymbol{x}_i|)}$, which is dominated by $q^{O(|\varphi|)}$. Also, since the number s of conjuncts is the number of polynomials in the Gröbner basis computed in the previous step, we know s is bounded by $q^{O(|\varphi|)}$. In sum, Step 3 takes $q^{O(|\varphi|)}$ time/space in worst case.

Thus, the algorithm has worst-case time and space complexity $q^{O(|\varphi|)}$ when $m \leq 2$.

• **Case $m > 2$:**
When $m > 2$, the main loop is iterated for more than one round. The key change in the second round is that, the initial number of conjunctions and disjunctions in each conjunct could both be exponential in the number of the remaining variables $(\boldsymbol{x}_1, ..., \boldsymbol{x}_{m-2})$. That means, writing the max of t_i as t (see Formula (2) in Step 3), both s and t can be of order $q^{O(|\varphi|)}$.

In Step 1 of the second round, the number of the $\boldsymbol{u}$ variables introduced for eliminating the negations is $s \cdot t$. The number of the $\boldsymbol{v}$ variables introduced for eliminating disjunctions is also $s \cdot t$. Hence the flattened formula may now contain $q^{O(|\varphi|)}$ variables.

In Step 2 of the second round, Gröbner basis computation takes time and space exponential in the number of variables. Therefore, Step 2 can now take $q^{q^{O(|\varphi|)}}$ in time and space.

In Step 3 of the second round, however, the number of conjuncts s *does not* become doubly exponential. This is because g_i in Step 3 no longer contains the exponentially many introduced variables – they were already eliminated in the previous step. Thus s is reduced back to single exponential in the number of the remaining variables; i.e., it is bounded by $q^{O(|\varphi|)}$. Similarly, the Gröbner basis computation on each $g_i u' - 1$, which now contains variables $\boldsymbol{x}_1, ..., \boldsymbol{x}_{m-1}, \boldsymbol{y}$, takes time and space $q^{O(|\varphi|)}$. In all, Step 3 takes time and space $q^{O(|\varphi|)}$.

In sum, the second round of the main loop can take time/space $q^{q^{O(|\varphi|)}}$. But at the end of the loop, the size of formula is reduced to $q^{O(|\varphi|)}$ after the Gröbner basis computations, because it is at most single exponential in the number of the remaining variables. Therefore, the double exponential bound remains for future iterations of the main loop. □

Recently, [17] reports a Gröbner basis computation algorithm in finite fields using polynomial space. This algorithm is theoretical and cannot be applied yet. Given the analysis above, if such a polynomial-space algorithm for Gröbner basis computation can be practically used, the intermediate expressions do not have the double-exponential blow-up. On the other hand, it does not lower the space bound of our algorithm to polynomial space, because during flattening of the disjunctions, the introduced terms are multiplied together. To expand the introduced terms, one may still use exponential space. It remains further work to investigate whether the algorithm can be practically used and how it compares with Buchberger's Algorithm.

Proposition 5.2. *If there exists a polynomial-space Gröbner basis computation algorithm over finite fields for ideals containing the field polynomials, the time/space complexity of our algorithm is bounded by* $q^{O(|\varphi|)}$.

6 Example and Application

6.1 A Walk-Through Example

Consider the following formula over F_3:

$$\varphi : \exists b \forall a \exists y \exists x.((y = ax^2 + bx + c) \wedge (y = ax))$$

which has three alternating quantifier blocks and one free variable. We ask for a quantifier-free formula $\psi(c)$ equivalent to φ.

We fix the lexicographic ordering to be $x \succ y \succ a \succ b \succ c$. First, we compute the Gröbner basis G_0 of the ideal: $\langle y - ax^2 - bx - c, y - ax, x^3 - x, y^3 - y, a^3 - a, b^3 - b, c^3 - c\rangle$,and obtain the Gröbner basis of the elimination ideal

$$G_1 = G_0 \cap F_3[a, b, c] = \{abc + ac^2 + b^2c - c, a^3 - a, b^3 - b, c^3 - c\}.$$

After this, x and y have been eliminated, and we have:

$$\begin{aligned} [\![\varphi]\!] &= [\![\exists b \forall a.((abc + ac^2 + b^2c - c = 0) \wedge (a^3 - a = 0) \wedge (b^3 - b = 0) \wedge (c^3 - c = 0))]\!] \\ &= [\![\exists b \forall a.(abc + ac^2 + b^2c - c = 0)]\!] \\ &= [\![\exists b.(\neg \exists a \exists u.(u(abc + ac^2 + b^2c - c) - 1 = 0))]\!] \end{aligned}$$

Now we eliminate quantifiers in $\exists a \exists u((abc + ac^2 + b^2c - c) \cdot u - 1 = 0)$, again by computing the Gröbner basis G_2 of the ideal

$$\langle (abc + ac^2 + b^2c - c)u - 1, a^3 - a, b^3 - b, c^3 - c, u^3 - u \rangle \cap F_3[b, c].$$

We obtain $G_2 = \{b^2 - bc, c^2 - 1\}$. Therefore $[\![\varphi]\!] = [\![\exists b(\neg(b^2 - bc = 0 \wedge c^2 - 1 = 0))]\!]$. (Note that if both b and c are both free variables, $b^2 - bc \neq 0 \vee c^2 - 1 \neq 0$ would be the quantifier-free formula containing b, c that is equivalent to φ.)

Next, we introduce u_1 and u_2 to eliminate the negations, and v to eliminate the disjunction:

$$[\![\varphi]\!] = [\![\exists b \exists u_1 \exists u_2 \exists v.(((b^2 - bc)u_1 - 1)v = 0 \wedge ((c^2 - 1)u_2)(1 - v) = 0)]\!].$$

We now do a final step of computation of the Gröbner basis G_3 of:

$$\langle ((b^2 - bc)u_1 - 1)v, ((c^2 - 1)u_2)(1 - v), b^3 - b, c^3 - c, u_1^3 - t_1, u_2^3 - t_2, v^3 - v \rangle \cap F_3[c].$$

We obtain $G_3 = \{c^3 - c\}$. This gives us the result formula $[\![\varphi]\!] = [\![c^3 - c = 0]\!]$, which means that c can take any value in F_3 to make the formula true.

6.2 Analyzing a Biological Controller Design

We studied a virus competition model named S2VD [11], which models the dynamics of virus competition as a polynomial system over finite fields. The authors aimed to design a controller to ensure that one virus prevail in the environment. They pointed out that there was no existing method for verifying its correctness. The current design is confirmed effective by computer simulation and lab experiments for a wide range of initializations. We attempted to establish the correctness of the design with formal verification techniques. However, we found bugs in the design.

All the Gröbner basis computations in this section are done using scripts in the SAGE system [1], which uses the underlying Singular implementation [2]. All the formulas below are solved within 5 seconds on a Linux machine with 2GHz CPU and 2GB RAM. They involve around 20 variables over F_4, with nonlinear polynomials containing multiplicative products of up to 50 terms.

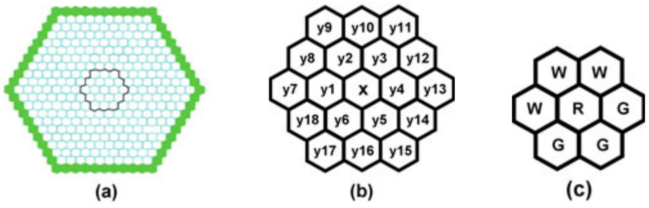


Fig. 1. (a) The ten rings of S2VD; (b) Cell x and its neighbor $\boldsymbol{y}$ cells; (c) The counterexample

The S2VD Model. The model consists of a hexagonal grid of cells. Each hexagon represents a cell, and each cell has six neighbors. There are four possible colors for each cell. A green cell is infected with (the good) Virus G, and a red cell is infected with (the bad) Virus R. When the two viruses meet in one cell, Virus G captures Virus R and the cell becomes yellow. A cell not infected by any virus is white. The dynamics of the system is determined by the interaction of the viruses.

There are ten rings of cells in the model, with a total of 331 cells (Figure 1(a)). In the initial configuration, the cells in Ring 4 to 10 are set to white, and the cells in Ring 1 to 3 can start with arbitrary colors. The aim is to have a controller that satisfies the following safety property: The cells in the outermost ring are either green or white at all times. The proposed controller detects if any cell has been infected by Virus R, and injects cells that are "one or two rings away" from it with Virus G. The injected Virus G is used to block the further expansion of Virus R.

Formally, the model is a polynomial system over the finite field $F_4 = \{0, 1, a, a+1\}$, with each element representing one color: $(0, green), (1, red), (a, white), (a+1, yellow)$. The dynamics is given by the function $f : F_4^{331} \to F_4^{331}$. For each cell x, its dynamics f_x is determined by the color of its six neighbors $y_1, ..., y_6$, specified by the nonlinear polynomial $f_x =_{df} \gamma_2^2 + \gamma_2\gamma_1^3 + a^2(\gamma_1^3 + \gamma_1^2 + \gamma_1)$, where $\gamma_1 = \sum_{i=1}^{6} y_i$ and $\gamma_2 = \sum_{i \neq j} y_i y_j$. The designed controller is specified by another function $g : F_4^{331} \to F_4^{331}$: For each cell x, with $y_1, ..., y_{18}$ representing the cells in the two rings surrounding it, we define $g_x =_{df} \prod_{i=1}^{18}(1 - y_i)^3$. More details can be found in [11].

Applying Quantifier Elimination. We first try checking whether the safety property itself forms an inductive invariant of the system (which is a strong sufficient check). To this end, we check whether the controlled dynamics of the system remain inside the invariant on the boundary (Ring 10) of the system. Let x be a cell in Ring 10 and $\boldsymbol{y} = (y_1, ..., y_{18})$ be the cells in its immediate two rings. We assume the cells outside Ring 10 $(y_8, ..., y_{12}, y_2, y_3)$ are white. See Figure 1(b) for the coding of the cells. We need to decide the formula:

$$\forall x(\underbrace{(\exists \boldsymbol{y}((\bigwedge_{i=8}^{12}(y_i = a) \wedge y_2 = a \wedge y_3 = a) \wedge \text{Safe}(\boldsymbol{y}) \wedge x = F_x(\boldsymbol{y})))}_{\varphi_1} \to \underbrace{x(x-a) = 0)}_{\text{"green/white"}} \quad (3)$$

where (writing $\gamma_1 = \sum_{i=1}^{6} y_i, \gamma_2 = \sum_{i \neq j \in \{1,...,6\}} y_i y_j$)

$$\text{Safe}(\boldsymbol{y}) =_{df} (y_1(y_1 - a) = 0 \wedge y_4(y_4 - a) = 0 \wedge y_7(y_7 - a) = 0 \wedge y_{13}(y_{13} - a) = 0)$$

$$F_x(\boldsymbol{y}) =_{df} (\gamma_2^2 + \gamma_2\gamma_1^3 + a^2(\gamma_1^3 + \gamma_1^2 + \gamma_1)) \cdot (\prod_{i=1}^{18}(1 - y_i))^3$$

After quantifier elimination, Formula (3) turns out to be false. In fact, we obtained $[\![\varphi_1]\!] = [\![x^4 - x = 0]\!]$. Therefore, the safety property itself is not an inductive invariant of the system. We realized that there is an easy counterexample of safety of the proposed controller design: Since the controller is only effective when red cells occur, it does not prevent the yellow cells to expand in all the cells. Although this is already a bug of the system, it may not conflict with the authors' original goal of controlling the red cells. However, a more serious bug is found by solving the following formula:

$$\underbrace{\forall x((\exists \boldsymbol{y}(\bigwedge_{i=1}^{18} y_i(y_i - a)(y_i - a^2) = 0) \wedge x = F_x(\boldsymbol{y}))}_{\varphi_2} \rightarrow \underbrace{\neg(x = 1)}_{\text{“not red”}}) \tag{4}$$

Formula (4) expresses the desirable property that when none of the neighbor cells of x is red, x never becomes red. However, we found again that $[\![\varphi_2]\!] = [\![x^4 - x = 0]\!]$, which means in this scenario the x cell can still turn red. Thus, the formal model is inconsistent with the informal specification of the system, which says that non-red cells can never interact to generate red cells. In fact, the authors mentioned that the dynamics F_x is not verified because of the combinatorial explosion. Finally, to give a counterexample of the design, we solve the formula

$$\varphi_3 =_{df} \exists \boldsymbol{y} \exists x.(x = 1 \wedge \bigwedge_{i=1}^{6} y_i(y_i - a)(y_i - a^2) = 0 \wedge x = F_x(\boldsymbol{y})) \tag{5}$$

The formula checks whether there exists a configuration of $y_1, ..., y_6$ which are all non-red, such that x becomes red. φ_3 evaluates to true. Further, we obtain $x = 1, \boldsymbol{y} = (a, a, a, 0, 0, 0)$ as a witness assignment for φ_3. This serves as the counterexample (see Figure 1(c)).

This example shows how our quantifier elimination procedure provides a practical way of verifying and debugging systems over finite fields that were previously not amenable to existing formal methods and cannot be approached by exhaustive enumeration.

7 Conclusion

In this paper, we gave a quantifier elimination algorithm for the first-order theory over finite fields based on the Nullstellensatz over finite fields and Gröbner basis computation. We exploited special properties of finite fields and showed the correspondence between elimination of quantifiers, projection of varieties, and computing elimination ideals. We also generalized the Tseitin transformation from Boolean formulas to formulas over finite fields using ideal operations. The complexity of our algorithm depends on the complexity of Gröbner basis computation. In an application of the algorithm, we successfully found bugs in a biological controller design, where the original authors expressed that no verification methods were able to handle the system. In future work, we expect to use the algorithm to formally analyze more systems with finite field arithmetic. The scalability of the method will benefit from further optimizations on Gröbner basis computation over finite fields. It is also interesting to combine Gröbner basis methods and other efficient Boolean methods (SAT and QBF solving). See [9] for a discussion on how the two methods are complementary to each other.

Acknowledgement

The authors are grateful for many important comments from Jeremy Avigad, Helmut Veith, Paolo Zuliani, and the anonymous reviewers.

References

1. The SAGE Computer Algebra system, `http://sagemath.org`
2. The Singular Computer Algebra system, `http://www.singular.uni-kl.de/`
3. Becker, T., Weispfenning, V.: Gröbner Bases. Springer, Heidelberg (1998)
4. Le Borgne, M., Benveniste, A., Le Guernic, P.: Polynomial dynamical systems over finite fields. In: Algebraic Computing in Control, vol. 165. Springer, Heidelberg (1991)
5. Marchand, H., Le Borgne, M.: On the Optimal Control of Polynomial Dynamical Systems over Z/pZ. In: 4th International Workshop on Discrete Event Systems, pp. 385–390 (1998)
6. Buchberger, B.: A Theoretical Basis for the Reduction of Polynomials to Canonical Forms. ACM SIGSAM Bulletin 10(3), 19–29 (1976)
7. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms. Springer, Heidelberg (1997)
8. Cox, D., Little, J., O'Shea, D.: Using Algebraic Geometry. Springer, Heidelberg (2005)
9. Gao, S.: Counting Zeroes over Finite Fields with Gröbner Bases. Master Thesis, Carnegie Mellon University (2009)
10. Germundsson, R.: Basic results on ideals and varieties in Finite Fields. Technical Report LiTH-ISY-I-1259, Linkoping University, S-581 83 (1991)
11. Jarrah, A., Vastani, H., Duca, K., Laubenbacher, R.: An Optimal Control Problem for in vitro Virus Vompetition. In: 43rd IEEE Conference on Decision and Control (2004)
12. Jarrah, A.S., Laubenbacher, R., Stigler, B., Stillman, M.: Reverse-engineering of polynomial dynamical systems. Advances in Applied Mathematics 39, 477–489 (2007)
13. Lakshman, Y.N.: On the Complexity of Computing a Gröbner Casis for the Radical of a Zero-dimensional Ideal. In: STOC 1990, New York, USA, pp. 555–563 (1990)
14. Lang, S.: Algebra, 3rd edn. Springer, Heidelberg (2005)
15. Marker, D.: Model theory. Springer, Heidelberg (2002)
16. Smith, E.W., Dill, D.L.: Automatic Formal Verification of Block Cipher Implementations. In: FMCAD, pp. 1–7 (2008)
17. Tran, Q.N.: Gröbner Bases Computation in Boolean Rings is PSPACE. International Journal of Applied Mathematics and Computer Sciences 5(2) (2009)

Appendix: Omitted Proofs

Proof of Lemma 2.1. This is a consequence of the Seidenberg's Lemma (Lemma 8.13 in [3]). It can also be directly proved as follows.

Proof. We need to show $\sqrt{J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle} = J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$. Since by definition, any ideal is contained in its radical, we only need to prove

$$\sqrt{J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle} \subseteq J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle.$$

Let R denote $F_q[x_1, ..., x_n]$. Consider an arbitrary polynomial f in the ideal $\sqrt{J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle}$. By definition, for some integer s, $f^s \in J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$. Let $[f]$ and $[J]$ be the images of, respectively, f and J, in $R/\langle x_1^q - x_1, ..., x_n^q - x_n\rangle$ under the canonical homomorphism from R to $R/\langle x_1^q - x_1, ..., x_n^q - x_n\rangle$. For brevity we write $S = \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$.

Now we have $[f]^s \in [J]$, and we further need $[f] \in [J]$. We prove, by induction on the structure of polynomials, that for any $[g] \in R/S$, $[g]^q = [g]$.

– If $[g] = cx_1^{a_1} \cdots x_n^{a_n} + S$ ($c \in F_q, a_i \in N$), then

$$[g]^q = (cx_1^{a_1} \cdots x_n^{a_n} + S)^q = (cx_1^{a_1} \cdots x_n^{a_n})^q + S = cx_1^{a_1} \cdots x_n^{a_n} + S = [g].$$

– If $[g] = [h_1] + [h_2]$, by inductive hypothesis, $[h_1]^q = [h_1], [h_2]^q = [h_2]$, and, since any element divisible by p is zero in F_q ($q = p^r$), then

$$[g]^q = ([h_1] + [h_2])^q = \sum_{i=0}^{q} \binom{q}{i} [h_1]^i [h_2]^{q-i} = [h_1]^q + [h_2]^q = [h_1] + [h_2] = [g]$$

Hence $[g]^q = [g]$ for any $[g] \in R/S$, without loss of generality we can assume $s < q$ in $[f]^s$. Then, since $[f]^s \in [J]$, $[f] = [f]^q = [f]^s \cdot [f]^{q-s} \in [J]$. □

Proof of Lemma 3.1

Proof. Let $\boldsymbol{a} \in F_q^{n+m}$ be an assignment vector for $(\boldsymbol{x}, \boldsymbol{y})$.

If $\boldsymbol{a} \in [\![\bigwedge_{i=1}^r f_i = 0]\!]$, then $f_1(\boldsymbol{a}) = \cdots = f_r(\boldsymbol{a}) = 0$ and $\boldsymbol{a} \in V(\langle f_1, ..., f_k\rangle)$.

If $\boldsymbol{a} \in V(\langle f_1, ..., f_r\rangle)$, then $\bigwedge_{i=1}^r f_i(\boldsymbol{a}) = 0$ is true and $\boldsymbol{a} \in [\![\bigwedge_{i=1}^r f_i = 0]\!]$. □

Proof of Lemma 3.2

Proof. We show set inclusion in both directions.

– For any $\boldsymbol{b} \in [\![\exists \boldsymbol{x} \varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$, by definition, there exists $\boldsymbol{a} \in F_q^n$ such that $(\boldsymbol{a}, \boldsymbol{b})$ satisfies $\varphi(\boldsymbol{x}; \boldsymbol{y})$. Therefore, $(\boldsymbol{a}, \boldsymbol{b}) \in [\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$, and $\boldsymbol{b} \in \pi_n([\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!])$.
– For any $\boldsymbol{b} \in \pi_n([\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!])$, there exists $\boldsymbol{a} \in F_q^n$ such that $(\boldsymbol{a}, \boldsymbol{b}) \in [\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$. By definition, $\boldsymbol{b} \in [\![\exists x \varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$. □

Proof of Lemma 3.3

Proof. We have $[\![\bigwedge_{i \in A_x} (x_i^q - x_i = 0) \wedge \bigwedge_{i \in A_y} (y_i^q - y_i = 0)]\!] = [\![\top]\!]$, which follows from Proposition 2.1. □

Proof of Lemma 4.1

Proof. Let $\varphi[\psi_1/\psi_2]$ denote substitution of ψ_1 in φ by ψ_2. Suppose the negative atomic formulas in φ are $f_1 \neq 0, ..., f_k \neq 0$.

We introduce a new variable z_1, and substitute $f_1 \neq 0$ by $p \cdot z_1 = 1$. Since the field F_q does not have zero divisors, all the solutions for $[\![f_1 \neq 0]\!] = [\![\exists z_1(p \cdot z_1 = 1)]\!]$ (the Rabinowitsch trick).

Iterating the procedure, we can use k new variables $z_1, ..., z_k$ so that:

$$[\![\varphi]\!] = [\![\varphi[f_1 \neq 0/(\exists z_1.(p \cdot z_1 - 1 = 0))] \cdots [f_k \neq 0/(\exists z_k.(p \cdot z_k - 1 = 0))]]\!]$$

Since the result formula contains no more negations and the z_is are new variables, it can be put into prenex form $\exists \boldsymbol{z}.(\varphi[f_1 \neq 0/(p \cdot z_1 - 1 = 0)] \cdots [f_k \neq 0/(p \cdot z_k - 1 = 0)])$. □

Proof of Lemma 4.2

Proof. $[\![\psi_1 \vee \psi_2]\!] = V(J_1) \cup V(J_2)$ follows from the definition of realization. We only need to show the second equality. Let $\boldsymbol{a} = (a_1, ..., a_n) \in F_q^n$ be a point.

- Suppose $\boldsymbol{a} \in V(J_1) \cup V(J_2)$. If $\boldsymbol{a} \in V(J_1)$, then $(1, a_1, ..., a_n) \in V(x_0 J_1 + (1 - x_0)J_2)$. If $\boldsymbol{a} \in V(J_2)$, then $\langle 0, a_1, ..., a_n\rangle \in V(x_0 J_1 + (1 - x_0)J_2)$. In both cases, $\boldsymbol{a} \in \pi_0(V(x_0 J_1 + (1 - x_0)J_2))$.

- Suppose $\boldsymbol{a} \in \pi_0(V(x_0 J_1 + (1 - x_0)J_2))$. There exists $a_0 \in F_q$ such that $(a_0, a_1, ..., a_n) \in V(x_0 J_1 + (1 - x_0)J_2)$. If $a_0 \notin \{0, 1\}$, then all the polynomials in J_1 and J_2 need to vanish on $\boldsymbol{a}$; if $a_0 = 1$ then J_1 vanishes on $\boldsymbol{a}$; if $a_0 = 0$ then J_2 vanishes on $\boldsymbol{a}$. In all cases, $\boldsymbol{a} \in V(J_1) \cup V(J_2)$. □

Proof of Theorem 5.1

Proof. We only need to show the intermediate formulas obtained in Step 1-3 are always equivalent to the original formula φ. In Step 1, the formula is flattened with ideal operations, which preserve the realization of the formula as proved in Theorem 4.1. In Step 2, we have (by Theorem 3.2) $[\![\exists \boldsymbol{x}_m \exists \boldsymbol{t} \exists \boldsymbol{s}(\bigwedge_{i=1}^r (f_i = 0))]\!] = [\![\bigwedge_{i=1}^u (g_i = 0)]\!]$.

Hence the formula obtained in Step 2 is equivalent to φ. In Step 3, the substitution preserves realization of the formula because

$$[\![\bigwedge_{i=1}^{u} \forall \boldsymbol{x}_{m-1}(g_i = 0)]\!] = [\![\bigwedge_{i=1}^{u} (\neg \exists \boldsymbol{x}_{m-1}(\neg g_i = 0))]\!] = [\![(\bigwedge_{i=1}^{u} (\bigvee_{j=1}^{v_i} h_{ij} \neq 0))]\!],$$

where the second equality is guaranteed by Theorem 3.2 again.
The loop terminates either at the end of Step 2 or Step 3. Hence the output quantifier-free formula ψ is always in conjunctive normal form, which contains only variables $\boldsymbol{y}$, and is equivalent to the original formula φ. □

$\mathbb{F}$-Rank-Width of (Edge-Colored) Graphs[⋆]

Mamadou Moustapha Kanté[1] and Michael Rao[2]

[1] Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France
mamadou.kante@isima.fr
[2] CNRS - Laboratoire J.V. Poncelet, Moscow, Russia
and Université de Bordeaux, LaBRI, CNRS, France
rao@labri.fr

Abstract. *Rank-width* is a complexity measure equivalent to the *clique-width* of undirected graphs and has good algorithmic and structural properties. It is in particular related to the *vertex-minor* relation. We discuss an extension of the notion of rank-width to all types of graphs - directed or not, with edge colors or not -, named $\mathbb{F}$-*rank-width*. We extend most of the results known for the rank-width of undirected graphs to the $\mathbb{F}$-rank-width of graphs: cubic-time recognition algorithm, characterisation by excluded configurations under *vertex-minor* and *pivot-minor*, and algebraic characterisation by graph operations. We also show that the rank-width of undirected graphs is a special case of $\mathbb{F}$-rank-width.

Keywords: rank-width, clique-width, local complementation, vertex-minor, pivot-minor, excluded configuration, 2-structure, sigma-symmetry.

1 Introduction

In their investigations of a recognition algorithm for undirected graphs of *clique-width* at most k, for fixed k, Oum and Seymour [21] introduced the notion of *rank-width* of undirected graphs. Rank-width is defined in a combinatorial way and is equivalent to the clique-width of undirected graphs in the sense that a class of graphs has bounded clique-width if and only if it has bounded rank-width [21]. But, being defined in a combinatorial way provides to rank-width better algorithmic properties than clique-width. In particular, for fixed k, there exists a cubic-time algorithm that decides whether the rank-width of an undirected graph is at most k and if so, constructs a rank-decomposition of width at most k [15]. Moreover, a characterisation in terms of graph operations is given in [5] which allows to solve MSOL properties without using clique-width operations.

Another advantage of rank-width over clique-width is that it is invariant with respect to the *vertex-minor* relation (no such notion, except for induced subgraph relation, is known for clique-width), *i.e.*, if H is a vertex-minor of G,

⋆ A part of this research is supported by the projects "Graph decompositions and algorithms (GRAAL)" and "Decomposition Of Relational Structures and Combinatorial Optimisation (DORSO)" of French "Agence Nationale Pour la Recherche" and was done when the first author was in Université Bordeaux 1, LaBRI.

then the rank-width of H is at most the rank-width of G [20]. Moreover, every class of undirected graphs of bounded rank-width is characterised by a finite list of undirected graphs to exclude as vertex-minors [20]. This later result generalises the one of Robertson and Seymour on undirected graphs of bounded tree-width [22].

Despite all these positive results of rank-width, the fact that clique-width is defined for graphs - directed or not, with edge colors or not - is an undeniable advantage over rank-width. It is thus natural to ask for a notion of rank-width for all types of graphs - directed or not, with edge colors or not - that has the same advantages as the rank-width of undirected graphs. We investigate in this paper such a notion of rank-width. It is worth noticing that there is no unique natural way to define it. For instance, Courcelle and Oum [8], and Courcelle [4] suggested to define the rank-width of a graph G as the rank-width of $B(G)$, where $B(G)$ is an undirected bipartite graph associated - in a unique way, up to isomorphism - to G. However, this definition, even if it approximates well clique-width, suffers from, among others, this important drawback: a vertex-minor of $B(G)$ does not always correspond to a coding of a graph. In this paper, we define a better notion of rank-width for graphs that, not only extends the rank-width of undirected graphs, but also shares most of its advantages. For these purposes, we will define the notion of *sigma-symmetric* matrices, which generalizes the notion of symmetric and skew-symmetric matrices. We then use this notion to represent graphs by matrices over finite fields and derive, from this representation, a notion of rank-width, called $\mathbb{F}$*-rank-width* (Section 3), that generalises the one of undirected graphs. A cubic-time recognition algorithm and a characterisation of graphs of $\mathbb{F}$-rank-width at most k by excluded configurations under vertex-minor are presented. We finish by characterisations in terms of graph operations.

This paper is related to a companion paper where the authors introduce a decomposition of graphs on a fixed field [18]. This decomposition plays a role similar to the *split decomposition* [9] for the rank-width of undirected graphs. Particularly we show that the $\mathbb{F}$-rank-width of a graph is exactly the maximum over the $\mathbb{F}$-rank-width over all prime graphs in the decomposition, and we give different characterisations of graphs of $\mathbb{F}$-rank-width one.

2 Preliminaries

If A and B are two sets, $A\backslash B$ denotes the set $\{x \in A \mid x \notin B\}$. The power-set of a set V is denoted by 2^V.

The set of natural integers is denoted by $\mathbb{N}$. We denote by $+$ and $\cdot$ the binary operations of any field and by 0 and 1 the neutral elements of $+$ and $\cdot$ respectively. We refer to [19] for our field terminology.

For sets R and C, an (R, C)*-matrix* is a matrix where the rows are indexed by elements in R and columns indexed by elements in C. For $X \subseteq R$ and $Y \subseteq C$, let $M[X, Y]$ be the sub-matrix of M where the rows and the columns are indexed by X and Y respectively. Let rk be the matrix rank-function (the field will be clear from the context). We denote by M^T the transpose of a matrix M. The order of an (R,C)-matrix is defined as $|R| \times |C|$. We often write $k \times \ell$-matrix to denote a matrix of order $k \times \ell$.

We use the standard graph terminology, see for instance [10]. A graph G is a couple (V_G, E_G) where V_G is the set of vertices and $E_G \subseteq V_G \times V_G$ is the set of edges. A graph G is said to be *oriented* if $(x,y) \in E_G$ implies $(y,x) \notin E_G$, and it is said *undirected* if $(x,y) \in E_G$ implies $(y,x) \in E_G$, hence we can write xy (equivalently yx). We let $G[X]$, called the sub-graph of G induced by $X \subseteq V_G$, the graph $(X, E_G \cap (X \times X))$ and write G-X for $G[V_G \backslash X]$. Two graphs G and H are *isomorphic* if there exists a bijection $h : V_G \to V_H$ such that $(x,y) \in E_G$ if and only if $(h(x), h(y)) \in E_H$. We suppose that all graphs are finite and loop-free (*i.e.* $(x,x) \notin E_G$ for every $x \in V_G$).

A *tree* is an acyclic connected undirected graph. The vertices of degree 1 are called *leaves* and the set of leaves in a tree T is denoted by L_T. A *sub-cubic tree* is an undirected tree such that the degree of each node is at most 3. For a tree T and an edge e of T, we let T-e denote the graph $(V_T, E_T \backslash \{e\}$.

Let C be a (possibly infinite) set that we call the *colors.* A *C-graphs* G is a tuple (V_G, E_G, ℓ_G) where (V_G, E_G) is a graph and $\ell_G : E_G \to C$ is a function. Its associated *underlying graph* $u(G)$ is the graph (V_G, E_G). Two C-graphs G and H are isomorphic if there is an isomorphism h between (V_G, E_G) and (V_H, E_H) such that for every $(x,y) \in E_G$, $\ell_G((x,y)) = \ell_H((h(x), h(y)))$. We let $\mathscr{G}(C)$ be the set of C-graphs for a fixed color set C. Even though we authorise infinite color sets in the definition, most of the results in this article concern only finite color sets. It is worth noticing that an edge-uncolored graph can be seen as an edge-colored graph where all the edges have the same color.

In our definition, an edge in a C-graph has only one color. However, this is not restrictive because if in an edge-colored graph an edge can have several colors from a set C, we just extend C to 2^C.

Remark 1 (2-structures and edge-colored graphs). A *2-structure* [11] is a pair (D,R) where D is a finite set and R is an equivalence relation on the set $D_2 = \{(x,y) \mid x,y \in D \text{ and } x \neq y\}$. Every 2-structure (D,R) can be seen as a C-colored graph $G = (D, D_2, \ell)$ where $C := \{[e] \mid [e] \text{ is an equivalence class of } R\}$ and for every edge e, $\ell(e) := [e]$. Equivalently, every C-graph G can be seen as a 2-structure (V_G, R) where eRe' if and only if $\ell_G(e) = \ell_G(e')$ and all the non-edges in G are equivalent with respect to R.

The *clique-width*, denoted by cwd, is a graph parameter defined by Courcelle and Olariu [7]. It is studied in many other papers (see for instance the survey [16]). But, most of the investigations concern edge-uncolored graphs. However, its edge-colored version has been investigated these last years (see [2,12]). Note that it is also defined in a more general case where edges can have several colors [3].

If $\mathcal{F}$ is a set of binary and unary function symbols and $\mathcal{C}$ a set of constants, we let $T(\mathcal{F}, \mathcal{C})$ be the set of finite well-formed terms built with $\mathcal{F} \cup \mathcal{C}$.

3 $\mathbb{F}$-Rank-Width

We want a notion of rank-width for edge-colored graphs which generalises the one on undirected graphs (recall that an undirected graph can be seen as an edge-colored graph with all edges having same color). For this purpose, we will identify each color by an non-zero element of a field. This representation will allow us to define the rank-width of edge-colored graphs using the matrix rank.

Let $\mathbb{F}$ be a field, and let $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$. One can note that there is a natural bijection between the class of $\mathbb{F}^*$-graphs and the class of $\mathbb{F}$-graphs with complete underlying graph (replace every non-edge by an edge of color 0). From now on, we do not distinguish these two representations, and we let $\ell_G((x,y)) = 0$ for all $(x,y) \notin E_G$.

We can represent every $\mathbb{F}^*$-graph G by a (V_G, V_G)-matrix M_G such that $M_G[x,y] := \ell_G((x,y))$ for every $x,y \in V_G$ with $x \neq y$, and $M_G[x,x] := 0$ for every $x \in V_G$.

Let $\sigma : \mathbb{F} \to \mathbb{F}$ be a bijection. We recall that σ is an *involution* if $\sigma(\sigma(a)) = a$ for all $a \in \mathbb{F}$. We say that σ is a *sesqui-morphism* if σ is an involution, and the mapping $[x \mapsto \sigma(x)/\sigma(1)]$ is an automorphism. It is worth noticing in this case that $\sigma(0) = 0$ and for every $a, b \in \mathbb{F}$, $\sigma(a+b) = \sigma(a) + \sigma(b)$ (*i.e.* σ is an automorphism for the addition).

A $\mathbb{F}^*$-graph is *σ-symmetric* if for every arc (x,y), $\ell_G((x,y)) = a$ if and only if $\ell_G((y,x)) = \sigma(a)$. Clearly, if G is a σ-symmetric $\mathbb{F}^*$-graph, then $M_G[x,y] = \sigma(M_G[y,x])$. We denote by $\mathscr{S}(\mathbb{F})$ (respectively $\mathscr{S}(\mathbb{F},\sigma)$) the set of $\mathbb{F}^*$-graphs (respectively σ-symmetric $\mathbb{F}^*$-graphs). Note that $\mathscr{S}(\mathbb{F}) = \mathscr{G}(\mathbb{F}^*)$.

To represent a C-graph, one can take an injection from C to $\mathbb{F}^*$ for a large enough field $\mathbb{F}$. Notice that the representation is not unique: on one hand, several incomparable fields are possible for $\mathbb{F}$, and on the other hand, the representation depends on the injection from C to $\mathbb{F}^*$. For example, oriented graphs can be represented by a $\mathbb{F}_3^*$-graph or by a $\mathbb{F}_4^*$-graph (see Section 3.4). Two different representations can give two different rank-width parameters, but the two parameters are equivalent when C is finite (direct consequence of Proposition 3).

We now explain how to see a $\mathbb{F}^*$-graph as a $\widetilde{\sigma}$-symmetric $(\mathbb{F}^2)^*$-graph, using the algebraic extension $\mathbb{F}^2$ of the field $\mathbb{F}$ (proofs are omitted because of space constraints).

Lemma 1. *There exists an element p in $\mathbb{F}^*$ such that the polynomial $X^2 - p(X+1)$ has no root in $\mathbb{F}$.*

We construct an algebraic extension of the finite field $\mathbb{F}$. Let $p \in \mathbb{F}^*$ such that $X^2 - p(X+1)$ has no root in $\mathbb{F}$ and let $\mathbb{F}^2$ be isomorphic to the field $\mathbb{F}[X] \bmod (X^2 - p(X+1))$ (*i.e.* $\mathbb{F}^2$ is the finite field with the same characteristic and order $|\mathbb{F}|^2$). Let $\alpha := X \mod (X^2 - p(X+1))$. Then every element of $\mathbb{F}^2$ is a polynomial on α of the form $a_0 + a_1\alpha$ where $a_0, a_1 \in \mathbb{F}$. Moreover, α is a root of $X^2 - p(X+1)$ in $\mathbb{F}^2$. We let $\gamma := 1 - p^{-1}\alpha$ and $\tau := p^{-1}\alpha$ be in $\mathbb{F}^2$. Notice that $\alpha = p\tau$ and $1 = \gamma + \tau$.

Lemma 2. *We have the following equalities:*

$$\begin{aligned}\gamma^2 &= (1+p^{-1})\gamma + p^{-1}\tau,\\ \tau^2 &= p^{-1}\gamma + (1+p^{-1})\tau,\\ \gamma\cdot\tau &= p^{-1}\gamma + p^{-1}\tau.\end{aligned}$$

To every pair of elements in $\mathbb{F}$, we associate an element in $\mathbb{F}^2$ by letting $\widetilde{f} : \mathbb{F}\times\mathbb{F}\to\mathbb{F}^2$ where, for every $(a,b)\in\mathbb{F}\times\mathbb{F}$, $\widetilde{f}(a,b) := a\gamma + b\tau$.

Lemma 3. *$\widetilde{f}$ is a bijection.*

For the sesqui-morphism in $\mathbb{F}^2$, we let $\widetilde{\sigma} : \mathbb{F}^2\to\mathbb{F}^2$ where $\widetilde{\sigma}(a\gamma+b\tau) := b\gamma + a\tau$. One easily verifies that $\widetilde{\sigma}(\widetilde{\sigma}(\beta)) = \beta$ for all $\beta\in\mathbb{F}^2$.

Lemma 4. *$\widetilde{\sigma}$ is an automorphism.*

For every $\mathbb{F}^*$-graph G, we let $\widetilde{G}$ be the $(\mathbb{F}^2)^*$-graph $(V_G, E_G, \ell_{\widetilde{G}})$ where, for every two distinct vertices x and y,

$$\ell_{\widetilde{G}}((x,y)) := \widetilde{f}(\ell_G((x,y)), \ell_G((y,x))).$$

By the definitions of $\widetilde{G}$ and $\widetilde{\sigma}$, and Lemmas 2-4, we get the following.

Proposition 1. *The mapping $[G\mapsto\widetilde{G}]$ from $\mathscr{S}(\mathbb{F})$ to $\mathscr{S}(\mathbb{F}^2,\widetilde{\sigma})$ is a bijection and for every $\mathbb{F}^*$-graph G, $\widetilde{G}$ is $\widetilde{\sigma}$-symmetric. Moreover, for two $\mathbb{F}^*$-graphs G and H, $\widetilde{G}$ and $\widetilde{H}$ are isomorphic if and only if G and H are isomorphic.*

If $\mathbb{F}$ is infinite, a mapping from $\mathscr{S}(\mathbb{F})$ to $\mathscr{S}(\mathbb{G},\sigma)$ is not always possible with the previous construction. For example, a mapping is possible from $\mathscr{S}(\mathbb{R})$ to $\mathscr{S}(\mathbb{C},\sigma)$ with $f(a,b) = (1+i)a + (1-i)b$ and $\sigma(a+ib) = a - ib$ (where $a,b\in\mathbb{R}$), but the construction fails for $\mathbb{F}=\mathbb{C}$ since the complexes are algebraically closed.

From Proposition 1, we can now concentrate to σ-symmetric $\mathbb{F}^*$-graphs.

3.1 Rank-Width of σ-Symmetric $\mathbb{F}^*$-Graphs

We say that a function $f : 2^V\to\mathbb{N}$ is *symmetric* if for any $X\subseteq V$, $f(X) = f(V\backslash X)$; it is *submodular* if for any $X, Y\subseteq V$, $f(X\cup Y) + f(X\cap Y)\leq f(X) + f(Y)$.

A *layout* of a finite set V is a pair $(T,\mathcal{L})$ of a sub-cubic tree T and a bijective function $\mathcal{L} : V\to \mathrm{L}_T$. For each edge e of T, the connected components of T-e induce a bipartition $(X_e, V\backslash X_e)$ of L_T, and thus a bipartition $(X^e, V\backslash X^e) = (\mathcal{L}^{-1}(X_e), \mathcal{L}^{-1}(V\backslash X_e))$ of V (we will omit the sub or superscript e when the context is clear).

Let $f : 2^V\to\mathbb{N}$ be a symmetric function and $(T,\mathcal{L})$ a layout of V. The *f-width of each edge e of T* is defined as $f(X^e)$ and the *f-width of* $(T,\mathcal{L})$ is the maximum f-width over all edges of T. The *f-width of V* is the minimum f-width over all layouts of V.

Along this section, we let $\mathbb{F}$ be a fixed field (of characteristic p and of order q if $\mathbb{F}$ is finite), and we let $\sigma : \mathbb{F}\to\mathbb{F}$ be a fixed sesqui-morphism. If G is a $\mathbb{F}^*$-graph, we let $\mathrm{cutrk}_G^{\mathbb{F}}(X) = \mathrm{rk}(M_G[X, V_G\backslash X])$ for all $X\subseteq V_G$.

Lemma 5. *For every σ-symmetric $\mathbb{F}^*$-graph G, the function $\mathrm{cutrk}_G^{\mathbb{F}}$ is symmetric and submodular.*

In order to prove Lemma 5, we first recall the *submodular inequality* of the matrix rank-function.

Proposition 2. *[20, Proposition 4.1] Let M be an (R, C)-matrix over a field $\mathbb{F}$. Then for all $X_1, Y_1 \subseteq R$ and $X_2, Y_2 \subseteq C$,*

$$\mathrm{rk}(M[X_1, X_2]) + rk(M[Y_1, Y_2]) \geq \mathrm{rk}(M[X_1 \cup Y_1, X_2 \cap Y_2]) + \mathrm{rk}(M[X_1 \cap Y_1, X_2 \cup Y_2]).$$

Proof (of Lemma 5). Let X and Y be subsets of V_G. We let $A_1 = M_G[X, V_G \backslash X]$ and $A_2 = M_G[Y, V_G \backslash Y]$. We first prove the first statement.

We let M' be the $(V_G \backslash X, X)$-matrix where $M'[y, x] = \sigma(A_1[x, y])/\sigma(1)$. Since σ is a sesqui-morphism, the mapping $[x \mapsto \sigma(x)/\sigma(1)]$ is an automorphism and then $\mathrm{rk}(M') = \mathrm{rk}((A_1)^T) = rk(A_1)$. But, $M_G[V_G \backslash X, X] = \sigma(1) \cdot M'$. Then, $\mathrm{rk}(M_G[V_G \backslash X, X]) = \mathrm{rk}(M') = \mathrm{rk}(M_G[X, V_G \backslash X])$.

For the second statement, we have by definition and Proposition 2,

$$\begin{aligned} \mathrm{cutrk}_G^{\mathbb{F}}(X) + \mathrm{cutrk}_G^{\mathbb{F}}(Y) &= \mathrm{rk}(A_1) + \mathrm{rk}(A_2) \\ &\geq \mathrm{rk}(M_G[X \cup Y, V_G \backslash X \cap V_G \backslash Y]) \\ &+ \mathrm{rk}(M_G[X \cap Y, V_G \backslash X \cup V_G \backslash Y]). \end{aligned}$$

Since $V_G \backslash X \cap V_G \backslash Y = V_G \backslash (X \cup Y)$ and $V_G \backslash X \cup V_G \backslash Y = V_G \backslash (X \cap Y)$, the second statement holds. □

The $\mathbb{F}$-*rank-width* of a σ-symmetric $\mathbb{F}^*$-graph G, denoted by $\mathrm{rwd}^{\mathbb{F}}(G)$, is defined as the $\mathrm{cutrk}_G^{\mathbb{F}}$-width of V_G.

This definition generalises the one for undirected graphs. If we let σ_1 be the identity automorphism on $\mathbb{F}_2$, every undirected graph is a σ_1-symmetric $\mathbb{F}_2^*$-graph. One easily verifies that $\mathbb{F}_2$-rank-width and rank-width ([21]) coincide.

One can easily verify that the $\mathbb{F}$-rank-width of a σ-symmetric $\mathbb{F}^*$-graph is the maximum of the $\mathbb{F}$-rank-width of its maximum connected components. The following proposition, which says that $\mathbb{F}$-rank-width and clique-width are equivalent when $\mathbb{F}$ is finite, has an easy proof. We omit it because its proof is an easy adaptation of the one comparing rank-width and clique-width of undirected graphs [21, Proposition 6.3].

Proposition 3. *Let G be a σ-symmetric $\mathbb{F}^*$-graph. Then, $\mathrm{rwd}^{\mathbb{F}}(G) \leq \mathrm{cwd}(G) \leq 2 \cdot q^{\mathrm{rwd}^{\mathbb{F}}(G)} - 1$.*

It is also easy to show that the clique-width and the $\mathbb{F}$-rank-width are equivalent if $\mathbb{F}$ is infinite but C is finite.

3.2 Vertex-Minor and Pivot-Minor

We say that λ in $\mathbb{F}^*$ is σ-*compatible*, for some sesqui-morphism $\sigma : \mathbb{F} \to \mathbb{F}$, if $\sigma(\lambda) = \lambda \cdot \sigma(1)^2$.

Definition 1 (λ-local complementation). *Let λ in $\mathbb{F}^*$. Let G be a $\mathbb{F}^*$-graph and x a vertex of G. The* λ-local complementation *at x of G is the $\mathbb{F}^*$-graph $G * (x, \lambda)$ represented by the (V_G, V_G)-matrix $M_{G*(x,\lambda)}$ where:*

$$M_{G*(x,\lambda)}[z,t] := \begin{cases} M_G[z,t] + \lambda \cdot M_G[z,x] \cdot M_G[x,t] & \textit{if } x \notin \{z,t\}, \\ M_G[z,t] & \textit{otherwise.} \end{cases}$$

One can easily verify that for every $\mathbb{F}^*$-graph G and every vertex x of G, the adjacency matrix of $G * (x, \lambda)$ is obtained by modifying the sub-matrix induced by the neighbors of x. Then for every vertex y of G, $M_G[x,y] = M_{G*(x,\lambda)}[x,y]$.

Since we are interested in σ-symmetric graphs, we have to restrict ourselves to a subset of λ-local complementations which preserve the σ-symmetry.

Lemma 6. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and let $\lambda \in \mathbb{F}^*$ be σ-compatible. Then every λ-local complementation of G is also σ-symmetric.*

Proof. Let $H := G * (x, \lambda)$ for some σ-compatible λ. It is sufficient to prove that $M_H[t,z] = \sigma(M_H[z,t])$ for any $z, t \in V_G,\ z \neq t$.

$$\begin{aligned} M_H[t,z] &= M_G[t,z] + \lambda \cdot M_G[t,x] \cdot M_G[x,z] \\ &= \sigma(M_G[z,t]) + \lambda \cdot \sigma(M_G[x,t]) \cdot \sigma(M_G[z,x]) \\ &= \sigma(M_G[z,t]) + \lambda \cdot \sigma(1) \cdot \sigma(M_G[z,x] \cdot M_G[x,t]) \\ &= \sigma(M_G[z,t]) + \sigma(\lambda) \cdot \sigma^{-1}(1) \cdot \sigma(M_G[z,x] \cdot M_G[x,t]) \\ &= \sigma(M_G[z,t]) + \sigma(\lambda \cdot M_G[z,x] \cdot M_G[x,t]) \\ &= \sigma(M_G[z,t] + \lambda \cdot M_G[z,x] \cdot M_G[x,t]) \\ &= \sigma(M_H[z,t]). \end{aligned}$$

□

Lemma 6 shows that λ-local complementation is well-defined on σ-symmetric $\mathbb{F}^*$-graphs for σ-compatible λ. Moreover, one can easily verify that, when $\mathbb{F}$ is the field $\mathbb{F}_2$, this notion of 1-local complementation coincides with the one defined by Bouchet (see [1,20]).

We call H a *σ-vertex-minor* of G if H is obtained from G by applying a sequence of λ-local complementations - with σ-compatibles λ - and deletions of vertices. Note that if no σ-compatible $\lambda \in \mathbb{F}^*$ exists, H is a σ-vertex-minor of G if and only if H is an induced subgraph of G.

The following lemma proves that λ-local-complementations do not increase $\mathbb{F}$-rank-width.

Lemma 7. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and x a vertex of G. For every subset X of V_G, $\operatorname{cutrk}^{\mathbb{F}}_{G*(x,\lambda)}(X) = \operatorname{cutrk}^{\mathbb{F}}_G(X)$.*

Proof. We can assume that $x \in X$ since $\operatorname{cutrk}^{\mathbb{F}}_G$ is a symmetric function (Lemma 5). For each $y \in X$, the λ-local-complementation at x results in adding a multiple of the row indexed by x to the row indexed by y. Precisely, we obtain $M_{G*(x,\lambda)}[y, V_G\backslash X]$ by adding $\lambda \cdot M_G[y,x] \cdot M_G[x, V_G\backslash X]$ to $M_G[y, V_G\backslash X]$. This operation is repeated for all $y \in X$. In each case, the rank of the matrix does not change. Hence, $\operatorname{cutrk}^{\mathbb{F}}_{G*(x,\lambda)}(X) = \operatorname{cutrk}^{\mathbb{F}}_G(X)$. □

Unfortunately, such a σ-compatible λ does not always exist. For instance, if the field is $\mathbb{F}_3$ and σ is such that $\sigma(x) = -x$ (see Section 3.4), no σ-compatible λ exists. We present now another $\mathbb{F}^*$-graph transformation which is defined for every couple $(\mathbb{F}, \sigma)$.

Definition 2 (Pivot-complementation). *Let G be a σ-symmetric $\mathbb{F}^*$-graph, and x and y two vertices of G such that $\ell_G((x,y)) \neq 0$. The* pivot-complementation at *xy of G is the $\mathbb{F}^*$-graph $G \wedge xy$ represented by the (V_G, V_G)-matrix $M_{G\wedge xy}$ where $M_{G\wedge xy}[z,z] := 0$ for every $z \in V_G$, and for every $z, t \in V_G \setminus \{x,y\}$ with $z \neq t$:*

$$M_{G\wedge xy}[z,t] := M_G[z,t] - \frac{M_G[z,x] \cdot M_G[y,t]}{M_G[y,x]} - \frac{M_G[z,y] \cdot M_G[x,t]}{M_G[x,y]}$$

$$M_{G\wedge xy}[x,t] := \frac{M_G[y,t]}{M_G[y,x]} \qquad M_{G\wedge xy}[y,t] := \frac{\sigma(1) \cdot M_G[x,t]}{M_G[x,y]}$$

$$M_{G\wedge xy}[z,x] := \frac{\sigma(1) \cdot M_G[z,y]}{M_G[x,y]} \qquad M_{G\wedge xy}[z,y] := \frac{M_G[z,x]}{M_G[y,x]}$$

$$M_{G\wedge xy}[x,y] := -\frac{1}{M_G[y,x]} \qquad M_{G\wedge xy}[y,x] := -\frac{\sigma(1)^2}{M_G[x,y]}$$

We call H a pivot-minor *of G if H is obtained from G by applying a sequence of pivot-complementations and deletions of vertices.*

Note that $G \wedge xy = G \wedge yx$ if $\sigma(1) = 1$. In the case of undirected graphs ($\mathbb{F} = \mathbb{F}_2$), this definition coincides with the pivot-complementation of undirected graphs [20]. The following lemma shows that this transformation is well defined.

Lemma 8. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and let xy be an edge of G. Then $G \wedge xy$ is also σ-symmetric.*

Proof. Let $z, t \in V$, with $z \neq t$. If $\{z,t\} \cap \{x,y\} = \emptyset$, then

$$\begin{aligned} M_{G\wedge xy}[t,z] &= M_G[t,z] - \frac{M_G[t,x] \cdot M_G[y,z]}{M_G[y,x]} - \frac{M_G[t,y] \cdot M_G[x,z]}{M_G[x,y]} \\ &= \sigma(M_G[z,t]) - \frac{\sigma(M_G[x,t]) \cdot \sigma(M_G[z,y])}{\sigma(M_G[x,y])} - \frac{\sigma(M_G[y,t]) \cdot \sigma(M_G[z,x])}{\sigma(M_G[y,x])} \\ &= \sigma(M_G[z,t]) - \sigma\left(\frac{M_G[x,t] \cdot M_G[z,y]}{M_G[x,y]}\right) - \sigma\left(\frac{M_G[y,t] \cdot M_G[z,x]}{M_G[y,x]}\right) \\ &= \sigma\left(M_G[z,t]) - \frac{M_G[x,t] \cdot M_G[z,y]}{M_G[x,y]} - \frac{M_G[y,t] \cdot M_G[z,x]}{M_G[y,x]}\right) \\ &= \sigma\left(M_{G\wedge xy}[z,t]\right). \end{aligned}$$

If $t \neq y$, then:

$$\begin{aligned} M_{G\wedge xy}[t,x] &= \frac{\sigma(1) \cdot M_G[t,y]}{M_G[x,y]} = \frac{\sigma(1) \cdot \sigma(M_G[y,t])}{\sigma(M_G[y,x])} \\ &= \sigma\left(\frac{M_G[y,t]}{M_G[y,x]}\right) = \sigma\left(M_{G\wedge xy}[x,t]\right). \end{aligned}$$

Finally:

$$\begin{aligned} M_{G\wedge xy}[y,x] &= -\frac{\sigma(1)^2}{M_G[x,y]} = -\frac{\sigma(1)^2}{\sigma(M_G[y,x])} \\ &= \sigma\left(-\frac{1^2}{M_G[y,x]}\right) = \sigma\left(M_{G\wedge xy}[x,y]\right). \end{aligned}$$ □

Similarly to Lemma 7, the following lemma proves that pivot-complementations do not increase $\mathbb{F}$-rank-width.

Lemma 9. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and xy an edge of G. For every subset X of V_G, $\operatorname{cutrk}^{\mathbb{F}}_{G\wedge xy}(X) = \operatorname{cutrk}^{\mathbb{F}}_G(X)$.*

Proof. Let $Y := V_G \setminus X$. We can assume w.l.o.g. that $x \in X$. If $y \in X$, then (with $X' := X \setminus \{x,y\}$)

$$\begin{aligned} \operatorname{rk}\left(M_{G\wedge xy}[X,Y]\right) &= \operatorname{rk}\begin{pmatrix} \frac{1}{M_G[y,x]}\cdot M_G[y,Y] \\ \frac{\sigma(1)}{M_G[x,y]}\cdot M_G[x,Y] \\ M_G[X',Y] - \frac{M_G[X',x]\cdot M_G[y,Y]}{M_G[y,x]} - \frac{M_G[X',y]\cdot M_G[x,Y]}{M_G[x,y]} \end{pmatrix} \\ &= \operatorname{rk}\begin{pmatrix} \frac{1}{M_G[y,x]}\cdot M_G[y,Y] \\ \frac{\sigma(1)}{M_G[x,y]}\cdot M_G[x,Y] \\ M_G[X',Y] - \frac{M_G[X',x]\cdot M_G[y,Y]}{M_G[y,x]} \end{pmatrix} \\ &= \operatorname{rk}\begin{pmatrix} \frac{1}{M_G[y,x]}\cdot M_G[y,Y] \\ \frac{\sigma(1)}{M_G[x,y]}\cdot M_G[x,Y] \\ M_G[X',Y] \end{pmatrix} = \operatorname{rk}\begin{pmatrix} M_G[y,Y] \\ M_G[x,Y] \\ M_G[X',Y] \end{pmatrix} \\ &= \operatorname{rk}\left(M_G[X,Y]\right). \end{aligned}$$

If $y \notin X$, then (with $X' := X \setminus \{x\}$ and $Y' := Y \setminus \{y\}$)

$$\begin{aligned} &\operatorname{rk}\left(M_{G\wedge xy}[X,Y]\right) \\ &\quad= \operatorname{rk}\begin{pmatrix} -\frac{1}{M_G[y,x]} & \frac{M_G[y,Y']}{M_G[y,x]} \\ \frac{M_G[X',x]}{M_G[y,x]} & M_G[X',Y'] - \frac{M_G[X',x]\cdot M_G[y,Y']}{M_G[y,x]} - \frac{M_G[X',y]\cdot M_G[x,Y']}{M_G[x,y]} \end{pmatrix} \\ &\quad= \operatorname{rk}\begin{pmatrix} -\frac{1}{M_G[y,x]} & \frac{M_G[y,Y']}{M_G[y,x]} \\ 0 & M_G[X',Y'] - \frac{M_G[X',y]\cdot M_G[x,Y']}{M_G[x,y]} \end{pmatrix} \\ &\quad= \operatorname{rk}\begin{pmatrix} -\frac{1}{M_G[y,x]} & 0 \\ 0 & M_G[X',Y'] - \frac{M_G[X',y]\cdot M_G[x,Y']}{M_G[x,y]} \end{pmatrix} \\ &\quad= \operatorname{rk}\begin{pmatrix} M_G[x,y] & 0 \\ 0 & M_G[X',Y'] - \frac{M_G[X',y]\cdot M_G[x,Y']}{M_G[x,y]} \end{pmatrix} \\ &\quad= \operatorname{rk}\begin{pmatrix} M_G[x,y] & 0 \\ M_G[X',y] & M_G[X',Y'] - \frac{M_G[X',y]\cdot M_G[x,Y']}{M_G[x,y]} \end{pmatrix} \\ &\quad= \operatorname{rk}\begin{pmatrix} M_G[x,y] & M_G[x,Y'] \\ M_G[X',y] & M_G[X',Y'] \end{pmatrix} \\ &\quad= \operatorname{rk}\left(M_G[X,Y]\right). \end{aligned}$$ □

The main result of this section is the following which is a generalization of Theorem [20, Theorem 5.4].

Theorem 1. *(i) For each positive integer $k \geq 1$, there is a set $\mathscr{C}_k^{(\mathbb{F},\sigma)}$ of σ-symmetric $\mathbb{F}^*$-graphs, each having at most $(6^{k+1}-1)/5$ vertices, such that a σ-symmetric $\mathbb{F}^*$-graph G has $\mathbb{F}$-rank-width at most k if and only if no σ-symmetric $\mathbb{F}^*$-graph in $\mathscr{C}_k^{(\mathbb{F},\sigma)}$ is isomorphic to a pivot-minor of G.*

(ii) Suppose that a σ-compatible $\lambda \in \mathbb{F}^*$ exists. Then for each positive integer $k \geq 1$, there is a set ${\mathscr{C}'}_k^{(\mathbb{F},\sigma)}$ of σ-symmetric $\mathbb{F}^*$-graphs, each having at most $(6^{k+1}-1)/5$ vertices, such that a σ-symmetric $\mathbb{F}^*$-graph G has $\mathbb{F}$-rank-width at most k if and only if no σ-symmetric $\mathbb{F}^*$-graph in ${\mathscr{C}'}_k^{(\mathbb{F},\sigma)}$ is isomorphic to a σ-vertex-minor of G.

Note that $\mathscr{C}_k^{(\mathbb{F},\sigma)}$ and ${\mathscr{C}'}_k^{(\mathbb{F},\sigma)}$ are finite when $\mathbb{F}$ is finite. The most important ingredients for proving Theorem 1 are Propositions 4 and 5, and Lemmas 10 and 11. All the other ingredients are already proved in [14,20] except that they are stated for the connectivity function of matroids in [14] and for undirected graphs in [20]. Their proofs rely only on the fact that the parameter is symmetric, submodular and integer valued. (Proofs are omitted because of space constraints.)

Proposition 4. *Let G be a σ-symmetric $\mathbb{F}^*$-graph, λ a σ-compatible element in $\mathbb{F}^*$ and x a vertex of G. For every subset X of $V_G \setminus \{x\}$,*

$$\mathrm{cutrk}^{\mathbb{F}}_{(G*(x,\lambda))\text{-}x}(X) = \mathrm{rk}\begin{pmatrix} -1 & M_G[x, V_G\backslash(X\cup x)] \\ M_G[X,x] & M_G[X, V_G\backslash(X\cup x)]\end{pmatrix} - 1.$$

The following lemma is thus the counterpart of [20, Lemma 4.4] and [14, Proposition 3.2].

Lemma 10. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and x a vertex in V_G. Assume that (X_1, X_2) and (Y_1, Y_2) are partitions of $V_G\backslash\{x\}$. Then,*

$$\mathrm{cutrk}^{\mathbb{F}}_{G\text{-}x}(X_1) + \mathrm{cutrk}^{\mathbb{F}}_{(G*(x,\lambda))\text{-}x}(Y_1) \geq \mathrm{cutrk}^{\mathbb{F}}_{G}(X_1\cap Y_1) + \mathrm{cutrk}^{\mathbb{F}}_{G}(X_2\cap Y_2) - 1.$$

Similarly, we get the followings for pivot-minor.

Proposition 5. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and xy an edge of G. For every subset X of $V_G \setminus \{x\}$,*

$$\mathrm{cutrk}^{\mathbb{F}}_{(G\wedge xy)\text{-}x}(X) = \mathrm{rk}\begin{pmatrix} 0 & M_G[x, V_G\backslash(X\cup x)] \\ M_G[X,x] & M_G[X, V_G\backslash(X\cup x)]\end{pmatrix} - 1.$$

Lemma 11. *Let G be a σ-symmetric $\mathbb{F}^*$-graph and xy an edge in V_G. Assume that (X_1, X_2) and (Y_1, Y_2) are partitions of $V_G\backslash\{x\}$. Then*

$$\mathrm{cutrk}^{\mathbb{F}}_{G\text{-}x}(X_1) + \mathrm{cutrk}^{\mathbb{F}}_{(G\wedge xy)\text{-}x}(Y_1) \geq \mathrm{cutrk}^{\mathbb{F}}_{G}(X_1\cap Y_1) + \mathrm{cutrk}^{\mathbb{F}}_{G}(X_2\cap Y_2) - 1.$$

3.3 Recognizing $\mathbb{F}$-Rank-Width at Most k

We give in this section a cubic-time algorithm that decides whether a $\mathbb{F}^*$-graph has $\mathbb{F}$-rank-width at most k, for fixed finite field $\mathbb{F}$ and a fixed k. This algorithm is an easy corollary of the one by Hliněný and Oum concerning representable matroids [15]. We refer to Schrijver [23] for our matroid terminology. We recall that if $\mathcal{M} := (S, \mathcal{I})$ is a matroid and $r_\mathcal{M}$ its rank-function, we let $\lambda_\mathcal{M}(U) := r_\mathcal{M}(U) + r_\mathcal{M}(S \backslash U) - r_\mathcal{M}(S) + 1$ for every $U \subseteq S$. This function is symmetric and submodular. The *branch-width* of $\mathcal{M}$ is the $\lambda_\mathcal{M}$-width of S. If $\mathcal{P}$ is a partition of S, we let $\lambda_\mathcal{M}^\mathcal{P}(Z) := \lambda_\mathcal{M}(\bigcup_{Y \in Z} Y)$ for any $Z \subseteq \mathcal{P}$. $\lambda_\mathcal{M}^\mathcal{P}$ is clearly symmetric and submodular. We recall the following important result by Hliněný and Oum [15].

Theorem 2 ([15]). *Let $\mathbb{F}$ be a fixed finite field, and k be a fixed positive integer. There exists a cubic-time algorithm that takes as input a representable matroid $\mathcal{M} = (S, \mathcal{I})$ given with its representation over $\mathbb{F}$ and a partition $\mathcal{P}$ of S, and outputs a layout of $\mathcal{P}$ of $\lambda_\mathcal{M}^\mathcal{P}$-width at most k or confirms that all layouts of $\mathcal{P}$ have $\lambda_\mathcal{M}^\mathcal{P}$-width at least $k+1$.*

We can now derive our recognition algorithm from Theorem 2. For a set X, we let X' be a disjoint copy of it defined as $\{x' \mid x \in X\}$. For G a $\mathbb{F}^*$-graph, we let $\mathcal{M}_G$ be the matroid on $V_G \cup V_G'$ represented by the $(V_G, V_G \cup V_G')$-matrix (I_n denotes the $n \times n$ identity matrix):

$$\begin{array}{cc} & \begin{array}{cc} V_G & V_G' \end{array} \\ V_G & \begin{pmatrix} I_{|V_G|} & M_G \end{pmatrix} \end{array}$$

For each $x \in V$, we let $P_x := \{x, x'\}$ and we let $\Pi(G) := \{P_x \mid x \in V_G\}$.

Proposition 6. *Let G be a σ-symmetric $\mathbb{F}^*$-graph. For every $X \subseteq V_G$,*

$$\lambda_{\mathcal{M}_G}^{\Pi(G)}(P) = 2 \cdot \mathrm{cutrk}_G^\mathbb{F}(X) + 1$$

where $P := \{P_x \mid x \in X\}$.

Proof. Since when G is σ-symmetric, we have:

$$\mathrm{rk}(M_G[X, V_G \backslash X]) = \mathrm{rk}(M_G[V_G \backslash X, X]) = \mathrm{cutrk}_G^\mathbb{F}(X),$$

it is sufficient to prove the following:

$$\lambda_{\mathcal{M}_G}^{\Pi(G)}(P) = \mathrm{rk}(M_G[X, V_G \backslash X]) + \mathrm{rk}(M_G[V_G \backslash X, X]) + 1.$$

We have:

$$\begin{aligned} \lambda_{\mathcal{M}_G}^{\Pi(G)}(P) &= r_{\mathcal{M}_G}(X \cup X') + r_{\mathcal{M}_G}(V_G \backslash X \cup (V_G \backslash X)') - r_{\mathcal{M}_G}(V_G \cup V_G') + 1 \\ &= rk \begin{pmatrix} 0 & M_G[V_G \backslash X, X] \\ I_{|X|} & M_G[X, X] \end{pmatrix} + rk \begin{pmatrix} 0 & M_G[X, V_G \backslash X] \\ I_{|V_G| - |X|} & M_G[V_G \backslash X, V_G \backslash X] \end{pmatrix} - |V_G| + 1 \\ &= |X| + \mathrm{rk}(M_G[V_G \backslash X, X]) + |V_G - X| + \mathrm{rk}(M_G[X, V_G \backslash X]) - |V_G| + 1 \\ &= \mathrm{rk}(M_G[X, V_G \backslash X]) + \mathrm{rk}(M_G[V_G \backslash X, X]) + 1. \end{aligned}$$

□

Theorem 3 (Checking $\mathbb{F}$-Rank-Width at most k). *For fixed k and a fixed finite field $\mathbb{F}$, there exists a cubic-time algorithm that, for a σ-symmetric $\mathbb{F}^*$-graph G, either outputs a layout of V_G of $\mathrm{cutrk}_G^{\mathbb{F}}$-width at most k or confirms that the $\mathbb{F}$-rank-width of G is larger than k.*

Proof. Let k be fixed and let $\mathcal{A}$ be the algorithm constructed in Theorem 2 for $2k+1$. Let G be a σ-symmetric $\mathbb{F}^*$-graph. We run the algorithm $\mathcal{A}$ with input $(\mathcal{M}_G, \Pi(G))$. If it outputs a layout of $\Pi(G)$ of $\lambda_{\mathcal{M}_G}^{\Pi(G)}$-width at most $2k+1$, we can transform it into a layout of V_G of $\mathrm{cutrk}_G^{\mathbb{F}}$-width at most k. Otherwise, we can confirm that the $\mathbb{F}$-rank-width of G is greater than k (Proposition 6). The fact that the algorithm $\mathcal{A}$ runs in cubic-time concludes the proof. $\square$

3.4 Specialisations to Graphs without Edge-Colors

We specialise in this section the $\mathbb{F}$-rank-width to directed and oriented graphs without edge-colors. As we already said, for undirected graphs without edge-colors, the $\mathbb{F}_2$-rank-width matches with the known rank-width of undirected graphs.

Directed Graphs. We recall that the adjacency matrix of a directed graph G is the (V_G, V_G)-matrix M_G over $\mathbb{F}_2$ where $M_G[x,y] := 1$ if and only if $(x,y) \in E_G$. This matrix is not symmetric except when G is undirected. In particular, $\mathrm{rk}(M_G[X, V_G \backslash X])$ is *a priori* different from $\mathrm{rk}(M_G[V_G \backslash X, X])$. The quest for finding another representation of directed graphs by matrices where $\mathrm{rk}(M_G[X, V_G \backslash X]) = \mathrm{rk}(M_G[V_G \backslash X, X])$ motivates the definition of sigma-symmetry. We now give this representation.

We recall that $\mathbb{F}_4$ is the finite field of order four. Let $\{0, 1, \partial, \partial^2\}$ be its elements with the property that $1+\partial+\partial^2 = 0$ and $\partial^3 = 1$. Moreover, it is of characteristic 2. Let $\sigma_4 : \mathbb{F}_4 \to \mathbb{F}_4$ be the automorphism where $\sigma_4(\partial) = \partial^2$ and $\sigma_4(\partial^2) = \partial$. It is clearly a sesqui-morphism.

For every directed graph G, let $\widetilde{G} := (V_G, E_G \cup \{(y,x) | (x,y) \in E_G\}, \ell_G)$ be the ${\mathbb{F}_4}^*$-graph where for every pair of vertices (x,y):

$$\ell_{\widetilde{G}}((x,y)) := \begin{cases} 1 & \text{if } (x,y) \in E_G \text{ and } (y,x) \in E_G, \\ \partial & (x,y) \in E_G \text{ and } (y,x) \notin E_G, \\ \partial^2 & (y,x) \in E_G \text{ and } (x,y) \notin E_G, \\ 0 & \text{otherwise.} \end{cases}$$

It is straightforward to verify that $\widetilde{G}$ is σ_4-symmetric. The rank-width of a directed graph G, denoted by $\mathrm{rwd}^{\mathbb{F}_4}(G)$, is the $\mathbb{F}_4$-rank-width of $\widetilde{G}$. Note that for an undirected graph G, we have clearly $\mathrm{rwd}^{\mathbb{F}_4}(G) = \mathrm{rwd}(G)$, since $\mathbb{F}_4$ is an extension of $\mathbb{F}_2$.

We now specialise the notion of σ-vertex-minor. We recall that given a sesqui-morphism $\sigma : \mathbb{F} \to \mathbb{F}$, an element λ of $\mathbb{F}^*$ is said σ-compatible if $\sigma(\lambda) = \lambda \cdot \sigma(1)^2$. Since $\sigma_4(1) = 1$, 1 is σ_4-compatible and is the only one. We then denote $G*(v,1)$ by $G*v$, and say that a directed graph H is a *vertex-minor* of a directed graph

G if $\widetilde{H}$ is a σ-vertex-minor of $\widetilde{G}$. One easily verifies that if a directed graph H is obtained from a directed graph G by applying a 1-local-complementation at x, then H is obtained from G by modifying the subgraph induced on the neighbours of x as shown on Table 1.

Table 1. We use the following notations: $x \rightarrow y$ means $\ell_{\widetilde{G}}((x,y)) = \eth$, $x \leftarrow y$ means $\ell_{\widetilde{G}}((x,y)) = \eth^2$, $x \leftrightarrow y$ means $\ell_{\widetilde{G}}((x,y)) = 1$, and $z \perp t$ means $\ell_{\widetilde{G}}((x,y) = 0)$.
(a) Uniform Case: $z \leftarrow x \rightarrow t$ or $z \rightarrow x \leftarrow t$ or $z \leftrightarrow x \leftrightarrow t$.
(b) Non Uniform Case: $z \leftarrow x \leftarrow t$ or $z \rightarrow x \leftrightarrow t$ or $z \leftrightarrow x \rightarrow t$.

G	$G * x$
$z \perp t$	$z \leftrightarrow t$
$z \rightarrow t$	$z \leftarrow t$
$z \leftarrow t$	$z \rightarrow t$
$z \leftrightarrow t$	$z \perp t$

(a)

G	$G * x$
$z \perp t$	$z \rightarrow t$
$z \rightarrow t$	$z \perp t$
$z \leftarrow t$	$z \leftrightarrow t$
$z \leftrightarrow t$	$z \leftarrow t$

(b)

Moreover, as in the undirected case, we have $G \wedge xy = G \wedge yx = G * x * y * x = G * y * x * y$. As corollaries of Theorems 1 and 3 we get that directed graphs of rank-width at most k can be recognised by a cubic-time algorithm, and are characterised by a finite list of directed graphs to exclude as vertex-minors.

Oriented Graphs. We can define another parameter in the case of oriented graphs. Let $G = (V, A)$ be an oriented graph, and let $\widetilde{G} = (V, E, \ell)$ be the $\mathbb{F}_3^*$-graph such that $E = A \cup A'$ where $A' = \{(y,x) | (x,y) \in A\}$, $\ell((x,y)) := 1$ if $(x,y) \in A$ and $\ell((x,y)) := -1$ if $(x,y) \in A'$. Clearly, $\widetilde{G}$ is a σ-symmetric $\mathbb{F}_3^*$-graph, with $\sigma(x) := -x$. Moreover, one can show immediately that σ is a sesqui-morphism. Note that there is no σ-compatible λ in $\mathbb{F}_3^*$, thus no σ-local-complementation is defined on σ-symmetric $\mathbb{F}_3^*$-graphs. Nevertheless, oriented graphs of $\mathbb{F}_3$-rank-width k are characterized by a finite set of oriented graphs $\mathscr{C}_k^{(\mathbb{F}_3,\sigma)}$ of forbidden pivot-minors (whereas sets $\mathscr{C}_k^{(\mathbb{F}_4,\sigma)}$ and $\mathscr{C}'^{(\mathbb{F}_4,\sigma)}_k$ contains directed graphs).

$\mathbb{F}_3$-rank-width and $\mathbb{F}_4$-rank-width of oriented graphs are two equivalent parameters (*i.e* the $\mathbb{F}_3$-rank-width of a graph class $\mathcal{G}$ is bounded if and only if the $\mathbb{F}_4$-rank-width of $\mathcal{G}$ is bounded), since they are both equivalent to the clique width.

4 Algebraic Operations for $\mathbb{F}$-Rank-Width

Graph operations that generalise the ones in [5] and that characterise exactly $\mathbb{F}$-rank-width are given here. We let $\mathbb{F}$ be a fixed finite field along this section. For a fixed positive integer k, we let $\mathbb{F}^k$ be the set of row vectors of length k.

We let $\sigma : \mathbb{F} \rightarrow \mathbb{F}$ be a fixed sesqui-morphism. If $u := (u_1, \ldots, u_k) \in \mathbb{F}^k$, we let $\sigma(u)$ be $(\sigma(u_1), \ldots, \sigma(u_k))$. Similarly, if $M = (m_{i,j})$ is a matrix, we let $\sigma(M)$ be the matrix $(\sigma(m_{i,j}))$. Throughout this section, we will say graph instead of σ-symmetric $\mathbb{F}^*$-graph to avoid overloading the text.

An $\mathbb{F}^k$-*coloring* of a graph G is a mapping $\gamma_G : V_G \to \mathbb{F}^k$ with no constraint on the values of γ for adjacent vertices. An $\mathbb{F}^k$-*colored graph* G is a tuple $(V_G, E_G, \ell_G, \gamma_G)$ where (V_G, E_G, ℓ_G) is a graph and γ_G is an $\mathbb{F}^k$-coloring of (V_G, E_G, ℓ_G). Notice that an $\mathbb{F}^k$-colored graph has not only its edges colored with colors from $\mathbb{F}$, but also its vertices with colors from $\mathbb{F}^k$.

The following is a binary graph operation that combines several operations consisting in adding colored edges between its disjoint arguments and recolor them independently.

Definition 3 (Bilinear Products). *Let k, ℓ and m be positive integers and let M, N and P be $k \times \ell$, $k \times m$ and $\ell \times m$ matrices, respectively, over $\mathbb{F}$. For an $\mathbb{F}^k$-colored graph G and an $\mathbb{F}^\ell$-colored graph H, we let $G \otimes_{M,N,P} H$ be the $\mathbb{F}^m$-colored graph $K := (V_G \cup V_H, E_G \cup E_H \cup E', \ell_K, \gamma_K)$ where:*

$$E' := \{xy \mid x \in V_G,\ y \in V_H \textit{ and } \gamma_G(x) \cdot M \cdot \sigma(\gamma_H(y))^T \neq 0\},$$

$$\ell_K((x,y)) := \begin{cases} \ell_G((x,y)) & \textit{if } x, y \in V_G, \\ \ell_H((x,y)) & \textit{if } x, y \in V_H, \\ \gamma_G(x) \cdot M \cdot \sigma(\gamma_H(y))^T & \textit{if } x \in V_G,\ y \in V_H, \\ \sigma\left(\gamma_G(y) \cdot M \cdot \sigma(\gamma_H(x))^T\right) & \textit{if } y \in V_G,\ x \in V_H. \end{cases}$$

$$\gamma_K(x) := \begin{cases} \gamma_G(x) \cdot N & \textit{if } x \in V_G, \\ \gamma_H(x) \cdot P & \textit{if } x \in V_H. \end{cases}$$

Definition 4 (Constants). *For each $u \in \mathbb{F}^k$, we let $\mathbf{u}$ be a constant denoting a $\mathbb{F}^k$-colored graph with one vertex colored by u and no edge.*

We denote by $\mathcal{C}_n^{\mathbb{F}}$ the set $\{\mathbf{u} \mid u \in \mathbb{F}^1 \cup \cdots \cup \mathbb{F}^n\}$. We let $\mathcal{R}_n^{(\mathbb{F},\sigma)}$ be the set of bilinear products $\otimes_{M,N,P}$ where M, N and P are respectively $k \times \ell$, $k \times m$ and $\ell \times m$ matrices for $k, \ell, m \leq n$. Each term t in $T(\mathcal{R}_n^{(\mathbb{F},\sigma)}, \mathcal{C}_n^{\mathbb{F}})$ defines, up to isomorphism, a graph $val(t)$.

One easily verifies that the operations $\otimes_{M,N,P}$ can be defined in terms of the disjoint union and quantifier-free operations. The following is thus a corollary of results in [6].

Theorem 4. *For each monadic second-order property φ, there exists an algorithm that checks for every term $t \in T(\mathcal{R}_n^{(\mathbb{F},\sigma)}, \mathcal{C}_n^{\mathbb{F}})$, in time $O(|t|)$, if the graph defined by this term, up to isomorphism, satisfies φ.*

The principal theorem of this section is the following. Its proof is omitted because of space constraints.

Theorem 5. *A graph G has $\mathbb{F}$-rank-width at most n if and only if it is isomorphic to $val(t)$ for some term t in $T(\mathcal{R}_n^{(\mathbb{F},\sigma)}, \mathcal{C}_n^{\mathbb{F}})$.*

The following is thus a corollary.

Theorem 6 ([5]). *An undirected graph has rank-width at most n if and only if it is isomorphic to $val(t)$ for some term t in $T(\mathcal{R}_n^{(\mathbb{F}_2,\sigma_1)}, \mathcal{C}_n^{\mathbb{F}_2})$.*

5 Conclusion

We extended the rank-width of undirected graphs and some related results to the C-graphs. Presented results imply in particular that every MSOL-definable property can be checked in polynomial time on C-graphs, when C is finite, and C-graphs of bounded $\mathbb{F}$-rank-width are characterised by a finite list of C-graphs to exclude as pivot-minors. We notice that the first author prove in [17] that graphs of bounded $\mathbb{F}$-rank-width are *well-quasi-ordered* by the pivot-minor relation. Every open question in the undirected case is of course still relevant for the C-graphs.

Recently, some authors investigated the clique-width of multigraphs [3] or weighted graphs [13]. These graphs can be seen as $\mathbb{N}$-graphs. It is straightforward to verify that the rank-width is not equivalent to the clique-width when C is infinite. It would be interesting to investigate the rank-width over an infinite field, and in particular its algorithmic aspects: the recognition of C-graphs of bounded rank-width, and the property checking on C-graphs of bounded rank-width.

References

1. Bouchet, A.: Digraph Decompositions and Eulerian Systems. SIAM Journal on Algebraic and Discrete Methods 8(3), 323–337 (1987)
2. Blumensath, A., Courcelle, B.: Recognizability, Hypergraph Operations and Logical Types. Information and Computation 204(6), 853–919 (2006)
3. Courcelle, B.: On the Model-Checking of Monadic Second-Order Formulas with Edge Set Quantifications. Discrete Applied Mathematics doi:10.1016/j.dam.2010.12.017 (in press)
4. Courcelle, B.: Graph Structure and Monadic Second-Order Logic. Book in preparation. Cambridge University Press, Cambridge
5. Courcelle, B., Kanté, M.M.: Graph Operations Characterizing Rank-Width. Discrete Applied Mathematics 157(4), 627–640 (2009)
6. Courcelle, B., Makowsky, J.A.: Fusion in Relational Structures and the Verification of Monadic Second-Order Properties. Mathematical Structures in Computer Science 12, 203–235 (2002)
7. Courcelle, B., Olariu, S.: Upper Bounds to the Clique-Width of Graphs. Discrete Applied Mathematics 101(1-3), 77–114 (2000)
8. Courcelle, B., Oum, S.: Vertex-Minors, Monadic Second-Order Logic and a Conjecture by Seese. Journal of Combinatorial Theory, Series B 97(1), 91–126 (2007)
9. Cunningham, W.H.: Decomposition of Directed Graphs. SIAM Journal on Algebraic and Discrete Methods 3(2), 214–228 (1982)
10. Diestel, R.: Graph Theory, 3rd edn. Springer, Heidelberg (2005)
11. Ehrenfeucht, A., Harju, T., Rozenberg, G.: The Theory of 2-Structures: A Framework for Decomposition and Transformation of Graphs. World Scientific, Singapore (1999)
12. Fisher, E., Makowsky, J.A., Ravve, E.V.: Counting Truth Assignments of Formulas of Bounded Tree-Width or Clique-Width. Discrete Applied Mathematics 156(4), 511–529 (2008)

13. Flarup, U., Lyaudet, L.: On the Expressive Power of Permanents and Perfect Matchings of Matrices of Bounded Path-Width/Clique-Width. Theory of Computing Systems 46(4), 761–791 (2010)
14. Geelen, J.F., Gerards, A.M.H., Robertson, N., Whittle, G.P.: On the Excluded Minors for the Matroids of Branch-Width k. Journal of Combinatorial Theory, Series B 88(2), 261–265 (2003)
15. Hliněný, P., Oum, S.: Finding Branch-Decompositions and Rank-Decompositions. SIAM Journal on Computing 38(3), 1012–1032 (2008)
16. Kaminski, M., Lozin, V., Milanic, M.: Recent Developments on Graphs of Bounded Clique-Width. Discrete Applied Mathematics 157(12), 2747–2761 (2009)
17. Kanté, M.M.: Well-Quasi-Ordering of Matrices under Principal Pivot Transforms, arxiv:1102.2134 (2011) revisited (Submitted)
18. Kanté, M.M., Rao, M.: Directed Rank-Width and Displit Decomposition. In: Habib, M., Paul, C. (eds.) WG 2009. LNCS, vol. 5911, pp. 214–225. Springer, Heidelberg (2010)
19. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopedia of Mathematics and its Applications, 2nd edn (1997)
20. Oum, S.: Rank-Width and Vertex-Minors. Journal of Combinatorial Theory, Series B 95(1), 79–100 (2005)
21. Oum, S., Seymour, P.D.: Approximating Clique-Width and Branch-Width. Journal of Combinatorial Theory, Series B 96(4), 514–528 (2006)
22. Robertson, N., Seymour, P.D.: Graph minors V:Excluding a Planar Graph. Journal of Combinatorial Theory, Series B 41, 92–114 (1986)
23. Schrijver, A.: Combinatorial Optimization, Polyhedra and Efficiency, vol. B. Springer, Heidelberg (2003)

An Algorithm for Computing a Basis of a Finite Abelian Group

Gregory Karagiorgos[1] and Dimitrios Poulakis[2]

[1] Department of Technology of Informatics and Telecommunications,
T.E.I of Kalamata / Branch of Sparta, Kladas, 23100 Sparta, Greece
greg@teikal.gr, greg@di.uoa.gr

[2] Aristotle University of Thessaloniki, Department of Mathematics,
Thessaloniki 54124, Greece
poulakis@math.auth.gr

Abstract. In this paper we consider the problem of computation of a basis for a finite abelian group G with N elements. We present a deterministic algorithm such that given a generating set for G and the prime factorization of N, it computes a basis of G.

Keywords: Abelian group, generating system, basis of abelian group.

1 Introduction

In recent years, interest in studying finite abelian groups has raised due to the increasing significant of its relationship with public key cryptography, quantum computing and error-correcting codes. Abelian groups as the groups $\mathbb{Z}_n^*$ of invertible elements of $\mathbb{Z}_n$, the multiplicative groups of finite fields, the groups of elements of elliptic curves over finite fields, the groups of Jacobian varieties of hyperelliptic curves, the class groups of quadratic fields and others have been used for the specification of public key cryptosystems [13]. On the other hand, in quantum computing, the famous hidden subgroup problem in case of a finite abelian group has been solved by a polynomial time quantum algorithm [6], [12], [16]. The Shor's algorithm to factorize integers is one very important special case [17]. Recently, an interesting application of finite abelian groups has been given in the construction of efficient error correcting codes [7].

A finite abelian group can be decomposed to a direct sum of cyclic groups with prime-power order. A set which consists of exactly one generator from each of those cyclic groups form a basis of the abelian group. The elements of a basis of a finite abelian group and their orders fully determine its structure. Therefore, the development of efficient algorithms for this task has fundamental significance in all the above applications.

In [4], Chen gave an $O(N^2)$ time algorithm for finding a basis of a finite abelian group G. Recently, in [5], Chen and Fu showed two $O(N)$ time algorithms for this task. In case where G is represented by an explicit set of M generators, a $O(MN^{1/2+\epsilon})$ time algorithm is given by Iliopoulos [10] and $O(MN^{1/2})$ time algorithm is obtained by Buchmann and Schmidt [3]. Also, Teske [19] gave an algorithm with expected running time $O(MN^{1/2})$. Moreover, a randomized algorithm for this task is proposed in [5]. When G is represented by a set of defining relations that is associated with an integer matrix $M(G)$, the computation of the structure of G can be reduced to computing the Smith Normal Form of $M(G)$. One such approach can be found in [11]. Finally, in [2], an algorithm is given for computing the structure of G, as well as a set of generators of G, based on Gröbner bases techniques.

In this paper we obtain a deterministic algorithm for the computation of a basis of a finite abelian group G with N elements in case where a generating system of G with M elements and the prime factorization of N are given. We express the time complexity of our algorithm in terms of M and the prime factorization of N. This has the advantage to relate the time complexity of the algorithm more closely with the structure of the finite abelian group. Note that the complexity of our deterministic algorithm is very close to the complexity of the randomized algorithm obtained in [5].

The paper is organized as follows. In Section 2 we recall some definitions from Group Theory. Section 3 is devoted to the presentation of our results. In Section 4, we give some auxiliary algorithms which are necessary for the presentation of our algorithms. In Section 5, we present an algorithm for the computation of the order of an element and the order of its images in the directed factors of a finite abelian group. The proof of Theorem 1 is given in Section 6. Section 7 contains the proofs of Corollaries 1 and 2. Finally, the last section concludes the paper.

2 Preliminaries

In this section we recall some definitions from Group Theory.

1. Let $(G,+)$ be a finite abelian group. We denote by $|G|$ the cardinality of a group G. For $x \in G$, the *order* of x, denoted by $\mathrm{ord}(x)$, is the smallest positive integer $k \geq 1$ such that $kx = 0$, where 0 is the the neutral element of G.
2. Let x be an element of a group G such that $\mathrm{ord}(x) = k$. The set
$$< x >= \{x, 2x, 3x, \ldots, kx = 0\}$$
is a subgroup of G called the *cyclic* group generated by x.
3. Let $H_1, \ldots, H_r$ be subgroups of G. The set
$$H_1 + \cdots + H_r = \{x_1 + \cdots + x_r / \ x_i \in H_i, \ i = 1, \ldots, r\}$$
is a subgroup of G called the *sum* of $H_1, \ldots, H_r$.
4. If for every $i = 1, \ldots, r$, we have
$$H_i \cap (H_1 + \cdots + H_{i-1} + H_{i+1} + \cdots + H_r) = \{0\},$$
then the set $H = H_1 + \cdots + H_r$ is called *direct sum* of $H_1, \ldots, H_r$ and in this case we write
$$H = H_1 \oplus \cdots \oplus H_r.$$

5. Let $S \subseteq G$. The group
$$< S >= \sum_{x \in S} < x >$$
is called the *group generated* by S. In case where $G =< S >$, the set S is called a *generating system* for G.
6. Suppose now that G has N elements and
$$N = p_1^{a_1} \cdots p_k^{a_k}$$
is the prime factorization of N. It is well known that any finite Abelian group G of order $N > 1$, can be represented as
$$G \cong G(p_1) \oplus \cdots \oplus G(p_k),$$
where $G(p_i)$ is a subgroup of G of order $p_i^{a_i}$ $(i = 1, \ldots, k)$ [15, Theorem 16, page 96] and is called the *p_i-primary component* of G.
7. Furthermore, each $G(p_i)$ can be decomposed to a direct sum of cyclic groups
$$G(p_i) \cong < x_{i,1} > \oplus \cdots \oplus < x_{i,\mu(p_i)} >$$
and the order of $x_{i,j}$ $(j = 1, \ldots, \mu(p_i))$ is a power of p_i.
The *set of elements* $x_{i,j}$ $(i = 1, \ldots, k,\ j = 1, \ldots, \mu(p_i))$ is called a *basis* of G.
8. The smallest prime power $p_i^{e(p_i)}$ such that $p_i^{e(p_i)} x = 0$, for every $x \in G(p_i)$ is called the *exponent* of $G(p_i)$.

Finally, if x is a real number, then we denote by $\lceil x \rceil$, as usually, the smallest integer z such that $x \leq z$.

We assume that for $a, b \in G$ we can compute $c = a + b$, we can test whether $a = b$ and for $a \in G$ we can compute $-a$. We call these *group operations.* Note that from every $a \in G$ we can compute the neutral element $0 = a + (-a)$. We assume that each group operation, arithmetic operation on $O(\log N)$ bits integers and comparison can be performed in constant time.

3 Our Results

In this section we present our results.

Theorem 1. *Let G be an abelian group with N elements. Suppose that a generating system with M elements for G and the prime factorization $N = p_1^{a_1} \cdots p_k^{a_k}$ of N are given. Then, there is a deterministic algorithm for computing a basis of G with time complexity*
$$O(M(\log N)^2/\log\log N + M\sum_{i=1}^{k} A(p_i) + \sum_{i=1}^{k} B(p_i)),$$
where
$$A(p_i) = e(p_i)\lceil p_i^{1/2} \rceil^{\mu(p_i)-1}, \quad B(p_i) = (a_i - e(p_i))(\log p_i^{a_i})^2 \log\log p_i^{a_i}.$$

The expression of the time complexity of our algorithm in terms of the prime factorization of N has the advantage to relate it more closely with the structure of G.

The basic idea of the proof of Theorem 1 is the following: We select elements of the generating system and computing their order and the orders of their images in $G(p)$, we construct bases of successively larger subgroups of every p-primary components $G(p)$ of G combining an algorithm of Teske [18] and an algorithm of Beynon and Iliopoulos [1], until a basis of $G(p)$ is obtained.

Next, we give two applications of Theorem 1. First, we present the important special case of cyclic groups.

Corollary 1. *Let G be a cyclic group with N elements. Suppose that a generating system with M elements for G and the prime factorization of N are given. Then, there is an algorithm for computing a basis of G with time complexity*

$$O(M(\log N)^2/\log\log N).$$

Our next application concerns the groups of elliptic curves over finite fields.

Corollary 2. *Let E be an elliptic curve over a finite field $\mathbb{F}_q$ and $E(\mathbb{F}_q)$ its group of points over $\mathbb{F}_q$. Let $|E(\mathbb{F}_q)| = N$. Suppose that a generating system with M elements for $E(\mathbb{F}_q)$ and the prime factorization of N are given. Then, there is an algorithm for computing a basis of $E(\mathbb{F}_q)$ in time*

$$O(MN^{1/4}\log N).$$

Note that there is an algorithm for the computation of a basis for $E(\mathbb{F}_q)$ with running time $O(q^{1/2+\epsilon})$ [14, Theorem 3]. Since $N = \Theta(q)$ [20, Theorem 4.2, page 91], the running time of this algorithm is $O(N^{1/2+\epsilon})$. So, in case when the computation of a generating system of $E(\mathbb{F}_q)$ with $M < N^{1/4}$ elements in time $O(N^{1/2})$ is possible, we obtain a faster algorithm.

When a generating set $\{x_1, \ldots, x_M\}$ for G and the prime factorization of $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$ are given, a randomized algorithm is proposed in [5] for the computation of a basis for G. Its time complexity is

$$O(M(\log N)^2 + \sum_{i=1}^{k} a_i p_i^{a_i/2})$$

which, as we see, is very close to the complexity of our deterministic algorithm.

We shall show that the hypothesis in [5] that the prime factorization of $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$ are given is equivalent to our hypothesis that the prime factorization of N is given. The integers $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$ are divisors of N and so, if the prime factorization of N is known, the prime factorizations of $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$ can be easily computed (see also Proposition 1 below). Conversely, consider a prime divisor p of N. Then there is $y \in G$ with $\mathrm{ord}(y) = p$. Further, there are $c_1, \ldots, c_M \in \mathbb{Z}$ such that

$$y = c_1x_1 + \cdots + c_Mx_M.$$

We denote by L the least common multiple of $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$. If for every $i \in \{1, \ldots, M\}$ we have $p \not| \mathrm{ord}(x_i)$, then $p \not| L$. On the other hand, we have $Ly = 0$ and hence $p|L$ which is a contradiction. Therefore, there is i such that $p|\mathrm{ord}(x_i)$. Thus, if the prime factorization of $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$ are known, then the prime divisors of N are known and so, one can easily compute their exponents in the prime factorization of N. Thus, the hypothesis that the prime factorization of $\mathrm{ord}(x_i)$ $(i = 1, \ldots, M)$ are given is equivalent to the hypothesis that the prime factorization of N is given.

4 Auxiliary Results

In this section we give two algorithms necessary for the presentation of our algorithms. We denote by G an abelian group with N elements.

The Extended Discrete Logarithm Problem

Suppose

$$B = \{b_1, \ldots, b_n\}$$

is a subset of G such that the group $H =< B >$ is the direct sum of the cyclic groups $< b_i >$ $(i = 1, \ldots, n)$.

The *extended discrete logarithm problem (EDLP)* is the following problem: Given a set $B \subseteq G$ as above and $w \in G$, determine the smallest positive integer z with $zw \in H$ and integers $z_1, \ldots, z_n$ with $0 \leq z_i < \mathrm{ord}(b_i)$ $(i = 1, \ldots, k)$ satisfying

$$zw = \sum_{i=1}^{n} z_i b_i.$$

Note that $z \leq \mathrm{ord}(w)$. If $z = \mathrm{ord}(w)$, then $H \cap < w >= \{0\}$. In [18], an algorithm is presented which solves the EDLP. We assume that the baby-step giant-step method is used to implement it [18, Remark 4.1, page 529]. The number of group operations needed for its application is

$$O(\max\{\lceil p^{1/2} \rceil^n e(p)\}),$$

where the maximum is taken over all prime divisors of N and $p^{e(p)}$ is the exponent of the p-component of G. It is called SOLVE-EDLP. Thus, we have

$$\text{SOLVE-EDLP}(w, B) = (z, z_1, \ldots, z_n).$$

The Basis Algorithm

Let p be a prime divisor of N and $G(p)$ the p-component of G. Suppose that

$$C = \{c_1, \ldots, c_n\}$$

is a subset of $G(p)$ such that the group $H =< C >$ is the direct sum of the cyclic groups $< c_i >$, $(i = 1, \dots, n)$. If $x \in G(p)$, then we denote by $H^\star$ the group generated by the set $C \cup \{x\}$. Suppose that the orders of elements of C are known and we have a relation of the form

$$p^l x = \sum_{i=1}^{n} \delta_i c_i,$$

where $l, \delta_i \in \mathbb{Z}$, $k \geq 0$ and $0 \leq \delta_i < \mathrm{ord}(c_i)$ $(i = 1, \dots, n)$. In [1], an algorithm is given for the computation of a basis $C^\star$ for $H^\star$ called BASIS which needs $O((\log |H^\star|)^2)$ group operations and $O((\log |H^\star|)^2 \log\log |H^\star|)$ arithmetic operations. Then we write

$$\mathrm{BASIS}(C, x, (p^l, \delta_1, \dots, \delta_n)) = C^\star.$$

5 Computing the Order of an Element

Let G be an abelian group with N elements and $N = p_1^{a_1} \cdots p_k^{a_k}$ the prime factorization of N. Let $x \in G$ and $m = \mathrm{ord}(x)$. By Lagrange's theorem [15, page 35], m divides N and so,

$$m = p_1^{b_1} \cdots p_k^{b_k}, \text{ where } 0 \leq b_i \leq a_i \quad (i = 1, \dots, k).$$

The following lemma gives the orders of elements $x_i = (m/p_i^{b_i})x$ $(i = 1, \dots, k)$ and the decomposition of $< x >$ in direct product of subroups of $G(p_i)$ $(i = 1, \dots, k)$.

Lemma 1. *If $x \in G$ has order $m = p_1^{b_1} \cdots p_k^{b_k}$, then for $i = 1, \dots, k$ the element $x_i = (m/p_i^{b_i})x$ has order $p_i^{b_i}$ and we have*

$$< x >=< x_1 > \oplus \cdots \oplus < x_k > .$$

Proof. See [15, page 96].

For the presentation of our algorithm for the computation of a basis of G in the next section, we need a procedure such that given $x \in G$, it computes m, the elements x_i $(i = 1, \dots, k)$ and theirs orders. This result is obtained in the following proposition.

Proposition 1. *Let G be an abelian group with N elements. Suppose that the prime factorization $N = p_1^{a_1} \cdots p_k^{a_k}$ of N is given. Then there is a deterministic algorithm such that for every $x \in G$ it computes its order $m = p_1^{b_1} \cdots p_k^{b_k}$, the elements $x_i = (m/p_i^{b_i})x$ $(i = 1, \dots, k)$ and theirs orders in time*

$$O((\log N)^2 / \log\log N).$$

Proof. The following algorithm achieves the above task.

Algorithm 1. ORDER (x)

```
% $N = p_1^{a_1} \dots p_k^{a_k}$
% Where $x \in G$
 1: $m = N$
 2: for $i = 1$ to $k$
 3:   for $b_i = 0$ to $a_i$
 4:     if $p_i^{b_i}\ (m/p_i^{a_i})x = 0$
 5:         break
 6:     end if
 7:   end for
 8:   $m = m/p_i^{a_i - b_i}$
 9:   end for
10: for $i = 1$ to $k$
11:    compute $m_i = m/p_i^{b_i}$.
12: end for
13: for $i = 1$ to $k$
14:    compute $x_i = m_i x$.
15: end for
16: return $[x, m, (x_1, p_1^{b_1}), \dots, (x_k, p_k^{b_k})]$
% Where $\mathrm{ord}(x) = m = p_1^{b_1} \cdots p_k^{b_k}$, $x_i = (m/p_i^{b_i})x$ and $\mathrm{ord}(x_i) = p_i^{b_i}$.
```

Proof of Correctness: In Steps 1-13, the algorithm computes the smallest positive integer m such that $mx = 0$ and so, $\mathrm{ord}(x) = m$. Since $m = p_1^{b_1} \cdots p_k^{b_k}$, then Lemma 1 implies that $\mathrm{ord}(x_i) = p_i^{b_i}$.

Proof of Time Complexity: For the computation of $p_i^{a_i}$, $m/p_i^{a_i}, p_i^{b_i}$ and m_i $(i = 1, \dots, k)$, the algorithm ORDER requires $O(a_1 + \dots + a_k)$ arithmetic operations. Since we have

$$a_1 + \dots + a_k = O(\log N),$$

the algorithm needs $O(\log N)$ arithmetic operations. For every $i = 1, \dots, k$ the computation of $m_i x$ needs $O(\log m_i)$ group operations and the computation of $p_i^j(m/p_i^{a_i})x$, $O(\log m)$ group operations [8, page 69]. By [9, page 355], we have

$$k = O(\log N / \log\log N).$$

Thus, we have $O((\log N)^2 / \log\log N)$ group operations. Hence the time complexity of ORDER is $O((\log N)^2 / \log\log N)$. $\square$

6 Proof of Theorem 1

In this section we devise an algorithm for computing a finite abelian group basis (FAGB). Our algorithm accepts as input an abelian group G with N elements, a generating system S of G with M elements and the prime factorization of N.

Algorithm 2. COMPUTE-FAGB(G, S, N)

```
% An abelian group $(G,+)$ with $|G| = N$,
% a generating system $S = \{g_1,\dots,g_M\}$ of $G$,
% and the prime factorization $N = p_1^{a_1}\cdots p_k^{a_k}$ of $N$.
 1: for $j = 1$ to $M$
 2:    compute ORDER$(g_j) = [g_j, m_j, (g_{j,1}, p_1^{b_{j,1}}), \dots, (g_{j,k}, p_k^{b_{j,k}})]$.
 3: end for
 4: for $i = 1$ to $k$
 5:    Sort the non-zero elements among $g_{1,i},\dots,g_{M,i}$ in decreasing order of quantities
       ord$(g_{j,i})$. These are the elements $\gamma_{1,i},\dots,\gamma_{m(i),i}$
       with ord$(\gamma_{1,i}) \geq \dots \geq$ ord$(\gamma_{m(i),i})$ and $m(i) \leq M$.
 6: end for
 7: for $i = 1$ to $k$
 8:    Set $B_{1,i} = \{\gamma_{1,i}\}$.
 9: end for
10: for $i = 1$ to $k$
11:   for $j = 2$ to $m(i)$
12:     if $| < B_{j-1,i} > | \neq p_i^{a_i}$
13:       Compute SOLVE-EDLP$(\gamma_{j,i}, B_{j-1,i}) = (z_{j,i}, z_{j,i,1}, \dots, z_{j,i,r(j,i)})$.
14:       Compute the largest integer $k_{j,i} \geq 0$ such that $p_i^{k_{j,i}}$ divides $z_{j,i}$
15:       if $k_{j,i} \neq 0$, then compute
16:          $s_{j,i} = z_{j,i}/p_i^{k_{j,i}}$ and $h_{j,i} = s_{j,i}\gamma_{j,i}$
17:          Compute BASIS$(B_{j-1,i}, h_{j,i}, (p_i^{k_{j,i}}, z_{j,i,1}, \dots, z_{j,i,r(j,i)})) = B_{j,i}$
18:       else $B_{j,i} = B_{j-1,i}$
19:       end if
20:     end if
21:   end for
22: end for
23: return For $i = 1,\dots,k$, output the couples $(y_{1,i}, n_{1,i}), \dots, (y_{l(i),i}, n_{l(i),i})$.
%Where $l(i)$ is the largest index $j$ for which $B_{j,i}$ is computed, $B_{l(i),i} = \{y_{1,i},\dots,y_{l(i),i}\}$
and ord$(y_{j,i}) = n_{j,i}$.
```

Proof of Correctness: Let $B_{j,i} = \{b_{j,i,1},\dots,b_{j,i,r(j,i)}\}$ $(i = 1,\dots,k,\ j = 1,\dots,m(i))$. For every $j = 1,\dots,m(i)$, the algorithm SOLVE-EDLP gives elements $z_{j,i}, z_{j,i,1},\dots,z_{j,i,n}$ such that

$$z_{j,i}\gamma_{j,i} = \sum_{s=1}^{r(j,i)} z_{j,i,s} b_{j,i,s}.$$

Let $k_{j,i}$ be the largest integer ≥ 0 such that $p_i^{k_{j,i}}$ divides $z_{j,i}$. Put $s_{j,i} = z_{j,i}/p_i^{k_{j,i}}$ and $h_{j,i} = s_{j,i}\gamma_{j,i}$. Thus, we have

$$p_i^{k_{j,i}} h_{j,i} = \sum_{s=1}^{r(j,i)} z_{j,i,s} b_{j,i,s}.$$

If $k_{j,i} = 0$, then $\gamma_{j,i} \in B_{j-1,i}$ and so, $B_{j,i} = B_{j-1,i}$. Otherwise, the algorithm BASIS applied on $B_{j-1,i}$, $h_{j,i}$, $p_i^{k_{j,i}}$ and $z_{j,i,1}, \ldots, z_{j,i,r(j,i)}$ gives the basis $B_{j,i}$ of $< \{\gamma_{j,i}\} \cup B_{j-1,i} >$. The procedure continuous until $| < B_{j,i} > | = p_i^{a_i}$ in which case $B_{j,i}$ is a basis of $G(p_i)$.

Proof of Time Complexity: We denote by $\mu(p_i)$ the number of cyclic subgroups which are direct factors of $G(p_i)$ and by $p_i^{e(p_i)}$ the exponent of $G(p_i)$.

In Steps 1-3 we apply the algorithm ORDER M times. Thus, the corresponding time complexity is $O(M(\log N)^2/\log\log N)$.

Steps 4-6 requires $O(kM\log M)$ comparisons. By [9, page 355], we have $k = O(\log N/\log\log N)$, and so we need $O(M(\log M)\log N/\log\log N)$ comparisons.

In Step 13, for $i = 1, \ldots, k$, we apply SOLVE-EDLP, provided that $\mu(p_i) > 1$, until a basis for $G(p_i)$ is obtained. Suppose that $|B_{j,i}| = \mu(p_i)$ and $< B_{j,i} >$ is generated by the elements $\gamma_{1,i}, \ldots, \gamma_{j,i}$. It follows that it contains all the elements of $G(p_i)$ having order $\leq \mathrm{ord}(\gamma_{1,i})$. Since we have

$$\mathrm{ord}(\gamma_{1,i}) \geq \ldots \geq \mathrm{ord}(\gamma_{m(i),i}),$$

the elements $\gamma_{j+1,i}, \ldots, \gamma_{m(i),i}$ belong to $< B_{j,i} >$ and so, we get $< B_{j,i} >= G(p_i)$. Hence, if $|B_{j,i}| = \mu(p_i)$, then $B_{j,i}$ is a basis of $G(p_i)$. So, we have applied SOLVE-EDLP in bases $B_{1,i}, \ldots, B_{j-1,i}$ which have at most $\mu(p_i) - 1$ elements. Thus, for $i = 1, \ldots, k$, Steps 10-13 need

$$O(e(p_i)M\lceil p_i^{1/2}\rceil^{\mu(p_i)-1})$$

group operations.

For $i = 1, \ldots, k$, Steps 14-16 require $O(M\log p_i^{a_i})$ group operations and $O(Ma_i)$ arithmetic operations.

Since we have $\mathrm{ord}(\gamma_{1,i}) \geq \mathrm{ord}(\gamma_{t,i})$ $(t = 2, \ldots, m(i))$, it follows that $B_{1,i} = \{\gamma_{1,i}\}$ and $\mathrm{ord}(\gamma_{1,i}) = p^{e(p_i)}$. For $i = 1, \ldots, k$ there are $a_i - e(p_i)$ subgroups of $G(p_i)$ containing $< B_{1,i} >$ and so in Step 17 we need at most $a_i - e(p_i)$ applications of the algorithm BASIS. Every application of this algorithm requires

$$O((a_i - e(p_i))(\log p_i^{a_i})^2)$$

group operations and

$$O((a_i - e(p_i))(\log p_i^{a_i})^2 \log\log p_i^{a_i})$$

arithmetic operations.

Hence, the time complexity of the algorithm COMPUTE-FAGB is

$$O(M(\log N)^2/\log\log N + M\sum_{i=1}^{k} A(p_i) + \sum_{i=1}^{k} B(p_i).$$

where

$$A(p_i) = e(p_i)\lceil p_i^{1/2}\rceil^{\mu(p_i)-1}, \quad B(p_i) = (a_i - e(p_i))(\log p_i^{a_i})^2 \log\log p_i^{a_i}.$$

7 Proofs of Corollaries 1 and 2

Proof of Corollary 1. Let $N = p_1^{a_1} \cdots p_k^{a_k}$ be the prime factorization of N. Since G is cyclic, we have $\mu(p_i) = 1$ and $a_i = e(p_i)$ $(i = 1, \ldots, k)$. On the other hand, we have

$$\sum_{i=1}^{k} e(p_i) = O(\log N).$$

Thus, Theorem 1 implies the result.

Proof of Corollary 2. By [20, Theorem 4.1, page 91], $E(\mathbb{F}_q) \cong \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$, where $N_1 | N_2$. If $N_1 = 1$, then Corollary 1 implies the result. Suppose that $N_1 > 1$. The prime factorizations of N_1 and N_2 are

$$N_1 = p_1^{a_1} \cdots p_k^{a_k}, \quad N_2 = p_1^{b_1} \cdots p_l^{b_l},$$

where $k \leq l$ and $a_i \leq b_i$ $(i = 1, \ldots, k)$. Hence

$$N = N_1 N_2 = p_1^{a_1+b_1} \cdots p_k^{a_k+b_k} p_{k+1}^{b_{k+1}} \cdots p_l^{b_l}$$

and so $e(p_i) = a_i + b_i$, $\mu(p_i) = 2$ $(i = 1, \ldots, k)$ and $e(p_i) = b_i$, $\mu(p_i) = 1$ $(i = k+1, \ldots, l)$. Therefore, Theorem 1 implies that the computation of a basis for $E(\mathbb{F}_q)$ requires $O(MN^{1/4} \log N)$ time.

8 Conclusion

In this paper we have presented a deterministic algorithm for the computation of a basis of a finite abelian group G. We have considered the case where a generating system of G and the prime factorization of N is given. Its time complexity is comparable with that of the randomized algorithm of Chen and Fu [5]. In a future work we plan to study more closely families of groups where our algorithm has better time complexity from the existing algorithms. Especially, the computation of a generating system S for the group of points of an elliptic curve E over a finite field $\mathbb{F}_q$ with $|S| < |E(\mathbb{F}_q)|^{1/4}$ will give a faster algorithm for the computation of a basis for the group $E(\mathbb{F}_q)$.

Acknowledgments

The authors are duly grateful to the referees for theirs helpful suggestions and comments. The first author gratefully acknowledges the Department of Informatics and Telecommunications of the University of Athens in which a part of his work was carried out and the support of the project Autonomic Network Arhitecture (ANA), under contract number IST-27489, which is funded by the IST FET Program of the European Commision.

References

1. Beynon, W.M., Iliopoulos, C.S.: Computing a basis for a finite abelian p-group. Information Processing Letters 20, 161–163 (1985)
2. Borges-Quintana, M., Borges-Trenard, M.A., Martínez-Moro, E.: On the use of gröbner bases for computing the structure of finite abelian groups. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E. (eds.) CASC 2005. LNCS, vol. 3718, pp. 52–64. Springer, Heidelberg (2005)
3. Buchmann, J., Schmidt, A.: Computing the structure of a finite abelian group. Mathematics of Computation 74, 2017–2026 (2005)
4. Chen, L.: Algorithms and their complexity analysis for some problems in finite group. Journal of Sandong Normal University 2, 27–33 (1984) (in Chinese)
5. Chen, L., Fu, B.: Linear Sublinear Time Algorithms for the Basis of Abelian groups. Theoretical Computer Science (2010), doi:10.1016/j.tcs.2010.06.011
6. Cheung, K.H., Mosca, M.: Decomposing finite abelian groups. Journal of Quantum Information and Computation 1(3), 26–32 (2001)
7. Dinur I., Grigorescu E., Kopparty S., Sudan M.:, Decodability of Group Homomorphisms beyond the Johnson Bound, ECCC Report, No 20 (2008) and in 40th STOC 275-284 (2008)
8. Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, Cambridge (1999)
9. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 5th edn. Oxford University Press, Oxford (1979)
10. Iliopoulos, C.S.: Analysis of algorithms on problems in general abelian groups. Information Processing Letters 20, 215–220 (1985)
11. Iliopoulos, C.S.: Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. SIAM Journal of Computing 18(4), 658–669 (1989)
12. Kitaev, A.Y.: Quantum computations: algorithms and error correction. Russian Math. Surveys 52(6), 1191–1249 (1997)
13. Koblitz, N., Menezes, A.J.: A survey of public-key cryptosystems. SIAM Review 46(4), 599–634 (2004)
14. Kohel, A.R., Shparlinski, I.E.: Exponential sums and group generators for elliptic curves over finite fields. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 395–404. Springer, Heidelberg (2000)
15. Ledermann, W.: Introduction to group theory. Longman Group Limited, London (1973)
16. Lomont C.: The hidden subgroup problem - review and open problems (2004), http://arxiv.org/abs/quantph/0411037
17. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal of Computing 26(5), 1484–1509 (1997)
18. Teske, E.: The Pohlig-Hellman Method Generalized for Group Structure Computation. Journal of Symbolic Computation 27, 521–534 (1999)
19. Teske, E.: A space efficient algorithm for group structure computation. Mathematics of Computation 67(224), 1637–1663 (1998)
20. Washington, L.C.: Elliptic Curves. Chapman and Hall, Boca Raton (2003)

Rewriting in Varieties of Idempotent Semigroups

Ondřej Klíma, Miroslav Korbelář, and Libor Polák*

Department of Mathematics and Statistics, Masaryk University
Kotlářská 2, 611 37 Brno, Czech Republic
{klima,korbelar,polak}@math.muni.cz
http://www.math.muni.cz

Abstract. We consider rewriting as a tool for solving identity problems in varieties of idempotent semigroups. It is known that there exist finite canonical term rewrite systems and finite canonical word rewrite systems for only a very limited number of those varieties. We present a finite canonical conditional word rewrite system for a particular variety in which the classical approaches cannot be applied. Moreover, we obtain infinite single letter deleting rewrite systems for each join-irreducible variety.

Keywords: Rewriting, identity problems, varieties of semigroups.

1 Introduction

Rewriting is one of the main tools in algorithmic algebra. In semigroup theory one considers the so-called word problems for finitely presented semigroups (i.e. one looks for an effective description of consequences of a given finite set of relations over a finite alphabet). It was quite unexpected that there are finite presentations where the word problem is solvable and where no finite canonical rewrite system exist – see Squier [14].

On the other hand, solving the identity problems in varieties of universal algebras (i.e. to decide effectively which identities are valid there) is other crucial topic in algorithmic algebra. Again, one possibility to solve them is to use rewriting techniques. In contrary to word problems we have to substitute terms (words) into our rewriting rools. We discuss here rewriting for solving the identity problems in varieties of idempotent semigroups.

The lattice $\mathfrak{L}(\mathcal{B})$ of all varieties of idempotent semigroups was described by Birjukov [3], Fennemore [5], Gerhard [6]. – see Figure 1 in Section 2. The authors also showed that each proper variety in $\mathfrak{L}(\mathcal{B})$ could be defined by $x^2 \simeq x$ and single additional identity. In [11,12] the third author presented a transparent way how to solve the identity problems in all members of $\mathfrak{L}(\mathcal{B})$ using certain

* The first and the third author were supported by the Ministry of Education of the Czech Republic under the project MSM 0021622409 and by the Grant no. 201/09/1313 of the Grant Agency of the Czech Republic. The second author was supported by the project LC 505, the Eduard Čech's Center for Algebra and Geometry.

invariants. In this paper, we explore the usage and limitations of rewriting to solve the identity problems in varieties of idempotent semigroups.

Basically three variants of rewrite systems for varieties of semigroups are studied currently: term rewrite systems, word rewrite systems and conditional word rewrite systems. In [1], Baader showed that the first two approaches (using finite systems) are quite restrictive, they apply only in a very limited number of varieties – see Section 3. Concerning the third approach, a remarkable conditional word rewrite system for the variety of all idempotent semigroups was found by Siekmann and Szabó in [13].

In our paper, we first show that a word rewrite system for a certain variety ($\mathcal{C}_1$ in our notation) from [1] can be simplified using a new finite conditional word rewriting system. In an other variety ($\mathcal{B}_2$ in our notation), where a finite word rewrite system does not exist, we can apply successfully a simple finite conditional word rewrite system.

All our rules are single letter deleting. Such rules are appropriate when showing confluency – see Remark 1, and when characterizing canonical forms. Therefore, it was a natural task to find single letter deleting identities for a wide class of varieties in $\mathfrak{L}(\mathcal{B})$; more precisely, we do this for all join-irreducible varieties in $\mathfrak{L}(\mathcal{B})$. Moreover, we show that in those varieties one can reach the canonical forms using single letter deleting rules (whose systems are infinite in general). The main result here is the existence of (infinite) word rewrite systems for those varieties. This can be considered as the first step when looking for finite conditional rewrite systems for such varieties. Other varieties of idempotent semigroups are joins of the join-irreducible ones and $u \simeq v$ is an identity in $\mathcal{U} \vee \mathcal{V}$ if and only if the words u and v have the same canonical forms both in $\mathcal{U}$ and in $\mathcal{V}$. This fact implies that in order to efficiently solve the identity problem in proper varieties of idempotent semigroup, it would suffice to have rewrite systems for join-irreducible varieties.

In our paper we first collect some basic facts from universal algebra and about varieties of idempotent semigroups. In Section 3 we start with a general approach of rewriting and we specify it for varieties of groupoids. Then we consider word rewrite systems for varieties of semigroups and we deal with conditional word rewrite systems (we modified a bit the usual definition – for instance, the system from [13] is finite for us – and we also distinguish between letters and words). Each subsection collects also known results.

In Section 4 we consider finite conditional word rewrite systems for the variety $\mathcal{C}_1$ and for the variety of all idempotent semigroups. A nontrivial example is presented in the next section. Finally, Section 6 deals with single letter deleting identities and single letter deleting rewrite systems.

2 Varieties of Idempotent Semigroups

Let $X = \{x_1, x_2, ...\}$ be a fixed countable set of *variables.* As usual, we denote by X^+ the free semigroup over X (i.e. the set of all words over X with the operation of concatenation). Let λ be the empty word and we denote by $X^* = X^+ \cup \{\lambda\}$

the free monoid over X. Let $p(r_1, r_2, \dots)$ be the word resulting from $p \in X^+$ after simultaneous substitutions r_1 for x_1, r_2 for x_2, ... $(r_1, r_2, \dots \in X^+)$.

An *identity* is a pair of words $(p, q) \in X^+ \times X^+$; we write $p \simeq q$. A semigroup S satisfies the identity $p \simeq q$ if for each homomorphism $\alpha : X^+ \to S$, we have $\alpha(p) = \alpha(q)$. We write $\mathsf{Mod}(\Sigma)$ for the class of all semigroups satisfying all identities from a given system Σ of identities. Such classes are called *varieties.* For a variety $\mathcal{V} = \mathsf{Mod}(\Sigma)$, let $\sim_\mathcal{V}$ be the set of all identities valid in all members of $\mathcal{V}$; in other words, the set of all *consequences* of the system Σ. Let

$$\to_\Sigma = \{\, (sp(r_1, r_2, \dots)t, sq(r_1, r_2, \dots)t) \mid (p, q) \in \Sigma, s, t \in X^*, r_1, r_2, \dots \in X^+ \,\}.$$

A well-known result, the so-called *completeness of equational logics*, by Birkhoff (see Theorem 14.19 in [4]) assures that $\sim_\mathcal{V}$ is the equivalence relation generated by $\to_\Sigma$. Moreover the relations of the form $\sim_\mathcal{V}$ are exactly the fully invariant congruences on X^+ (i.e. congruences invariant with respect to substitutions).

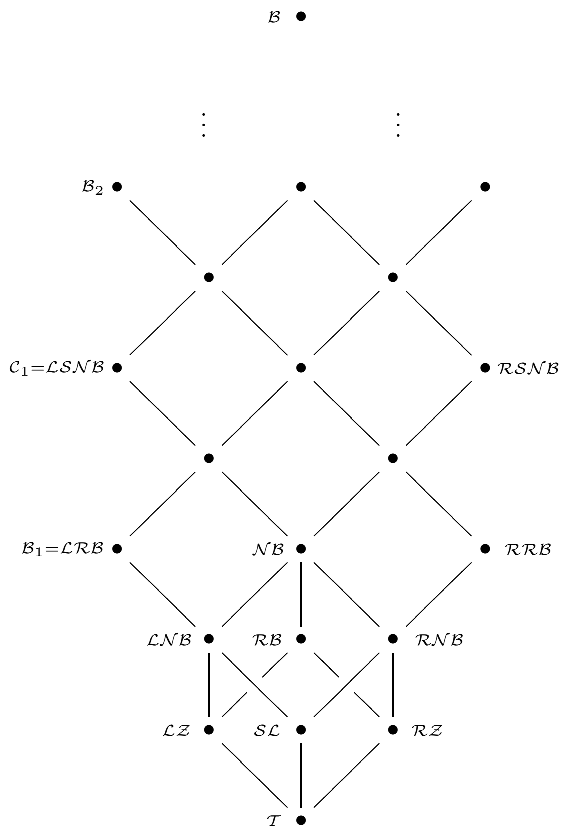


Fig. 1. The lattice of varieties of idempotent semigroups

The lattice of all varieties of idempotent semigroups was independently described by Birjukov [3], Fennemore [5] and Gerhard [6].

Varieties of semigroups are usually presented by systems of identities or by structural properties of their members. In [11,12] we studied the varieties of unions of groups (idempotent semigroups are unions of trivial groups) and the basic tools were alternative descriptions of the relations $\sim_{\mathcal{V}}$'s. We used the following "invariants".

For $p \in X^+$, we define

- the *content* $\mathsf{c}(p) \subseteq X$ of p as the set of all variables in p,
- the *head* $\mathsf{h}(p) \in X$ of p as the leftmost variable in p,
- the *tail* $\mathsf{t}(p) \in X$ of p as the rightmost variable in p,
- $0(p) \in X^*$ as the longest initial segment of p containing all but one variable,
- $1(p) \in X^*$ as the longest final segment of p containing all but one variable,
- $\overrightarrow{p} \in X^+$ as the sequence of the first occurrences of variables when reading p from the left,
- $\overleftarrow{p} \in X^+$ as the sequence of the first occurrences of variables when reading p from the right,
- $|p|$ denotes the length of p.

We also put $\mathsf{h}(\lambda) = \mathsf{t}(\lambda) = 0(\lambda) = \lambda$, $0^0(p) = p$, $0^2(p) = 0(0(p))$ and so on.

For the quite simple case of idempotent semigroups the descriptions of the relations $\sim_{\mathcal{V}}$'s is transparently explained in [7], Section 1.1.3.

3 Rewriting on Varieties of Semigroups

3.1 Generalities

An excellent source on rewriting is the book by Baader and Nipkow [2]. We recall here only facts needed in our text.

Consider a binary relation $\rightarrow$ on a set M, called *rewrite relation*. The problem consists in finding an effective description of the equivalence relation $\mathsf{eq}(\rightarrow)$ generated by $\rightarrow$. We denote by ρ^* the reflexive transitive closure of $\rho \subseteq M \times M$. The relation $\rightarrow$ is

- *terminating* if there is no infinite sequence $a_1 \rightarrow a_2 \rightarrow \dots$, $a_1, a_2, \dots \in M$,
- *locally confluent* if for each $a, b, c \in M$ with $b \leftarrow a \rightarrow c$, there exists $d \in M$ such that
$$b \rightarrow^* d \leftarrow^* c\,,$$
- *confluent* if for each $a, b, c \in M$ with $b \leftarrow^* a \rightarrow^* c$, there exists $d \in M$ such that
$$b \rightarrow^* d \leftarrow^* c\,,$$
- *canonical* if it is terminating and confluent.

In [9], Neuman proves that a terminating locally confluent relation is confluent. A $\rightarrow$-*canonical* form of $a \in M$ is an element $b \in M$ that $a \rightarrow^* b$ and there is no $c \in M$ with $b \rightarrow c$. In general, an element needs not have a canonical form or it can have several of them. In the case of a canonical relation $\rightarrow$, every element a possesses exactly one $\rightarrow$-canonical form which we denote by $[a]_\rightarrow$. In this case, the elements a and b are in the same equivalence class of the relation $\mathsf{eq}(\rightarrow)$ if and only if $[a]_\rightarrow = [b]_\rightarrow$.

In fact, the main task in rewriting consists in the following: let $\sim$ be a given (often not effectively) equivalence relation on a set M and we are looking for a finite canonical rewrite relation $\longrightarrow$ on M generating the relation $\sim$.

3.2 Term Rewriting and Known Results for Idempotent Semigroups

We are interested only in the signature of single binary operational symbol. Let G be the free groupoid over X, i.e. the set of all terms over X in the above signature. For $p, r_1, r_2, \dots \in G$ we define $p(r_1, r_2, \dots)$ as the term resulting from $p = p(x_1, x_2, \dots)$ after simultaneous substitutions r_1 for x_1, r_2 for x_2, ... A *term rewrite system* (TRS in short) is a subset T of $G \times G$. The corresponding rewrite relation on G is

$$\rightarrow_T = \{\, (t, t') \in G \times G \mid \text{where } t, r_1, r_2, \dots \in G,\ (u, v) \in T,\ u(r_1, r_2, \dots) \\ \text{being a subterm of } t \text{ and } t' \text{ results from } t \text{ by putting} \\ v(r_1, r_2, \dots) \text{ in place of } u(r_1, r_2, \dots) \,\} \,.$$

A usage of TRS's for varieties of idempotent semigroups is very restrictive; namely:

Result 1 (Baader [1]). *Let $\mathcal{V}$ be a variety of idempotent semigroups. Then there exists a TRS $T_\mathcal{V}$ such that the rewrite relation $\rightarrow_{T_\mathcal{V}}$ is canonical on G and the equivalence it generates coincides with the fully invariant congruence on G corresponding to the variety $\mathcal{V}$ (i.e. with the set of all groupoid identities which are valid in $\mathcal{V}$) if and only if $\mathcal{V} \in \{\mathcal{LZ}, \mathcal{RZ}, \mathcal{RB}\}$. Moreover, one can take*

- $T_{\mathcal{LZ}} = \{\, (xy)z \rightarrow x(yz),\ xy \rightarrow x \,\}$,
- $T_{\mathcal{RZ}} = \{\, (xy)z \rightarrow x(yz),\ xy \rightarrow y \,\}$,
- $T_{\mathcal{RB}} = \{\, (xy)z \rightarrow xz,\ x(yz) \rightarrow xz,\ xx \rightarrow x \,\}$.

3.3 Word Rewriting and Known Results for Idempotent Semigroups

According to Baader [1], a *word rewrite system* (WRS in short) is a subset W of $X^+ \times X^+$. For a *rule* $(p, q) \in W$ we also write $p \rightarrow q$. A WRS W also defines a rewrite relation $\rightarrow_W$ on X^* by

$$\rightarrow_W = \{\, (sp(r_1, r_2, \dots)t, sq(r_1, r_2, \dots)t) \mid (p, q) \in W, s, t \in X^*, r_1, r_2, \dots \in X^+\}.$$

A usage of WRS's for varieties of idempotent semigroups is applicable also only for a small number of varieties; namely:

Result 2 (Baader [1]). *Let $\mathcal{V}$ be a variety of idempotent semigroups. Then there exists a WRS $W_{\mathcal{V}}$ such that the rewrite relation $\rightarrow_{W_{\mathcal{V}}}$ is canonical and the equivalence it generates coincides with $\sim_{\mathcal{V}}$ if and only if $\mathcal{V}$ equals one of the following varieties:*

(i) $\mathcal{LZ} = \mathsf{Mod}(\, xy \simeq x \,)$,
(ii) $\mathcal{RB} = \mathsf{Mod}(\, x^2 \simeq x, xyx \simeq x \,)$,
(iii) $\mathcal{LRB} = \mathsf{Mod}(\, x^2 \simeq x, xyx \simeq xy \,)$,
(iv) $\mathcal{LQNB} = \mathsf{Mod}(\, x^2 \simeq x, xyxz \simeq xyz \,)$,
(v) $\mathcal{LSNB} = \mathsf{Mod}(\, x^2 \simeq x, xyzxz \simeq xyz \,)$.

or the left-right duals for items (i), (iii)–(v). Moreover, one can take

– $W_{\mathcal{LZ}} = \{\, xy \rightarrow x \,\}$,
– $W_{\mathcal{RB}} = \{\, x^2 \rightarrow x,\ xyz \rightarrow xz \,\}$,
– $W_{\mathcal{LRB}} = \{\, x^2 \rightarrow x,\ xyx \rightarrow xy \,\}$,
– $W_{\mathcal{LQNB}} = \{\, x^2 \rightarrow x,\ xyxz \rightarrow xyz \,\}$,
– $W_{\mathcal{LSNB}} = \{\, x^2 \rightarrow x,\ xyztzx \rightarrow xyztx,\ xzyzx \rightarrow xyzx,\ zxyzx \rightarrow zxyx,$
$zyxzx \rightarrow zyx,\ zyxtzx \rightarrow zyxtx \,\}$.

3.4 Conditional Word Rewriting Systems: Definitions and Examples

For our purposes we formalize the concept of a conditional word rewrite system as follows. Here we use two alphabets; we substitute variables for elements of the first set and words for elements of the second one.

Let $A = \{a_1, a_2, \dots\}$ and $P = \{p_1, p_2, \dots\}$ be the so-called *rule alphabets.* A *conditional rule* is an triple (ℓ, r, φ) where $\ell, r \in (A \cup P)^+$ and φ is a finite relation on $(A \cup P)^+$. A *φ-substitution* is a mapping σ from A to X and from P to X^* (in fact, a pair of mappings), naturally extended to the domain $(A \cup P)^+$, satisfying

$$(u, v) \in \varphi \text{ implies } \mathsf{c}(\sigma(u)) \subseteq \mathsf{c}(\sigma(v))\,.$$

A *conditional word rewrite system* (CWRS in short) C is a set of conditional rules. It defines a rewrite relation $\rightarrow_C$ on X^+ by

$$\rightarrow_C = \{\, (s\sigma(\ell)t,\ s\sigma(r)t) \mid (\ell, r, \varphi) \in C,\ \sigma \text{ is a } \varphi\text{-substitution},\ s, t \in X^* \,\}\,.$$

In what follows we are a bit informal, for instance, we write

$$pxqxr \rightarrow pxqr,\ p, q, r \in X^*,\ x \in X,\ \mathsf{c}(q) \subseteq \mathsf{c}(r) \subseteq \mathsf{c}(pxq),$$

instead of

$$(\, pxqxr,\ pxqr,\ \{(q, r),\ (r, pxq)\}\,),\ x \in A,\ p, q, r \in P\,.$$

Notice that a WRS is a special case of a CWRS; we identify P with X and we do not use A and conditions.

A significant example of a finite CRWS follows.

Result 3 (Siekmann and Szabó [13]). *Let $\mathcal{B} = \mathsf{Mod}(x^2 \simeq x)$ be the variety of all idempotent semigroups. Then the conditional rules*

- $p^2 \to p,\ p \in X^+$,
- $pqr \to pr,\ p, q, r \in X^+,\ \mathsf{c}(q) \subseteq \mathsf{c}(p) = \mathsf{c}(r)$

determine a canonical CWRS on X^+ such that the equivalence it generates is exactly $\sim_{\mathcal{B}}$.

The proof of the local confluency is extremely complicated there. Another type of proof is presented in [8] by Neto and Sezinando; the idea of the proof is to show that each class of $\sim_{\mathcal{B}}$ contains just one word on which conditional rules can not be applied. We return to that result in Section 4.

Also Nordahl in [10] claims that a certain finite set of conditional rules determines a canonical CWRS such that the equivalence it generates is the fully invariant congruence on X^+ corresponding to the join of the variety of all idempotent semigroups and the variety of all commutative semigroups.

4 Two Examples of Finite CWRS with Single Letter Deleting Rules

First we present a simple CWRS for the variety $\mathcal{C}_1 = \mathcal{LSNB} = \mathsf{Mod}(\,x^2 \simeq x, xyzxz \simeq xyz\,)$. According to [11,12], the corresponding fully invariant congruence $\approx_1 = \sim_{\mathcal{C}_1}$ can be described as follows:

$$\text{for all } u, v \in X^*, \text{we have } u \approx_1 v \text{ iff } \overrightarrow{u} = \overrightarrow{v} \text{ and } (\,\forall\, k \geq 0\,)\ \mathsf{t}(0^k(u)) = \mathsf{t}(0^k(v)).$$

Let C be a CWRS consisting of the following rules:
(C1) $x^2 \to x,\ x \in X$,
(C2) $pxy \to py,\ x, y \in X,\ p \in X^+,\ x, y \in \mathsf{c}(p)$.

Note that each $u \in X^+$ can be written in a unique way in the following form:

$$(*) \quad u = y_1 w_1 y_2 w_2 \dots y_n w_n \text{ where } n \geq 1,\ y_1, \dots, y_n \in X,\ w_1, \dots, w_n \in X^*,$$

$$\text{and for all } k \in \{0, \dots, n-1\}, \text{ we have } 0^k(u) = y_1 w_1 y_2 w_2 \dots y_{n-k} w_{n-k}\,.$$

Lemma 1. *The word of the form $(*)$ is a $\to_C$-canonical form if and only if*
(1) $w_1, \dots, w_n \in X \cup \{\lambda\}$, *and*
(2) *for all $j \in \{1, \dots, n\}$, we have $y_j \neq w_j$.*

Proof. First we show that for an arbitrary word $u \in X^+$ written in the form $(*)$ there is such a word v in the form $(*)$ satisfying conditions (1) and (2) such that $u \to_C^* v$. Indeed, if u has form $(*)$ and for some j the length of w_j is more than 1, we apply rule (C2) to shorten this word. Using this repeatedly, we get a word satisfying (1). Then using rule (C1) (repeatedly) we can get the desired form.

By definition of the rules (C1) and (C2), they cannot be applied to reduce words with shape $(*)$ that satisfy properties (1) and (2). □

Lemma 2. *(i) We can derive the defining identities for $\mathcal{C}_1$ from the rules of C.*

(ii) The system C is consistent with $\mathcal{C}_1$, i.e. both rules are identities in $\mathcal{C}_1$.

(iii) If both u and v are $\rightarrow_C$-canonical forms and $u \approx_1 v$, then $u = v$.

Proof. (i): Using rule (C1) one can derive x from xx. Similarly, $xyzxz \rightarrow xyzz$ by (C2) and $xyzz \rightarrow xyz$ by (C1).

(ii): By definition of $\mathcal{C}_1$ we have $(x^2, x) \in \approx_1$ for $x \in X$. Let us consider $x, y \in X$, $p \in X^+$ such that $x, y \in \mathsf{c}(p)$, i.e. we have $pxy \rightarrow py$. We have $\overrightarrow{pxy} = \overrightarrow{p} = \overrightarrow{py}$ and $\mathsf{t}(pxy) = y = \mathsf{t}(py)$. Moreover, $0(pxy) = 0(p) = 0(py)$.

(iii): Let u and v be $\rightarrow_C$-canonical forms. By Lemma 1 we can write $u = y_1 w_1 \dots y_n w_n$ and $v = y'_1 w'_1 \dots y'_m w'_m$, with $n, m \geq 1$, $y_1 \dots, y_n, y'_1, \dots y'_m \in X$ and $w_1, \dots, w_n, w'_1, \dots, w'_n \in X \cup \{\lambda\}$. Since $u \approx_1 v$, we have $y_1 y_2 \dots y_n = \overrightarrow{u} = \overrightarrow{v} = y'_1 y'_2 \dots y'_m$ from which $n = m$ and $y_1 = y'_1, \dots, y_n = y'_n$ follows. Now for each $k \in \{0, \dots, n-1\}$ we consider w_{n-k} and w'_{n-k}. If $w_{n-k} \in X$ then $\mathsf{t}(0^k(u)) = w_{n-k}$ and if $w_{n-k} = \lambda$ then $\mathsf{t}(0^k(u)) = y_{n-k}$. Similarly for $\mathsf{t}(0^k(v))$. Recall that $\mathsf{t}(0^k(u)) = \mathsf{t}(0^k(v))$ follows from the assumption $u \approx_1 v$. If $w_{n-k} = \lambda$ and $w'_{n-k} \in X$ at the same moment, then $y'_{n-k} = y_{n-k} = \mathsf{t}(0^k(u)) = \mathsf{t}(0^k(v)) = w'_{n-k}$ which contradicts condition (ii) in Lemma 1. The case $w_{n-k} \in X$ and $w'_{n-k} = \lambda$ is impossible from the same reason. Thus $w_{n-k} = w'_{n-k} = \lambda$ or $w_{n-k}, w'_{n-k} \in X$. In the second case we have $w_{n-k} = \mathsf{t}(0^k(u)) = \mathsf{t}(0^k(v)) = w'_{n-k}$ and we can conclude with $w_{n-k} = w'_{n-k}$ in all cases. Hence we get $u = v$. □

Theorem 1. *For each $u, v \in X^+$, we have that $u \simeq v$ is an identity in $\mathcal{C}_1$ if and only if the words u and v have the same $\rightarrow_C$-canonical forms.*

Proof. Since the rewriting using the rules C shortens the words, the relation $\rightarrow_C$ is terminating. We show that this relation is also locally confluent. Let $u \in X^+$ that can be rewritten to v and to w in single step using C. Those two words have $\rightarrow_C$-canonical forms, say $\overline{v}$ and $\overline{w}$. By Lemma 2 (ii) we have that $\overline{v} \approx_1 u \approx_1 \overline{w}$ and Lemma 2 (iii) gives that $\overline{v} = \overline{w}$.

To complete the proof we have to show that $\mathsf{eq}(\rightarrow_C) = \approx_1$. The $\subseteq$-part follows from Lemma 2 (ii). Lemma 2 (iii) gives the opposite inclusion. □

The second example of finite CWRS with single letter deleting rules follows.

Remark 1. Having letters instead of words in certain places of a CWRS often leads to the same canonical forms and showing the (local) confluency is much easier. For instance, we can modify the second rule from Result 3 to

$$pxr \rightarrow pr,\ p, r \in X^+,\ x \in X,\ x \in \mathsf{c}(p) = \mathsf{c}(r)\,.$$

On the other hand such modified rules slow down the rewriting.

5 A Finite CWRS for the Variety $\mathcal{B}_2$

We consider the variety $\mathcal{B}_2 = \mathsf{Mod}(\,x^2 \simeq x,\, xyz \simeq xyzxzyz\,)$ using the identities from [5]. According to Proposition 1, proved later in Section 6, the second identity can be replaced by $xyzxyx \simeq xyzyx$. By [11,12], the corresponding fully invariant congruence $\sim_2$ can be effectively described as follows:

$$\text{for all } u, v \in X^*, \text{ we have } u \sim_2 v \text{ if and only if } (\ \forall\ k \geq 0\ )\ \overleftarrow{0^k(u)} = \overleftarrow{0^k(v)}\,.$$

Let D be the CWRS consisting of the following rules:
(D1) $xx \to x,\ x \in X$,
(D2) $pxqx \to pqx,\ p, q \in X^*,\ x \in X,\ \mathsf{c}(qx) \subseteq \mathsf{c}(p)$,
(D3) $pxqxr \to pxqr,\ p, q, r \in X^*,\ x \in X,\ \mathsf{c}(q) \subseteq \mathsf{c}(r) \subseteq \mathsf{c}(pxq)$.

Lemma 3. *Let u be as in $(*)$ with*

$$w_1 = y_{1,1} \dots y_{1,\ell_1},\ \dots,\ w_n = y_{n,1} \dots y_{n,\ell_n},\ \textit{where } y_{i,j} \in X,\ \ell_i \geq 0\,.$$

Then u is a $\to_D$-canonical form if and only if
(1) $y_1 \neq y_{1,1},\ \dots,\ y_n \neq y_{n,1}$,
(2) $|\{y_{1,1}, \dots, y_{1,\ell_1}\}| = \ell_1,\ \dots,\ |\{y_{n,1}, \dots, y_{n,\ell_n}\}| = \ell_n$,
(3) *for $j = 2, \dots, n$, if $0^{n+1-j}(u) = sy_{j,1}t$ with $s, t \in X^*$ and $y_{j,1} \notin \mathsf{c}(t)$, then $\mathsf{c}(ty_j) \nsubseteq \{y_{j,2}, \dots, y_{j,\ell_j}\}$.*

Proof. First we show that for an arbitrary word $u \in X^+$ written in the form $(*)$ there is such a word v in the form $(*)$ satisfying conditions (1) – (3) such that $u \to_C^* v$. We use rule (D1) to guarantee condition (1).

Let u have the form $(*)$ with (1) being satisfied. Let $j \in \{2, \dots, n\}$, $y_{j,\ell} = y_{j,\ell'}$, $\ell \neq \ell'$. We use rule (D2) with the first x being $y_{j,\ell}$ and the second one being $y_{j,\ell'}$. Using this repeatedly, we get a word satisfying (1) and (2).

Let u have the form $(*)$ with (1) and (2) being satisfied. Let $j \in \{2, \dots, n\}$, $0^{n+1-j}(v) = sy_{j,1}t$, $s, t \in X^*$, $y_{j,1} \notin \mathsf{c}(ty_j) \subseteq \{y_{j,2}, \dots, y_{j,\ell_j}\}$. We use rule (D3) where the x's in (D3) are the above mentioned occurrences of $y_{j,1}$ and $p = s$, $q = ty_j$ and $r = y_{j,2} \dots y_{j,\ell_j}$. Using this repeatedly, we get a word satisfying (1) – (3).

Now we show that rules (D1) – (D3) are not applicable to a word u of the form $(*)$ satisfying (1) – (3). (D1) cannot be applied to such a u because (1) and (2) prevent the occurrence of a subword xx in u.

Concerning (D2): due to $\mathsf{c}(qx) \subseteq \mathsf{c}(p)$ the xqx part of $pxqx$ should be placed between some y_j and y_{j+1} in u. But it contradicts (2).

Finally, we show that also rule (D3) is not applicable. Indeed, take a word u of the form $(*)$ satisfying (1) – (3). Let us examine the possible subwords of u with shape $pxqxr$, as in (D3). Notice that $q = \lambda$ is not possible and therefore also $r \neq \lambda$. Due to $\mathsf{c}(r) \subseteq \mathsf{c}(pxq)$ the word r is a segment of some w_j in u. Due to $y_j \notin \mathsf{c}(y_1w_1 \dots y_{j-1}w_{j-1})$, the right x in $pxqxr$ is not y_j from $(*)$. We can suppose that $x \notin \mathsf{c}(q)$, otherwise $q = q_1xq_2$ and we can put $p' = pxq_1$ and use $p'xq_2xr \to p'xq_2r$ instead with the same effect. If $w_j = w'xrw''$ then $w' = \lambda$ due to $\mathsf{c}(w') \subseteq \mathsf{c}(q) \subseteq \mathsf{c}(r)$ and condition (2). Hence $x = y_{j,1}$ and if we consider s and t from (3) we have $q = ty_j$. Thus $\mathsf{c}(q) \nsubseteq \{y_{j,2}, \dots, y_{j,\ell_j}\}$ and consequently $\mathsf{c}(q) \nsubseteq \mathsf{c}(r)$, leading to a contradiction. □

Lemma 4. *(i) We can derive the defining identities for $\mathcal{B}_2$ from the rules of D.*

(ii) The system D is consistent with $\mathcal{B}_2$, i.e. the rules are identities in $\mathcal{B}_2$.

(iii) If both u and v are $\rightarrow_D$-canonical forms and $u \sim_2 v$, then $u = v$.

Proof. (i): Using rule (D1) one can derive x from xx. Using (D2) with $p = xyz$ and $q = y$ one can obtain $xyzxyx \rightarrow_D xyzyx$.

(ii): By construction, we have $xx \sim_2 x$ for $x \in X$.

Consider $pxqx \rightarrow pqx$, $p, q \in X^*$, $x \in X$, $\mathsf{c}(qx) \subseteq \mathsf{c}(p)$. Then $\overleftarrow{pxqx} = \overleftarrow{pqx}$ and $0(pxqx) = 0(pqx)$. Thus $pxqx \sim_2 pqx$.

Consider $pxqxr \rightarrow pxqr$, $p, q, r \in X^*$, $x \in X$, $\mathsf{c}(q) \subseteq \mathsf{c}(r) \subseteq \mathsf{c}(pxq)$. Then $\overleftarrow{pxqxr} = \overleftarrow{pxqr}$ and $0(pxqxr) = 0(pxqr)$. Thus $pxqxr \sim_2 pxqr$.

(iii): Notice that, for a canonical form w with $|\mathsf{c}(w)| \geq 2$, the word $0(w)$ is again a canonical form. Furthermore, for $w, t \in X^*$ with $|\mathsf{c}(w)| \geq 2$, the fact $w \sim_2 t$ gives $0(w) \sim_2 0(t)$. Indeed, $w \sim_2 t$ implies

$$(\ \forall\ k \geq 0\ )\ \overleftarrow{0^k(u)} = \overleftarrow{0^k(v)}\,.$$

Using it for $k = \ell + 1$, $\ell \geq 0$ we obtain

$$(\ \forall\ \ell \geq 0\ )\ \overleftarrow{0^\ell(0(w))} = \overleftarrow{0^\ell(0(t))}$$

which gives $0(w) \sim_2 0(t)$.

Let u be as in Lemma 3 and let v be another word satisfying $u \sim_2 v$. We use induction with respect to $n = |\mathsf{c}(u)|$. For $n = 1$, we have $u = v \in X$. Let $n \geq 2$. Then $0(u) \sim_2 0(v)$ by the remark in the previous paragraph and by the induction assumptions we have $0(v) = 0(u)$. We can write $u = 0(u)y_n y_{n,1} \dots y_{n,\ell_n}$ and $v = 0(u)y_n z_{n,1} \dots z_{n,\ell'_n}$. Now suppose that $\ell_n < \ell'_n$ (the case $\ell_n > \ell'_n$ is similar). Due to (2) and $\overleftarrow{u} = \overleftarrow{v}$ we have $v = 0(u)y_n z_{n,1} \dots z_{n,\ell''_n} y_{n,1} \dots y_{n,\ell_n}$. From (1) we have $y_n \neq z_{n,1}$. Let consider the last occurrence of $z_{n,1}$ in $0(u)$ from the right, i.e. $0(u) = u' z_{n,1} u''$, where $z_{n,1} \notin \mathsf{c}(u'')$. Again due to $\overleftarrow{u} = \overleftarrow{v}$ we have $\mathsf{c}(u'' y_n) \subseteq \mathsf{c}(z_{n,2} \dots z_{n,\ell''_n} y_{n,1} \dots y_{n,\ell_n})$. Now we can use rule (D3) on v for $p = u'$, $x = z_{n,1}$, $q = u'' y_n$, $r = z_{n,2} \dots z_{n,\ell''_n} y_{n,1} \dots y_{n,\ell_n}$, leading to a contradiction.

Thus $\ell_n = \ell'_n$ then $u = v$ by (2) and by $\overleftarrow{u} = \overleftarrow{v}$. □

The proof of the following result is, in fact, the same as that of Theorem 1. The only difference is the usage of Lemma 4 instead of Lemma 2.

Theorem 2. *For each $u, v \in X^+$, we have that $u \simeq v$ is an identity in $\mathcal{B}_2$ if and only if the words u and v have the same $\rightarrow_D$-canonical forms.* □

6 Single Letter Deleting Identities and Single Letter Deleting Rewrite Systems

To solve the identity problem, a natural goal is to have a WRS for each variety of idempotent semigroups. As mentioned in the introduction one can restrict consideration to the join-irreducible varieties from $\mathfrak{L}(\mathcal{B})$ which are on the sides of the lattice $\mathfrak{L}(\mathcal{B})$ on Figure 1. They can be described inductively by using the invariant 0 and left-right duality. These descriptions follow from [12], Theorem 3.6.

For a word $u = x_1x_2 \dots x_m \in X^*$, with $x_1, x_2, \dots, x_m \in X$, the word $u^{\mathsf{r}} = x_m \dots x_2x_1$, is called the *reverse* of u. Furthermore, for a relation ρ on X^+ we consider the *reverse relation* ρ^{r} given by $\rho^{\mathsf{r}} = \{ (u^{\mathsf{r}}, v^{\mathsf{r}}) \mid (u, v) \in \rho \}$.

Now we define ρ^0 in the following way: for all $u, v \in X^*$, we have

$$u \ \rho^0 \ v \quad \text{if} \quad (\ \forall\ k \geq 0\ )\ (0^k(u)\ \rho\ 0^k(v))\,.$$

Denote $\sim_1 = \sim_{\mathcal{LRB}}$, i.e $u \sim_1 v$ if and only if $\overrightarrow{u} = \overrightarrow{v}$. Then $\sim_1^{\mathsf{r}} = \sim_{\mathcal{RRB}}$. For each $n \geq 1$, we inductively define $\sim_{n+1} = (\sim_n^{\mathsf{r}})^0$. In particular, $\sim_2$ coincides with the relation used in Section 5. We denote the corresponding varieties of idempotent semigroups $\mathcal{B}_n$, i.e. $\sim_{\mathcal{B}_n} = \sim_n$. We also denote by $\sim_0 = \{ (u, v) \in X^* \times X^* \mid \mathsf{c}(u) = \mathsf{c}(v) \}$. Then $\sim_0^{\mathsf{r}} = \sim_0$, $\sim_1 = (\sim_0^{\mathsf{r}})^0$ and $\mathcal{B}_0 = \mathcal{SL}$.

If we start from the variety $\mathcal{C}_1 = \mathcal{LSNB}$ we obtain the following similar sequence of varieties We denote $\approx_1 = \sim_{\mathcal{C}_1} = \sim_{\mathcal{LSNB}}$ (see Section 4). Now for each $n \geq 1$ we define inductively $\approx_{n+1} = (\approx_n^{\mathsf{r}})^0$ and we denote the corresponding varieties of idempotent semigroups $\mathcal{C}_n$, i.e. $\approx_{\mathcal{C}_n} = \approx_n$. We also put $u \approx_0 v$ if and only if u and v have the same content and the first letter. Now $\approx_1 = (\approx_0^{\mathsf{r}})^0$, $\approx_0^{\mathsf{r}} \neq \approx_0$ and $\mathcal{C}_0 = \mathcal{LNB}$ is not a join-irreducible variety. The positions in the lattice of varieties of idempotent semigroups is depicted at Figure 2.

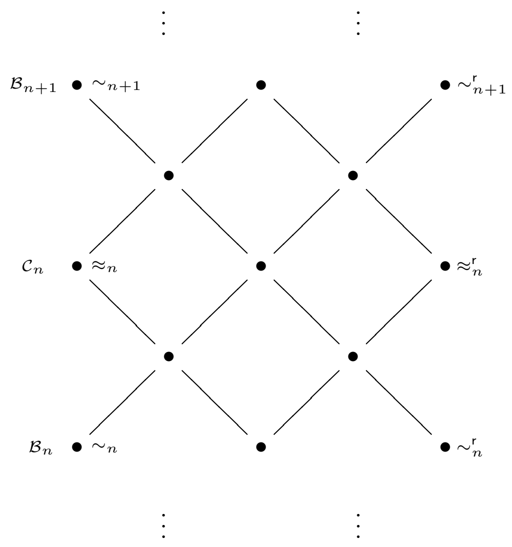


Fig. 2. A part of the lattice of varieties of idempotent semigroups

Remark 2. For each $n \geq 1$ we have $\sim_n \subseteq \approx_{n-1} \subseteq \sim_{n-1}$ and consequently the considered relations are contained in $\sim_1$. Let $\sim$ be a relation $\sim_n$ or $\approx_n$ for some

$n \geq 1$. For $u, v \in X^*$ such that $u \sim v$ we have $\overrightarrow{u} = \overrightarrow{v}$. In particular, if $0(u) = u_0$, which means $u = u_0 x u_1$, $u_0, u_1 \in X^*$, $x \notin \mathsf{c}(u_0)$, $\mathsf{c}(u_1) \subseteq \mathsf{c}(u_0 x)$, then $v = v_0 x v_1$ such that $v_0, v_1 \in X^*$, $x \notin \mathsf{c}(v_0)$, $\mathsf{c}(v_1) \subseteq \mathsf{c}(v_0 x)$ and consequently $0(v) = v_0$. Note also that for each $k \geq 1$ from $u \sim v$ it follows that $0^k(u) \sim 0^k(v)$. These easy observations will be used many times later without a precise reference.

We show that each of the varieties $\mathcal{B}_n$ and $\mathcal{C}_n$ is defined by single identity of the following special form. For our purpose we need identities different from the identities given in [3,5,6]. We denote $u_1 = xy_1x$, $v_1 = xy_1$ and then for each $n \geq 1$ we define inductively $u_{n+1} = xy_1y_2 \dots y_{n+1}u_n^{\mathsf{r}}$ and $v_{n+1} = xy_1y_2 \dots y_{n+1}v_n^{\mathsf{r}}$. The identity $u_n \simeq v_n$ is referred as π_n.

Proposition 1. *For each $n \geq 1$, we have $\mathcal{B}_n = \mathsf{Mod}(\, x^2 \simeq x, \pi_n \,)$.*

Proof. For $n = 1$ the statement follows from $\mathcal{B}_n = \mathcal{LRB}$. Thus assume $n \geq 2$. We denote $w_n = xy_1 \dots y_n$.

It is enough to prove that $u_n \sim_n v_n$ and that the identity π_n is not valid in the variety $\mathcal{V}$ which covers variety $\mathcal{B}_n$ in the lattice of varieties of idempotent semigroups. Indeed, from this we have $\mathcal{B}_n \subseteq \mathsf{Mod}(\, x^2 \simeq x, \pi_n \,) \subsetneq \mathcal{V}$ which gives the result. The variety $\mathcal{V}$ is the join of the variety $\mathcal{B}_n$ and the dual variety for $\mathcal{C}_{n-1}$. This means that we need to show $(u_n, v_n) \in \sim_n$ and $(u_n, v_n) \notin \approx_{n-1}^{\mathsf{r}}$. One can show these statements by induction with respect to n.

We see that for each $k > 0$ we have $0^k(u_n) = 0^k(w_n) = 0^k(v_n)$ and trivially $0^k(u_n) \sim_{n-1}^{\mathsf{r}} 0^k(v_n)$. Hence $(u_n, v_n) \in \sim_n$ if and only if $(u_n, v_n) \in \sim_{n-1}^{\mathsf{r}}$. By the induction assumption we have $(u_{n-1}, v_{n-1}) \in \sim_{n-1}$, hence $(u_{n-1}^{\mathsf{r}}, v_{n-1}^{\mathsf{r}}) \in \sim_{n-1}^{\mathsf{r}}$ and consequently $u_n = w_n u_{n-1}^{\mathsf{r}} \sim_{n-1}^{\mathsf{r}} w_n v_{n-1}^{\mathsf{r}} = v_n$, which finishes the proof of the first statement.

For the relation $\approx_{n-1}^{\mathsf{r}}$ and $n = 2$ we can see that $u_2 = xy_1y_2xy_1x$ and $v_2 = xy_1y_2y_1x$, i.e. $u_2^{\mathsf{r}} = xy_1xy_2y_1$ and $v_2^{\mathsf{r}} = xy_1y_2y_1x$. Since $0(u_2^{\mathsf{r}}) = xy_1x$ and $0(v_2^{\mathsf{r}}) = xy_1$ have not the same last letters, we can conclude $(u_2^{\mathsf{r}}, v_2^{\mathsf{r}}) \notin \approx_1$, thus $(u_2, v_2) \notin \approx_1^{\mathsf{r}}$. We proved the second part for $n = 2$ and we can assume $n \geq 3$.

Now, $(u_n, v_n) \in \approx_{n-1}^{\mathsf{r}}$ if and only if $(u_{n-1}w_n^{\mathsf{r}}, v_{n-1}w_n^{\mathsf{r}}) \in \approx_{n-1}$ which is not true because $0(u_{n-1}w_n^{\mathsf{r}}) = u_{n-1}$, $0(v_{n-1}w_n^{\mathsf{r}}) = v_{n-1}$ and we have the induction assumption that $(u_{n-1}, v_{n-1}) \notin \approx_{n-2}^{\mathsf{r}}$. □

In a similar way we can construct identities for the varieties $\mathcal{C}_n$. We put $s_1 = xy_1y_2xy_1$ and $t_1 = xy_1y_2y_1$. Furthermore, for every $n \geq 1$ we put $s_{n+1} = xy_1 \dots y_{n+2}s_n^{\mathsf{r}}$ and $t_{n+1} = xy_1 \dots y_{n+1}t_n^{\mathsf{r}}$. The identity $s_n \simeq t_n$ is referred as σ_n. Then one can prove the following result in the same way as Proposition 1.

Proposition 2. *For each n we have $\mathcal{C}_n = \mathsf{Mod}(x^2 \simeq x, \sigma_n)$.*

For $n \geq 1$ we consider a rewrite relation $\longrightarrow_n$ on X^* given in the following way: for $u, v \in X^*$ we put $u \longrightarrow_n v$ if v is a subword of u, $|v| = |u| - 1$ and $u \sim_n v$. Similarly, for $u, v \in X^*$ we put $u \Longrightarrow_n v$ if v is a subword of u, $|v| = |u| - 1$ and $u \approx_n v$. Note that the relations $\longrightarrow_n$ and $\Longrightarrow_n$ are not defined for $n = 0$ although some statements concerning $\sim_n$ and $\approx_n$ are also valid in this case.

Lemma 5. *Let $\mathcal{V}$ be one of the varieties $\mathcal{B}_n$, $\mathcal{C}_n$ or their dual, where $n \geq 0$. If $u, v, w \in X^*$ are such that $|v| \geq 2$ and $uvw \sim_{\mathcal{V}} uw$, then there exist words v_0, v_1, v_2 such that $v = v_0v_1v_2$, $1 \leq |v_1| < |v|$ and $uvw \sim_{\mathcal{V}} uv_0v_2w$.*

Proof. We prove the statement by induction with respect to n and the size of the set $\mathsf{c}(uvw)$. We show the detailed proof for varieties $\mathcal{B}_n$ and their duals. If $\mathcal{V} = \mathcal{B}_0$ then we have $\mathsf{c}(uvw) = \mathsf{c}(uw)$. This means that $\mathsf{c}(v) \subseteq \mathsf{c}(uw)$. Let v_0 be the empty word, v_1 be the first letter of v and v_2 be such that $v = v_0v_1v_2 = v_1v_2$. Then $\mathsf{c}(v_2) \subseteq \mathsf{c}(uw)$ and $\mathsf{c}(uv_2w) = \mathsf{c}(uw) = \mathsf{c}(uvw)$ follows.

Now let $n \geq 1$ and let $\mathcal{V} = \mathcal{B}_n$. Let u, v, w be as in the statement, in particular we have $uvw \sim_n uw$. If $\mathsf{c}(uvw) = \mathsf{c}(u)$ then $0(uvw) = 0(u)$. By the induction assumption for $uvw \sim^{\mathsf{r}}_{n-1} uw$ there are v_0, v_1, v_2 such that $v = v_0v_1v_2$, $1 \leq |v_1| < |v|$ and $uvw \sim^{\mathsf{r}}_{n-1} uv_0v_2w$. Since $0(uvw) = 0(u) = 0(uv_0v_2w)$ we get $uvw \sim_n uv_0v_2w$.

Let assume now, that $0(uvw) = us$, where s is a prefix of v such that $|s| < |v|$. This means that $v = sxt$, where $x \in X$, $\mathsf{c}(tw) \subseteq \mathsf{c}(usx)$, $x \notin \mathsf{c}(us)$. Since $\mathsf{c}(uw) = \mathsf{c}(uvw)$ we have $x \in \mathsf{c}(w)$. We consider the first occurrence of x in w, i.e. $w = w_0xw_1$, where $w_0, w_1 \in X^*$ and $x \notin \mathsf{c}(w_0)$. Now from the assumption $uvw \sim_n uw$ we get $us = 0(uvw) \sim_n 0(uw) = uw_0$. We can multiply it by $w = w_0xw_1$ to obtain $usw \sim_n uw_0w_0xw_1 \sim_{\mathcal{B}} uw_0xw_1 = uw$. So, if s is not the empty word λ we are done, we can put $v_0 = s$, $v_1 = xt$ and $v_2 = \lambda$. If $s = \lambda$ then $t \neq \lambda$ and we have $u \sim_n uw_0$. We can multiply it by $xw = xw_0xw_1$ to obtain $uxw \sim_n uw_0xw_0xw_1 \sim_{\mathcal{B}} uw_0xw_1 = uw$. Thus we have the statement for $v_0 = sx = x$, $v_1 = t$ and $v_2 = \lambda$.

Finally, assume that $0(uvw) = uvw_0$ for a certain prefix w_0 of w. This means $w = w_0xw_1$ where $\mathsf{c}(w_1) \subseteq \mathsf{c}(uvw_0x)$ and $x \notin \mathsf{c}(uvw_0)$. Hence we have $0(uw) = uw_0$. Now we use the induction assumption for the smaller set $\mathsf{c}(uvw_0) \subsetneq \mathsf{c}(uvw)$. From $uvw \sim_n uw$ we have $uvw_0 \sim_n uw_0$ and there are v_0, v_1, v_2 such that $v = v_0v_2v_2$ and $uvw_0 \sim_n uv_0v_2w_0$. When we multiply it by xw_1 we obtain $uvw \sim_n uv_0v_2w$. □

Lemma 6. *Let $n \geq 1$ and $u, v, w \in X^*$.*

i) If $uvw \sim_n uw$, then $uvw \longrightarrow^*_n uw$.
ii) If $uvw \approx_n uw$, then $uvw \Longrightarrow^*_n uw$.

Proof. It follows from Lemma 5 by induction with respect to the length of v. □

Lemma 7. *Let $\sim$ be one of the relations $\sim_n$, $\sim^{\mathsf{r}}_n$, $\approx_n$ and $\approx^{\mathsf{r}}_n$ with $n \geq 1$. Let $u, v, w, s \in X^*$ and $x, y \in X$, $x \neq y$.*

i) If $s = uxvyw \sim uxvw \sim uvyw$, then $uvw \sim s$ or $uw \sim s$.
ii) If $s = uxvxw \sim uxvw \sim uvxw$, then $uvw \sim s$ or $uxw \sim s$.

Proof. i) We prove the statement by induction with respect to n and with respect to the number of letters used in s. We prove it even for $n = 0$. So, let $\sim = \sim_0$, which means that $t_1 \sim t_2$ if and only if $\mathsf{c}(t_1) = \mathsf{c}(t_2)$. From the assumption $uxvyw \sim uxvw \sim uvyw$ we get that $x \in \mathsf{c}(uvyw)$ and $y \in \mathsf{c}(uxvw)$. Since $x \neq y$, we have $x, y \in \mathsf{c}(uvw)$ and we get $\mathsf{c}(uvw) = \mathsf{c}(s)$, i.e. $uvw \sim s$.

Assume that the statement holds for $\sim_n$, which gives the statement for $\sim_n^{\mathsf{r}}$ immediately. Let $\sim\, = \,\sim_{n+1}$ and $s = uxvyw \sim uxvw \sim uvyw$. Similarly to the proof of Lemma 5 we distinguish the cases depending on $0(uxvyw)$.

If $0(uxvyw) = u_0$, where u_0 is a proper prefix of u, then $u = u_0 z u_1$, $z \in X$, $u_0, u_1 \in X^*$, $z \notin \mathsf{c}(u_0)$ and $\mathsf{c}(u_1 xvyw) \subseteq \mathsf{c}(u_0 z)$. Hence $0(uxvw) = 0(uvyw) = 0(uvw) = 0(uw) = u_0$. Since these words are identical we have $uvw \sim_{n+1} s$ if and only if $uvw \sim_n^{\mathsf{r}} s$ and we have the same for uw. From the assumption $s = uxvyw \sim_{n+1} uxvw \sim_{n+1} uvyw$ we have $s = uxvyw \sim_n^{\mathsf{r}} uxvw \sim_n^{\mathsf{r}} uvyw$ and by the induction assumption we obtain $uvw \sim_n^{\mathsf{r}} s$ or $uw \sim_n^{\mathsf{r}} s$. Thus we get $uvw \sim s$ or $uw \sim s$.

If $0(uxvyw) = u$ then $x \notin \mathsf{c}(u)$ and $\mathsf{c}(vyw) \subseteq \mathsf{c}(ux)$. We distinguish two cases $x \in \mathsf{c}(v)$ and $x \notin \mathsf{c}(v)$. In the first case let $v_0, v_1 \in X^*$ be words such that $v = v_0 x v_1$, $x \notin \mathsf{c}(v_0)$. Then $0(uvyw) = uv_0$ and we have $u \sim uv_0$. We multiply it by $xvw = xv_0xv_1w$ and we obtain $uxvw \sim uv_0xv_0xv_1w \sim_{\mathcal{B}} uv_0xv_1w = uvw$ which means that $uvw \sim s$. In the second case $x \notin \mathsf{c}(v)$, since $x \in \mathsf{c}(uvyw)$ we have $x \in \mathsf{c}(w)$, i.e. $w = w_0 x w_1$ for some words $w_0, w_1 \in X^*$, $x \notin \mathsf{c}(w_0)$. Then $u = 0(uxvyw) \sim_{n+1} 0(uvyw) = uvyw_0$. If we multiply it by $w = w_0xw_1$ we get $uw \sim_{n+1} uvyw_0w_0xw_1 \sim_{\mathcal{B}} uvyw_0xw_1 = uvyw \sim s$.

If $0(uxvyw) = uxv_0$, where $v_0 \in X^*$ is a prefix of v, $|v_0| < |v|$ then $v = v_0 z v_1$, $z \in X$, $v_1 \in X^*$, $z \notin \mathsf{c}(uv_0)$. Then $0(uvyw) = uv_0$ and we get $uxv_0 = 0(uxvyw) \sim 0(uvyw) = uv_0$. If we multiply it by zv_1w we obtain $s \sim uxvw \sim uvw$.

If $0(uxvyw) = uxv$ then $y \notin \mathsf{c}(uxv)$, but $y \in \mathsf{c}(uxvyw) = \mathsf{c}(uxvw)$. Let w_0, w_1 be words such that $w = w_0yw_1$ and $y \notin \mathsf{c}(w_0)$. Hence we have $uxv = 0(uxvyw) \sim 0(uvyw) = uv$. We multiply it by $w = w_0yw_1$ and we obtain $uxvw \sim uvw$.

Finally, if $0(uxvyw) = uxvyw_0$ where $w_0 \in X^*$ is a prefix of w, $|w_0| < |w|$ then $w = w_0zw_1$, $z \in X$, $w_1 \in X^*$, $z \notin \mathsf{c}(uxvyw_0)$. Then from the assumption $s = uxvyw \sim uxvw \sim uvyw$ we obtain $s' = uxvyw_0 \sim uxvw_0 \sim uvyw_0$. If we use the induction assumption for the set $\mathsf{c}(s')$ then we get $uvw_0 \sim s'$ or $uw_0 \sim s'$. If we multiply it by zw_1 we obtain the statement.

For relations $\approx_n$ the only difference is that at the beginning for $\approx_0$ we need to check in addition that all considered words have the same first letter.

ii) For $\sim_n$ the statement can be proved in the same manner. The only difference is that it does not hold for $n = 0$. Indeed, if $u = w = \lambda$, and $v = z \neq x$, $z \in X$, then we have the assumption $xzx \sim_0 xz \sim_0 zx$ because they have the content $\{x, z\}$. But $v = z$ and x have different content. This means that we need to prove the statement for $\sim_1$ first. Assume that $\overrightarrow{uxvxw} = \overrightarrow{uvxw}$. If $x \in \mathsf{c}(u)$ then we see that $\overrightarrow{uvxw} = \overrightarrow{uvw}$. If $x \notin \mathsf{c}(uv)$ then from $\overrightarrow{uxvxw} = \overrightarrow{uvxw}$ we see that $\mathsf{c}(v) \subseteq \mathsf{c}(u)$ and consequently $\overrightarrow{uvxw} = \overrightarrow{uxw}$. Finally, if $x \notin \mathsf{c}(u)$ but $x \in \mathsf{c}(v)$, then $\overrightarrow{uvxw} = \overrightarrow{uvw}$. We prove the statement for $n = 1$. The induction step can we done in the same way as in the proof of item i). For the relations $\approx_n$ one can prove the statement in similar way. □

Theorem 3. *Let n be an arbitrary natural number. Then the rewrite relations $\longrightarrow_n$ and $\Longrightarrow_n$ are canonical. Moreover, for all $u, v \in X^*$, we have $u \sim_n v$ if and only if $[u]_{\longrightarrow_n} = [v]_{\longrightarrow_n}$ and $u \approx_n v$ if and only if $[u]_{\Longrightarrow_n} = [v]_{\Longrightarrow_n}$.*

Proof. The relation $\longrightarrow_n$ is strictly size-decreasing, hence it is terminating. We show that $\longrightarrow_n$ is locally confluent. Assume that $s = uxvyw \longrightarrow_n uxvw$ and $uxvyw \longrightarrow_n uvyw$. We need to find t such that $uxvw \longrightarrow_n^* t$ and $uvyw \longrightarrow_n^* t$. First, assume in addition that $x \neq y$. By Lemma 7 (i) we know that $uvw \sim_n s$ or $uw \sim_n s$. In the first case we have $uxvw \longrightarrow_n uvw$ and also $uvyw \longrightarrow_n uvw$. In the second case we have $uxvw \sim_n uw$ and $uvyw \sim_n uw$. Now we use Lemma 6 to state $uxvw \longrightarrow_n^* uw$ and also $uvyw \longrightarrow_n^* uw$. Now we assume that $x = y$. By Lemma 7 part ii) we have that $uvw \sim s$ or $uxw \sim s$ and one can finish the proof in the same way as in the case $x \neq y$. We proved that $\longrightarrow_n$ is terminating and locally confluent and, consequently, it is confluent.

To prove the second part of the statement we first assume that $[u]_{\longrightarrow_n} = w = [v]_{\longrightarrow_n}$. Then $u \longrightarrow_n^* w$ and $v \longrightarrow_n^* w$ from which it follows that $u \sim_n w \sim_n v$.

Now we assume that $u \sim_n v$. This means that $u \simeq v$ is an identity for $\mathcal{B}_n$. By Proposition 1, we know that $\mathcal{B}_n = \mathsf{Mod}(x^2 \simeq x, \pi_n)$. From the completeness of equational logic, there is a sequence of words $u = u_1, u_2, \dots, u_n = v$ such that each pair (u_i, u_{i+1}) is of the form $(sp(r_1, r_2, \dots)t, sq(r_1, r_2, \dots)t)$ where $p(x_1, x_2, \dots) \simeq q(x_1, x_2, \dots)$. More precisely

$$(p, q) \in I = \{(x, x^2), (x^2, x), (u_n, v_n), (v_n, u_n)\}$$

where u_n and v_n form the identity π_n from Proposition 1. Each identity from I is of a very special form, namely it is a letter deleting one. After applying a considered substitution, we get that $p(r_1, r_2, \dots)$ arises from $q(r_1, r_2, \dots)$ by removing a certain factor or vice-versa $q(r_1, r_2, \dots)$ arises from $p(r_1, r_2, \dots)$ in the same way. Consequently, the same holds for each pair (u_i, u_{i+1}). Since $u_i \sim_n u_{i+1}$ we can apply Lemma 6 to get that for each i we have $u_i \longrightarrow_n^* u_{i+1}$ or $u_{i+1} \longrightarrow_n^* u_i$. In both cases we can deduce $[u_i]_{\longrightarrow_n} = [u_{i+1}]_{\longrightarrow_n}$ because $\longrightarrow_n$ is a canonical rewrite relation. Hence we get $[u]_{\longrightarrow_n} = [u_1]_{\longrightarrow_n} = [u_2]_{\longrightarrow_n} = \cdots = [u_n]_{\longrightarrow_n} = [v]_{\longrightarrow_n}$.

The proof for $\Longrightarrow_n$ is analogical. □

References

1. Baader, F.: Rewrite Systems for Varieties of Semigroups. In: Stickel, M.E. (ed.) CADE 1990. LNCS, vol. 449, pp. 396–410. Springer, Heidelberg (1990)
2. Baader, F., Nipkow, T.: Term Rewriting and All That. Camb. Univ. Press, Cambridge (1999)
3. Birjukov, A.P.: Varieties of Idempotent Semigroups. Algebra i Logika 9, 255–273 (1970); English translation in Algebra and Logic 9, 153–164 (1971)
4. Burris, S., Sankappanavar, H.P.: A Course in Universal Algebra. Springer, New York (1981)
5. Fennemore, C.F.: All Varieties of Bands I, II. Math. Nachr. 48, 237–252, 253-262 (1971)
6. Gerhard, J.A.: The Lattice of Equational Classes of Idempotent Semigroups. J. Algebra 15, 195–224 (1970)
7. Klíma, O.: Ph.D. thesis: Unification Modulo Associativity and Idempotency (2003), http://www.math.muni.cz/~klima/Math/Thesis/thesis.html

8. Neto, O., Sezinando, H.: Band Monoid Languages Revisited. Semigroup Forum 61(1), 32–45 (2000)
9. Newman, M.H.A.: On Theories with a Combinatorial Definition of 'Equivalence'. Annals of Mathematics 43(2), 223–243 (1942)
10. Nordahl, T.E.: On the Join of the Variety of All Bands and the Variety of All Commutative Semigroups via Conditional Rewrite Rules. In: Ito, M. (ed.) Words, Languages & Combinatorics, pp. 365–372. World Scientific, Singapore (1992)
11. Polák, L.: On Varieties of Completely Regular Semigroups I. Semigroup Forum 32, 97–123 (1985)
12. Polák, L.: On Varieties of Completely Regular Semigroups II. Semigroup Forum 36(3), 253–284 (1987)
13. Siekmann, J.H., Szabó, P.: A Noetherian and Confluent Rewrite System for Idempotent Semigroups. Semigroup Forum 25(1), 83–110 (1982)
14. Squier, C.C.: Word Problems and a Homological Finiteness Condition for Monoids. Journal of Pure and Applied Algebra 49, 201–217 (1987)

Simplifying Algebraic Functional Systems

Cynthia Kop

Department of Computer Science
VU University Amsterdam, The Netherlands

Abstract. A popular formalism of higher order rewriting, especially in the light of termination research, are the *Algebraic Functional Systems* (AFSs) defined by Jouannaud and Okada. However, the formalism is very permissive, which makes it hard to obtain results; consequently, techniques are often restricted to a subclass. In this paper we study termination-preserving transformations to make AFS-programs adhere to a number of standard properties. This makes it significantly easier to obtain general termination results.

Keywords: higher order term rewriting, algebraic functional systems, termination, transformations, currying, η-expansion.

1 Introduction

The last years have seen a rise in the interest in higher order rewriting, especially the field of termination. While this area is still far behind its first order counterpart, various techniques for proving termination have been developed, such as monotone algebras [11], path orderings [4,1] and dependency pairs [12,9,8]. Since 2010 the annual termination competition [13] has a higher order category.

However, a persistent problem is the lack of a fixed standard. There are several formalisms, each dealing with the higher order aspect in a different way, as well as variations and restrictions. Each style has different applications it models better, so it is hard to choose one over the others. Because of the differences, results in one formalism do not, in general, carry over to the next.

Consequently, when the topic of a higher order termination competition was brought up last year, the first question was: "What formalism?" Even having settled on monomorphic Algebraic Functional Systems, neither of the participating groups could immediately deal with the other's benchmarks, since one group used functional syntax and the other applicative.

In this paper we seek to alleviate this situation by studying transformations of Algebraic Functional Systems. AFSs, which were introduced as a modelling of functional programming languages, are an often recurring format in higher-order termination research, although most of the time they appear with some restrictions. Here we study an unrestricted version, with polymorphic types. After transforming an AFS it will satisfy a number of pleasant properties, such as β-normality and possibly η-normality, and we can freely swap between applicative and functional notation. The transformations are designed to have as little impact as possible and to preserve both termination and non-termination.

The aim of this work is to simplify termination research and tools for AFS. Without changing the syntactical freedom of the formalism, most unwelcome intricacies of the syntax can be eliminated in the input module of a tool. Techniques which are defined on a restricted subclass of AFSs (for example when rules are assumed to be η-normal) become generally applicable.

We will first present the definition of (polymorphic) AFSs. Section 3 sketches the problems we aim to solve; Sections 4–8 discuss various transformations. In Section 9 we will say a few words about generalising existing results.

2 Preliminaries

Algebraic Functional Systems (AFSs) were first defined in [3], but we follow (roughly) the more common definitions of [4]. Rather than using type declarations for variables in an environment, we annotate variables with their types directly in terms. This avoids the need to keep track of an environment.

Types. Given a set of *type constructors* $\mathcal{B}$, each with a fixed arity $ar(b)$, and a set of *type variables* $\mathcal{A}$, the set of *polymorphic types* is defined by the grammar:

$$\mathcal{T} = \alpha \mid b(\mathcal{T}^n) \mid \mathcal{T} \to \mathcal{T} \qquad (\alpha \in \mathcal{A},\ b \in \mathcal{B},\ ar(b) = n)$$

A *monomorphic* type does not contain type variables. We assume at least one type constructor has arity 0, so monomorphic types exist. A type $\sigma \to \tau$ is *functional*, and a type $b(\sigma_1, \dots, \sigma_n)$ is a *data type*. Types are written as σ, τ, ρ, data types as ι, κ and type variables as $\alpha, \varepsilon, \omega$. The $\to$ operator associates to the right. A *type declaration* is an expression of the form $(\sigma_1 \times \dots \times \sigma_n) \longrightarrow \tau$ with $\sigma_1, \dots, \sigma_n, \tau \in \mathcal{T}$. A type declaration $() \longrightarrow \tau$ is usually just denoted τ. For any type σ, let $FTVar(\sigma)$ be the set of type variables occurring in σ.

Example 1. Examples of monomorphic types are $\mathtt{nat}$, $\mathtt{nat} \to \mathtt{bool}$, and $\mathtt{list}(\mathtt{nat})$, and an example of a non-monomorphic type is $\alpha \to \mathtt{list}(\alpha)$.

A *type substitution* θ is a finite mapping $[\alpha_1 := \sigma_1, \dots, \alpha_n := \sigma_n]$; $\mathrm{dom}(\theta) = \{\alpha_1, \dots, \alpha_n\}$, and $\tau\theta$ is the result of replacing all α_i in τ by σ_i (with τ a type or type declaration). We say $\sigma \geq \tau$ if $\tau = \sigma\theta$ for some type substitution θ. For example, $\alpha \geq \alpha \geq \alpha \to \varepsilon \geq \mathtt{nat} \to \mathtt{nat}$, but not $\alpha \to \alpha \geq \mathtt{nat} \to \mathtt{bool}$. Substitutions θ, χ *unify* types σ, τ if $\sigma\theta = \tau\chi$[1]. We will use the following lemma:

Lemma 1 (most general unifiers). *If σ, τ are unifiable, there exist type substitutions θ, χ which unify σ, τ such that for any unifying substitutions θ', χ', there is a type substitution d such that $\theta'_{\upharpoonright FTVar(\sigma)} = d \circ \theta$ and $\chi'_{\upharpoonright FTVar(\tau)} = d \circ \chi$.*

The type substitutions θ, χ in this Lemma are called *most general unifiers*. The composition $d \circ \theta$ is defined as $[\alpha := d(\theta(\alpha)) | \alpha \in \mathrm{dom}(\theta)]$, and $\theta'_{\upharpoonright FTVar(\sigma)}$ is defined as $[\alpha := \theta'(\alpha) | \alpha \in \mathrm{dom}(\theta') \cap FTVar(\sigma)]$.

[1] Note that this definition of unifiers considers the type variables in σ and τ as different entities: the types α and $b(\alpha)$ are unified by $\theta = [\alpha := b(\varepsilon)]$ and $\chi = [\alpha := \varepsilon]$.

Terms. Given a set $\mathcal{F}$ of *function symbols*, each equipped with a type declaration (notation f_σ), and an infinite set $\mathcal{V}$ of *variables*, the set of *terms* consists of those expressions for which we can derive $s : \sigma$ for some type σ, using the clauses:

$$\begin{array}{lll} (\texttt{var}) & x_\sigma : \sigma & \text{if } x \in \mathcal{V} \text{ and } \sigma \text{ a type} \\ (\texttt{fun}) & f_\sigma(s_1, \dots, s_n) : \tau & \text{if } f_\rho \in \mathcal{F} \text{ and } \rho \geq \sigma = (\sigma_1 \times \dots \times \sigma_n) \longrightarrow \tau \\ & & \quad \text{and } s_1 : \sigma_1, \dots, s_n : \sigma_n \\ (\texttt{abs}) & \lambda x_\sigma . s : \sigma \to \tau & \text{if } x \in \mathcal{V},\ \sigma \text{ a type and } s : \tau \\ (\texttt{app}) & s \cdot t : \tau & \text{if } s : \sigma \to \tau \text{ and } t : \sigma \end{array}$$

Moreover, variables must have a unique type in s: if for $x \in \mathcal{V}$ both x_σ and x_τ occur in s then $\sigma = \tau$. The abstraction operator λ binds occurrences of a variable as in the λ-calculus; term equality is modulo renaming of variables bound by an abstraction operator (α-conversion). Write $FVar(s)$ for the set of variables in s not bound by a λ. The $\cdot$ operator for application associates to the left. To maintain readability, we will regularly omit explicit type notation in function symbols and variables, and just assume the most general possible type.

Example 2. As a running example we will use the system $\mathcal{F}_{\texttt{map}}$ with symbols:

$$\begin{Bmatrix} \texttt{map}_{((\alpha\to\alpha)\times\texttt{list}(\alpha))\longrightarrow\texttt{list}(\alpha)}, & \texttt{op}_{(\omega\to\varepsilon\times\alpha\to\omega)\longrightarrow\alpha\to\varepsilon}, & \texttt{nil}_{\texttt{list}(\alpha)}, & \texttt{0}_{\texttt{nat}}, \\ \texttt{cons}_{(\alpha\times\texttt{list}(\alpha))\longrightarrow\texttt{list}(\alpha)}, & \texttt{pow}_{(\alpha\to\alpha\times\texttt{nat})\longrightarrow\alpha\to\alpha}, & \texttt{s}_{(\texttt{nat})\longrightarrow\texttt{nat}} & \end{Bmatrix}$$

An example term in this system is $\texttt{map}(\lambda x.\texttt{s}(x), \texttt{cons}(\texttt{0}, \texttt{nil}))$. Since type annotations have been omitted, they should be imagined as general as possible to keep the term well-typed, so cons, for instance, would be $\texttt{cons}_{(\texttt{nat}\times\texttt{list}(\texttt{nat}))\longrightarrow\texttt{list}(\texttt{nat})}$.

We extend type substitutions and $\geq$ to terms in the obvious way, with a type substitution θ replacing α by $\theta(\alpha)$ in all type denotations in the term.

Example 3. Using the symbols of Example 2, $\texttt{op}_{(\alpha\to\varepsilon\times\varepsilon\to\alpha)\longrightarrow\varepsilon\to\varepsilon}(F_{\alpha\to\varepsilon}, G_{\varepsilon\to\alpha})$ $[\alpha := \varepsilon, \varepsilon := \texttt{nat}] = \texttt{op}_{(\varepsilon\to\texttt{nat}\times\texttt{nat}\to\varepsilon)\longrightarrow\texttt{nat}\to\texttt{nat}}(F_{\varepsilon\to\texttt{nat}}, G_{\texttt{nat}\to\varepsilon})$.

A (term) substitution is the homomorphic extension of a mapping $[x^1_{\sigma_1} := s_1, \dots, x^n_{\sigma_n} := s_n]$, where each $s_i : \sigma_i$. Substitutions are denoted $\gamma, \delta, \dots$, the result of applying a substitution $s\gamma$. A substitution cannot affect bound variables; applying a substitution $(\lambda x_\sigma.s)\gamma$ assumes x occurs neither in domain nor range of γ (a safe assumption since we can rename bound variables). A *context* is a term C containing a special symbol $\Box_\sigma$. The result of replacing $\Box_\sigma$ in C by a term s of type σ is denoted $C[s]$. Here, s may contain variables bound by C.

β and η. $\Rightarrow_\beta$ is the monotonic relation generated by $(\lambda x.s) \cdot t \Rightarrow_\beta s[x := t]$. This relation is strongly normalising and has unique normal forms.

For a given set V of variables, we define restricted η-expansion: $C[s] \hookrightarrow_{\eta,V} C[\lambda x_\sigma.s \cdot x_\sigma]$ if $s : \sigma \to \tau$, x is fresh and the following conditions are satisfied:

1. s is neither an abstraction nor a variable in V
2. s in $C[s]$ is not the left part of an application.

By (2), s is not expanded if it occurs in a subterm of the form $s\ t_1 \cdots t_n$; (1) and (2) together guarantee that $\hookrightarrow_{\eta,V}$ terminates. Therefore every term s has a unique *η, V-long form* $s\!\uparrow^{\eta}_{V}$ which can be found by applying $\hookrightarrow_{\eta,V}$ until it is no longer possible. We say a term s is in η-long form if $s = s\!\uparrow^{\eta} = s\!\uparrow^{\eta}_{\emptyset}$.

Rules and Rewriting. An AFS consists of an alphabet $\mathcal{F}$ and a (finite or countably infinite) set $\mathcal{R}$ of *rules*. Rules are tuples $l \Rightarrow r$ where l and r are terms of the same type such that all variables and type variables in r also occur in l. The relation $\Rightarrow_{\mathcal{R}}$ induced by $\mathcal{R}$ is the monotonic relation generated by the β-rule and: $l\theta\gamma \Rightarrow_{\mathcal{R}} r\theta\gamma$ if $l \Rightarrow r \in \mathcal{R}$, θ is a type substitution and γ a substitution.

Example 4. Using the symbols $\mathcal{F}_{\texttt{map}}$ from Example 2, let $R_{\texttt{map}}$ be the set:

$$\left\{\begin{array}{ll}
\texttt{map}(F,\texttt{nil}) & \Rightarrow \texttt{nil} \\
\texttt{map}(F,\texttt{cons}(x,y)) & \Rightarrow \texttt{cons}(F\cdot x,\texttt{map}(F,y)) \\
\texttt{pow}(F,0) & \Rightarrow \lambda x.x \\
\texttt{pow}(F,\texttt{s}(x)) & \Rightarrow \texttt{op}(F,\texttt{exp}(F,x)) \\
\texttt{op}(F,G)\cdot x & \Rightarrow F\cdot(G\cdot x) \\
\lambda x.F\cdot x & \Rightarrow F
\end{array}\right\}$$

As before, types should be imagined as general as possible. The last rule, for example, should be read as $\lambda x_\alpha.F_{\alpha\to\epsilon}\cdot x_\alpha \Rightarrow F_{\alpha\to\epsilon}$. An example reduction:

$$\begin{array}{ll}
\texttt{map}(\underline{\texttt{pow}(\lambda x.\texttt{s}(\texttt{s}(x)),\texttt{s}(0))},\texttt{cons}(0,\texttt{nil})) & \Rightarrow_{\mathcal{R}} \\
\underline{\texttt{map}(\texttt{op}(\lambda x.\texttt{s}(\texttt{s}(x)),\texttt{pow}(\lambda x.\texttt{s}(\texttt{s}(x)),0)),\texttt{cons}(0,\texttt{nil}))} & \Rightarrow_{\mathcal{R}} \\
\texttt{cons}(\texttt{op}(\lambda x.\texttt{s}(\texttt{s}(x)),\texttt{pow}(\lambda x.\texttt{s}(\texttt{s}(x)),0))\cdot 0,\underline{\texttt{map}(\ldots,\texttt{nil})}) & \Rightarrow_{\mathcal{R}} \\
\texttt{cons}(\texttt{op}(\lambda x.\texttt{s}(\texttt{s}(x)),\underline{\texttt{pow}(\lambda x.\texttt{s}(\texttt{s}(x)),0)})\cdot 0,\texttt{nil}) & \Rightarrow_{\mathcal{R}} \\
\texttt{cons}(\underline{\texttt{op}(\lambda x.\texttt{s}(\texttt{s}(x)),\lambda y.y)\cdot 0},\texttt{nil}) & \Rightarrow_{\mathcal{R}} \\
\texttt{cons}((\lambda x.\texttt{s}(\texttt{s}(x)))\cdot(\underline{(\lambda y.y)\cdot 0}),\texttt{nil}) & \Rightarrow_{\beta} \\
\texttt{cons}(\underline{(\lambda x.\texttt{s}(\texttt{s}(x)))\cdot 0},\texttt{nil}) & \Rightarrow_{\beta} \\
\texttt{cons}(\texttt{s}(\texttt{s}(0)),\texttt{nil}) &
\end{array}$$

Note that the $\Rightarrow_\beta$ steps in this are also $\Rightarrow_{\mathcal{R}}$ steps since $\Rightarrow_\beta$ is included in $\Rightarrow_{\mathcal{R}}$ by definition; they are named separately for clarity.

3 Problems

The permissive nature of AFS-syntax makes it difficult to obtain general results. The first issue is the status of application. When extending first order results it is convenient to consider the $\cdot$ operator as a (polymorphic) binary function symbol. But this doesn't work very well with applicative systems, which have rules like $\texttt{map}\cdot F\cdot(\texttt{cons}\cdot x\cdot y) \Rightarrow \texttt{cons}\cdot(F\cdot x)\cdot(\texttt{map}\cdot F\cdot y)$; AFSs generated from functional programs in e.g. Haskell will commonly have such a form. Due to the repeated occurrence of the $\cdot$ symbol, no version of a higher-order path ordering [4,1] can handle this system. To avoid this problem, we might consider

$\cdot$ as a stronger construction, much like function application; this is done in Nipkow's *Higher-Order Rewriting Systems* (HRSs) [10], where terms are built using only abstraction and application. Ideally, a term $\mathtt{map} \cdot x \cdot (\mathtt{cons} \cdot y \cdot z)$ could be translated to its functional counterpart $\mathtt{map}(x, \mathtt{cons}(y, z))$. But initially this is impossible: a system with the rule $x \cdot 0 \Rightarrow f \cdot 0$ admits a self-loop $f \cdot 0 \Rightarrow f \cdot 0$, whereas the corresponding functional rule, $x \cdot 0 \Rightarrow f(0)$, is terminating.

Another difficulty is the form of the left-hand side of a rule. Methods like the dependency pair approach crucially rely on rules having a form $f(\ldots) \Rightarrow r$ or, in the recent dependency pair analysis for AFSs in [8], $f(l_1, \ldots, l_n) \cdot l_{n+1} \cdots l_m \Rightarrow r$. Consequently, systems with rules like $\lambda x.F \cdot x \Rightarrow F$ cannot be handled.

Termination techniques are often defined only on a restricted subset of AFSs. Since most common examples are expressed in a well-behaved manner, this does not seem too high a price. However, a transformational approach, where for instance a term $f(s, t)$ is replaced by $t \cdot s$, is likely to create rules which do not follow the usual assumptions. Instead, we will see how any AFS can be transformed so that all rules have a form $l = f(\boldsymbol{s}) \cdot \boldsymbol{t} \Rightarrow r$ with l and r both β-normal and l not containing leading free variables. We tackle standard assumptions (monomorphism and η-normality) which can be made about terms, and show that, after the first transformations, functional and applicative syntax are interchangeable. We aim to keep the complexity of the transformations minimal: a finite system remains finite after transforming, a monomorphic system remains monomorphic.

4 Polymorphism

In a first step towards simplifying the system, let us investigate polymorphism. To start, observe that polymorphism is only needed to define rules, not terms.

Theorem 1. *If a system is non-terminating, there is an infinite reduction on monomorphic terms.*

Proof. Given an infinite reduction $s_0 \Rightarrow_{\mathcal{R}} s_1 \Rightarrow_{\mathcal{R}} \ldots$, let θ be a type substitution which replaces all type variables in s_0 by a type constructor b of arity 0. Since $\Rightarrow_{\mathcal{R}}$ does not create type variables and is closed under type substitution, $s_0\theta \Rightarrow_{\mathcal{R}} s_1\theta \Rightarrow_{\mathcal{R}} \ldots$ is an infinite monomorphic reduction.

Polymorphism has its purpose in defining rules: any set of rules corresponds with a monomorphic set, but instantiating type variables leads to infinitely many rules. Finiteness is a high price to pay, since both humans and computers have trouble with the infinite. Nevertheless, from a perspective of reasoning we might as well use monomorphic rules, as long as we remember how they were generated.

Let a *rule scheme* be a pair $l \Rightarrow r$ of equal-typed (polymorphic) terms, such that all free variables and type variables of r also occur in l. Given a set R of rule schemes, let $\mathcal{R}_R = \{l\theta \Rightarrow r\theta | l \Rightarrow r \in R$ and θ a type substitution mapping all type variables in l to monomorphic types$\}$. The following is evident:

Theorem 2. *For a given set of rule schemes R, the set $\mathcal{R}_R$ is a set of monomorphic rules and $\Rightarrow_R$ is terminating if and only if $\Rightarrow_{\mathcal{R}_R}$ is.*

Henceforth, *rules* are understood to be monomorphic. *Rule schemes* may not be. $\mathcal{R}$ indicates a set of rules, R a set of rule schemes.

A pleasant consequence of using monomorphic rules is that type substitution is no longer needed to define the rewrite relation $\Rightarrow_{\mathcal{R}}$; $s \Rightarrow_{\mathcal{R}} t$ if either $s \Rightarrow_{\beta} t$ or $s = C[l\gamma]$ and $t = C[r\gamma]$ for some substitution γ, context C and rule $l \Rightarrow r$.

In the following sections we will define transformations on a set of rule schemes. Note that a set $\mathcal{R}$ of rules is always generated from a set of rule schemes, since even for monomorphic systems $R := \mathcal{R}$ is a suitable set.

5 Leading Variables

The presence of leading variables in the left-hand side of a rule $l \Rightarrow r$ (that is, subterms $x \cdot s$ where x is free in l) hinders techniques like dependency pairs[2] and makes it impossible to swap freely between functional and applicative notation (see also Section 7). We can avoid this problem by making application a function symbol: replace $s \cdot t$ everywhere by $@_{(\sigma \to \tau \times \sigma) \longrightarrow \tau}(s, t)$ and add $@_{(\alpha \to \varepsilon \times \alpha) \longrightarrow \tau}(x, y) \Rightarrow x \cdot y$ to R. The resulting system is terminating if and only if the original was. However, as discussed in Section 3, this transformation is not very good. A mostly applicative system would become almost impossible to handle with conventional techniques. In addition, the new rule scheme uses type variables, while the original system might be monomorphic. Thus, we will use a more complicated transformation that leads to an easier system.

Sketch of the Transformation. We sketch the idea for transforming a monomorphic system. Polymorphism brings in additional complications, but the rough idea is the same. First we instantiate headmost variables with functional terms: for any rule $l = C[x \cdot s] \Rightarrow r$ all possible rules $l[x := f(\boldsymbol{y}) \cdot \boldsymbol{z}] \Rightarrow r[x := f(\boldsymbol{y}) \cdot \boldsymbol{z}]$ are added. Now when a rule with leading variables is used, we can assume these variables are not instantiated with a functional term. Second, we introduce a symbol @ and replace occurrences $s \cdot t$ in any rule by $@(s, t)$ if s is not functional, and its type corresponds with the type of a leading variable in any left-hand side. We add rules $@_\sigma(x, y) \Rightarrow x \cdot y$ only for those $@_\sigma$ occurring in the changed rules. With this transformation, the applicative map rule $\mathtt{map}_{(\mathtt{nat} \to \mathtt{nat}) \to \mathtt{list(nat)} \to \mathtt{list(nat)}} \cdot F \cdot (\mathtt{cons} \cdot x \cdot y) \Rightarrow \mathtt{cons} \cdot (F \cdot x) \cdot (\mathtt{map} \cdot F \cdot y)$ either stays unchanged (if there are no rules with a leading variable of type $\mathtt{nat} \to \mathtt{nat}$ in the left-hand side) or becomes $\mathtt{map} \cdot F \cdot (\mathtt{cons} \cdot x \cdot y) \Rightarrow \mathtt{cons} \cdot @(F, x) \cdot (\mathtt{map} \cdot F \cdot y)$ (if there are).

5.1 Output Arity

Polymorphism complicates this transformation. Even with finite $\mathcal{F}$ there may be infinitely many terms of the form $f(\boldsymbol{x}) \cdot \boldsymbol{y}$. So assign to every $f_\sigma \in \mathcal{F}$ an integer $oa(f) \geq 0$; terms of the form $f(\boldsymbol{s}) \cdot t_1 \cdots t_m$ with $m \leq oa(f)$ are "protected".

[2] Leading variables in the right-hand side *also* complicate dependency pairs, but are harder to avoid; existing methods use various techniques to work around this issue.

The choice for oa is not fixed, but $oa(f) > 0$ may only hold for finitely many f. Typically, if $\sigma = (\boldsymbol{\tau}) \longrightarrow \rho_1 \to \ldots \to \rho_m \to \iota$ we choose $oa(f) = m$. We may also choose the highest m such that $f(\boldsymbol{s}) \cdot t_1 \cdots t_m$ occurs in a rule scheme. Or, in a (mostly) functional system we could choose $oa(f) = 0$ for all f; Transformations 1-2 have no effect then. A term is *limited functional* if it has the form $f(\boldsymbol{s}) \cdot t_1 \cdots t_m$ with $m < oa(f)$ (note that $f(\boldsymbol{s})$ is *not* limited functional if $oa(f) = 0$!).

Example 5. We follow the second guideline, so $oa(f)$ is the highest number m such that $f(\boldsymbol{s}) \cdot t_1 \cdots t_m$ occurs in a rule scheme. In the system $R_{\mathtt{map}}$ from Example 4 this gives output arity 0 for all symbols except `op`, which gets output arity 1.

To start, we adjust rules for the chosen output arity. For any rule $f(\boldsymbol{s}) \cdot t_1 \cdots t_n \Rightarrow r$ with $n < oa(f)$, we add a new rule $f(\boldsymbol{s}) \cdot t_1 \cdots t_n \cdot x \Rightarrow r \cdot x$. This is done because an application of the form $f(\boldsymbol{s}) \cdot \boldsymbol{t} \cdot u$ will be "protected" while $r \cdot u$ may not be.

Transformation 1 *(respecting output arity).* Given a set of rule schemes R, for every $l \Rightarrow r \in R$ with l limited functional add a rule scheme $l \cdot x \Rightarrow r \cdot x$ if this is well-typed (if $l : \alpha$ first apply the type substitution $[\alpha := \alpha \to \varepsilon]$). Repeat for all newly added rule schemes. This process terminates because $oa(f)$ is bounded, and the result, R^{res}, is finite if R is. $\Rightarrow_{R^{res}}$ and $\Rightarrow_R$ define the same relation.

Example 6. Since none of the left-hand sides of $R_{\mathtt{map}}$ are limited functional, Transformation 1 has no effect. If we had chosen, for example, $oa(\mathtt{pow}) = 2$ (this is allowed, even if there is no point in doing so), then we would have had to add four additional rules:

$$\begin{array}{llll} \mathtt{pow}_\sigma(F, 0) \cdot y & \Rightarrow (\lambda x.x) \cdot y & \mathtt{pow}_\sigma(F, \mathtt{s}(x)) \cdot y & \Rightarrow \mathtt{op}(F, \mathtt{pow}(F, x)) \cdot y \\ \mathtt{pow}_\tau(F, 0) \cdot y \cdot z & \Rightarrow (\lambda x.x) \cdot y \cdot z & \mathtt{pow}_\tau(F, \mathtt{s}(x)) \cdot y \cdot z & \Rightarrow \mathtt{op}(F, \mathtt{pow}(F, x)) \cdot y \cdot z \end{array}$$

Here, σ is the type declaration $(\alpha \to \alpha \times \mathtt{nat}) \longrightarrow \alpha \to \alpha$ and τ is $((\alpha \to \varepsilon) \to (\alpha \to \varepsilon) \times \mathtt{nat}) \longrightarrow (\alpha \to \varepsilon) \to \alpha \to \varepsilon$.

5.2 Filling in Head Variables

With this preparation, we can proceed to a larger transformation. Let $HV(s)$ be the set of those $x_\sigma \in FVar(s)$ where x_σ occurs at the head of an application in s (so $s = C[x_\sigma \cdot t]$ for some C, t). We will replace any rule $l = C[x_\sigma \cdot t] \Rightarrow r$ by a set of rules where a limited functional term $f(\boldsymbol{y}) \cdot \boldsymbol{z}$ is substituted for x_σ.

Transformation 2 *(filling in head variables).* For every rule scheme $l \Rightarrow r$ in R^{res}, every $x_\sigma \in HV(l)$, every function symbol $f_\tau \in \mathcal{F}$ and $n < oa(f)$ such that $(\ldots) \longrightarrow \alpha_1 \to \ldots \to \alpha_n \to \sigma$ unifies with τ, let θ, χ be their most general unifiers and add a rule scheme $l\theta\delta \Rightarrow r\theta\delta$, where $\delta = [x_{\sigma\theta} := f_{\tau\chi}(\boldsymbol{y}) \cdot z_1 \cdots z_n]$ (with $y_1, \ldots, y_k, z_1, \ldots, z_n$ fresh variables). Repeat this for the newly added rule schemes. If R^{res} is finite, this process terminates and the result, R^{fill}, is also finite. Otherwise define R^{fill} as the limit of the procedure.

Example 7. Following on Example 6, the only rule with a leading variable in the left-hand side is the η-reduction rule $\lambda x_{\alpha_1}.F_{\alpha_1 \rightarrow \alpha_2}\ x_{\alpha_1} \Rightarrow F_{\alpha_1 \rightarrow \alpha_2}$. Consequently, Transformation 2 completes after one step, with a single new rule; R^{fill} contains:

$$\begin{array}{lll@{\quad}lll}
\mathtt{map}(F,\mathtt{nil}) & \Rightarrow & \mathtt{nil} & \mathtt{op}(F,G)\cdot x & \Rightarrow & F\cdot(G\cdot x)\\
\mathtt{map}(F,\mathtt{cons}(x,y)) & \Rightarrow & \mathtt{cons}(F\cdot x,\mathtt{map}(F,y)) & \lambda x.F\cdot x & \Rightarrow & F\\
\mathtt{pow}(F,0) & \Rightarrow & \lambda x.x & \lambda x.\mathtt{op}(F,G)\cdot x & \Rightarrow & \mathtt{op}(F,G)\\
\mathtt{pow}(F,\mathtt{s}(x)) & \Rightarrow & \mathtt{op}(F,\mathtt{exp}(F,x)) & & &
\end{array}$$

It is not hard to see that R^{fill} and R^{res} generate the same relation. Moreover:

Lemma 2. *If $s \Rightarrow_{R^{fill}} t$ with a topmost step, then there are $l \Rightarrow r \in R^{fill}$, type substitution θ and substitution γ such that $s = l\theta\gamma$, $t = r\theta\gamma$ and $\gamma(x)$ is not limited functional for any $x \in HV(l)$.*

Proof. By definition of topmost step, there exist l, r, θ, γ such that $s = l\theta\gamma$ and $t = r\theta\gamma$; using induction on the size of $\{x_\sigma | x_\sigma \in HV(l) | \gamma(x_{\sigma\theta})$ is limited functional$\}$ and the definition of R^{fill} the Lemma follows without much effort.

5.3 Preparing Polymorphic Types

As suggested in the sketch of the transformation, we will introduce new symbols $@_\sigma$ only for those σ where it is necessary. Formally, let S be a set of functional types such that its closure under type substitution, S^c, contains all types σ where $x_\sigma \in HV(l)$ for some variable x and $l \Rightarrow r \in R^{fill}$. Transformation 4 will replace subterms $u \cdot v$ by $@(u,v)$ if $u : \tau \leq \sigma$ and u is not limited functional. There is one remaining problem: a subterm $u \cdot v$ where the type of u unifies with a type in S but is not an instance. We deal with this problem by adding some rule schemes.

Transformation 3 *(S-normalising the rules).* For every rule scheme $l \Rightarrow r \in R^{fill}$, add a rule scheme $l\theta \Rightarrow r\theta$ if either l or r has a subterm $s \cdot t$ with $s : \sigma$ not limited functional, and σ unifies with a type $\tau \in S$ such that $\tau \not\geq \sigma$. Here, θ and some χ are the most general unifiers of σ and τ. Repeat this for the newly added rules. If S and R^{fill} are both finite, this procedure terminates and the result, R^{normS}, is finite. Otherwise we define R^{normS} as the limit of the procedure.

Example 8. Consider our main Example; R^{fill} is given in Example 7. We must choose $S = \{\alpha \rightarrow \varepsilon\}$ due to the rule $\lambda x.F_{\alpha\rightarrow\varepsilon} \cdot x \Rightarrow F$. But $\alpha \rightarrow \varepsilon \geq$ any functional type, so Transformation 3 has no effect.

Again it is evident that the rewrite relations generated by R^{normS} and R^{fill} are the same. Moreover, we can derive the following (technical) result:

Lemma 3. *For $l \Rightarrow r \in R^{fill}$, type substitution θ, there are $l' \Rightarrow r' \in R^{normS}$ and type substitution χ such that $l\theta = l'\chi$, $r\theta = r'\chi$ and for $u \cdot v$ occurring in l' or r' with $u : \sigma$ not limited functional, either both $\sigma, \sigma\chi \in S^c$ or both $\sigma, \sigma\chi \notin S^c$.*

5.4 Explicit Application

Finally, then, we move on to the main substitution. For every type $\sigma \rightarrow \tau \in S$, introduce a new symbol $@_{(\sigma \rightarrow \tau \times \sigma) \longrightarrow \tau}$ and for all terms s define $\mathtt{exp}(s)$ as follows:

$$\begin{array}{ll} \mathtt{exp}(f(s_1,\dots,s_n)) & = f(\mathtt{exp}(s_1),\dots,\mathtt{exp}(s_n)) \\ \mathtt{exp}(x) & = x \ (x \text{ a variable}) \\ \mathtt{exp}(\lambda x.s) & = \lambda x.\mathtt{exp}(s) \\ \mathtt{exp}(s \cdot t) & = \begin{cases} \mathtt{exp}(s) \cdot \mathtt{exp}(t) \text{ if } s \text{ limited functional or type}(s) \notin S^c \\ @(\mathtt{exp}(s), \mathtt{exp}(t)) \ \ \text{otherwise} \end{cases} \end{array}$$

That is, subterms $s \cdot t$ are replaced by $@_\sigma(s,t)$, provided the split does not occur in a "protected" functional term, and s has a "dangerous" type.

Transformation 4 *(Embedding head symbols).* Let $R^{\text{noapp}} = \{\mathtt{exp}(l) \Rightarrow \mathtt{exp}(r) | l \Rightarrow r \in R^{normS}\} \cup \{@_{(\sigma \rightarrow \tau \times \sigma) \longrightarrow \tau}(x,y) \Rightarrow x \cdot y | \sigma \rightarrow \tau \in S\}$.

Transformations 1–4 preserve monomorphism and finiteness, yet R^{noapp} will not have leading (free) variables. We pose the main theorem of Section 5.

Theorem 3. *The rewrite relation $\Rightarrow_{R^{\text{noapp}}}$ generated by R^{noapp} is terminating if and only if $\Rightarrow_R$ is.*

Proof (Sketch). For one direction, if $s \Rightarrow_{R^{\text{noapp}}} t$ then also $s' \Rightarrow^{=}_{R^{normS}} t'$, where s', t' are s, t with occurrences of $@(u,v)$ replaced by $u \cdot v$. Equality only occurs if s has less @ symbols than t, so any infinite $\Rightarrow_{R^{\text{noapp}}}$ reduction leads to an infinite $\Rightarrow_{R^{normS}}$ reduction, and R^{normS} defines the same relation as $\mathcal{R}$. For the other direction, $s \Rightarrow_{R^{res}} t$ implies $\mathtt{exp}(s) \Rightarrow^{+}_{R^{\text{noapp}}} \mathtt{exp}(t)$ by induction on the size of s. For the induction step the only difficult case is when $s = u \cdot v \Rightarrow_{R^{res}} u' \cdot v$ with u limited functional while u' is not, but using Transformation 1 we can assume this is a topmost step. For the base case, if $s \Rightarrow_{R^{res}} t$ by a topmost rule step, we note that using Lemmas 2 and 3, $s = l\theta\gamma$ and $t = r\theta\gamma$ with $l \Rightarrow r \in R^{normS}$, $\gamma(x)$ not limited functional if $x \in HV(l\theta)$ and for any subterm $u \cdot v$ of l with $u : \tau$, either both τ and $\tau\theta \in S^c$ or neither. With these facts it is easy to show (using induction on the definition of $\mathtt{exp}$) that $\mathtt{exp}(l\theta\gamma) = \mathtt{exp}(l)\theta\gamma^{\mathtt{exp}}$ and $\mathtt{exp}(r)\theta\gamma^{\mathtt{exp}} \Rightarrow^{*}_{R^{\text{noapp}}} \mathtt{exp}(r\theta\gamma)$. If $s \Rightarrow_\beta t$ we use that $\mathtt{exp}(u)[x := \mathtt{exp}(v)] \Rightarrow^{*}_{R^{\text{noapp}}} \mathtt{exp}(u[x := v])$. Thus, any $\Rightarrow_{\mathcal{R}}$ reduction leads to a $\Rightarrow_{R^{\text{noapp}}}$ reduction of at least equal length.

Example 9. Considering our example with $S = \{\alpha \rightarrow \varepsilon\}$, R^{noapp} consists of:

$$\begin{array}{lll@{\qquad}lll} \mathtt{map}(F,\mathtt{nil}) & \Rightarrow & \mathtt{nil} & \mathtt{op}(F,G) \cdot x & \Rightarrow & @(F,@(G,x)) \\ \mathtt{map}(F,\mathtt{cons}(x,y)) & \Rightarrow & \mathtt{cons}(@(F,x),\mathtt{map}(F,y)) & \lambda x.@(F,x) & \Rightarrow & F \\ \mathtt{pow}(F,0) & \Rightarrow & \lambda x.x & \lambda x.\mathtt{op}(F,G) \cdot x & \Rightarrow & \mathtt{op}(F,G) \\ \mathtt{pow}(F,\mathtt{s}(x)) & \Rightarrow & \mathtt{op}(F,\mathtt{exp}(F,x)) & @(F,x) & \Rightarrow & F \cdot x \end{array}$$

6 Abstractions in Left-hand Sides and β-Redexes

The next step is to get rid of rule schemes $\lambda x.l \Rightarrow r$, where an abstraction is reduced directly; rules like this will form a definite blockade to working with η-expanded terms and they make it hard to define dependency pairs. The solution is very similar to the one employed in Section 5: we identify the types of all rule schemes of this form, and replace abstractions $\lambda x.s$ of such a type σ by $\Lambda_\sigma(\lambda x.s)$, where Λ_σ is a new function symbol. As a side bonus, we will get rid of any remaining β-redexes in the rule schemes (note that the transformations of Section 5 may already have removed such redexes).

Formally, let Q be a set of types such that its closure under type substitution, Q^c, contains all types σ such that $\lambda x.l : \sigma \Rightarrow r \in R$, or $(\lambda x.s) \cdot t$ occurs in any rule scheme. We could choose the set of all such types, or for instance $\{\alpha \to \varepsilon\}$. As before we need to prepare polymorphic rule schemes for a type match.

Transformation 5 *(Q-normalising the rules).* . For every rule scheme $l \Rightarrow r \in R$, add a rule scheme $l\theta \Rightarrow r\theta$ if either l or r has a subterm $\lambda x.s : \sigma$, and σ unifies with a type $\tau \in Q$ such that $\tau \not\geq \sigma$. Here, θ and some χ are the most general unifiers of σ and τ. Repeat this for the newly added rules. If Q and R are both finite, this procedure terminates and the result, R^{normQ}, is finite. Otherwise define R^{normQ} as the limit of the procedure.

We can derive a Lemma very similar to Lemma 3, but it would not bring much news. Let us instead pass straight to the main transformation:

$$\begin{array}{ll}
\mathtt{expL}(f(s_1,\dots,s_n)) & = f(\mathtt{expL}(s_1),\dots,\mathtt{expL}(s_n)) \\
\mathtt{expL}(s \cdot t) & = \mathtt{expL}(s) \cdot \mathtt{expL}(t) \\
\mathtt{expL}(x) & = x \ (x \text{ a variable}) \\
\mathtt{expL}(\lambda x.s) & = \begin{cases} \Lambda_{(\sigma)\longrightarrow\sigma}(\lambda x.\mathtt{expL}(s)) & \text{if } \lambda x.s : \sigma \in Q^c \\ \lambda x.\mathtt{expL}(s) & \text{otherwise} \end{cases}
\end{array}$$

Transformation 6 *(Marking Abstractions).* $R^\Lambda := \{\mathtt{expL}(l) \Rightarrow \mathtt{expL}(r) | l \Rightarrow r \in R^{normQ}\} \cup \{\Lambda_{(\sigma)\longrightarrow\sigma}(x) \Rightarrow x | \sigma \in S\}$

It is evident that R^Λ has no rule schemes of the form $\lambda x.l \Rightarrow r$ and is β-normal. Moreover, its termination is equivalent to termination of the original system.

Theorem 4. *$\Rightarrow_{R^\Lambda}$ is terminating if and only if $\Rightarrow_R$ is.*

Proof. It is not too hard to derive that $s \Rightarrow_R t$ implies $\mathtt{expL}(s) \Rightarrow^+_{R^\Lambda} \mathtt{expL}(t)$, using that $\mathtt{expL}(C[u]) = \mathtt{expL}(C)[\mathtt{expL}(u)]$ and a separate induction for the top step (using Transformation 5 to choose the right rule and type substitution). Defining s', t' as s, t with occurrences of any Λ_σ erased, it is also obvious that $s \Rightarrow_{R^\Lambda} t$ implies $s' \Rightarrow^=_R t'$, with equality only if the former was Λ-erasing.

Example 10. Continuing the transformation of $R_{\mathtt{map}}$, we choose $Q = \{\alpha \to \varepsilon\}$ (we have no other choice, because of the rule $\lambda x.@(F_{\alpha\to\varepsilon}, x) \Rightarrow F_{\alpha\to\varepsilon}$). Transformation 5 has no effect, and Transformation 6 introduces Λ around all abstractions:

$$\begin{array}{llll}
\mathtt{map}(F, \mathtt{nil}) & \Rightarrow \mathtt{nil} & \Lambda(\lambda x.@(F, x)) & \Rightarrow F \\
\mathtt{map}(F, \mathtt{cons}(x, y)) & \Rightarrow \mathtt{cons}(@(F, x), \mathtt{map}(F, y)) & \Lambda(\lambda x.\mathtt{op}(F, G) \cdot x) & \Rightarrow \mathtt{op}(F, G) \\
\mathtt{pow}(F, 0) & \Rightarrow \Lambda(\lambda x.x) & \Lambda_{\alpha\to\varepsilon}(F) & \Rightarrow F \\
\mathtt{pow}(F, \mathtt{s}(x)) & \Rightarrow \mathtt{op}(F, \mathtt{exp}(F, x)) & @(F, x) & \Rightarrow F \cdot x \\
\mathtt{op}(F, G) \cdot x & \Rightarrow @(F, @(G, x)) & &
\end{array}$$

Summing Up. Combining Sections 5 and 6, we can transform a set of rule schemes, without affecting termination, to satisfy the following properties:

1. both sides of rule schemes are β-normal
2. left-hand sides l have no subterms $x \cdot s$ with x a free variable
3. left-hand sides have the form $f(l_1, \dots, l_n) \cdot l_{n+1} \cdots l_m$ with $m \geq n$

Property (3) holds by elimination: after transforming, the left-hand side of a rule scheme is neither an abstraction, nor an application headed by an abstraction or variable. If it is a variable, $x \Rightarrow r \in R$, the AFS is non-terminating and (since termination is all we are interested in) we might replace R by the set $\{a \Rightarrow a\}$.

Henceforth, rule schemes are assumed to satisfy the requirements listed above.

7 Currying

Let us turn our eyes to the status of application. As mentioned in Section 3, an applicative AFS cannot be handled with most existing termination techniques, nor can we naively turn it into a functional system. The issues are partial application (an applicative `map` system has terms like $\mathtt{map} \cdot \mathtt{s}$ which have no functional counterpart) and leading free variables (a terminating rule $g \cdot (x \cdot 0) \Rightarrow g \cdot f(0)$ has an applicative counterpart $g \cdot (x \cdot 0) \Rightarrow g \cdot (f \cdot 0)$ which is not terminating). However, we have excluded rules with leading free variables in the left-hand side. The issue of partial application can be dealt with using η-expansion.

There are two directions we might take. Usually, we would like to *uncurry* an applicative system, transforming a term $f \cdot s \cdot t$ into $f(s, t)$. Such a form is more convenient in for instance path orderings, or to define argument filterings. On the other hand, we will have to deal with application anyway, since it is part of the term syntax; to simplify the formalism it might be a good move to *curry* terms, making the system entirely applicative.

Transformation 7 *(Currying).* Let R be a set of rules schemes over a set of function symbols $\mathcal{F}$. We define the following mapping on type declarations: $\mathtt{flat}((\sigma_1 \times \dots \sigma_n) \longrightarrow \tau) = \sigma_1 \to \dots \to \sigma_n \to \tau$. Next we define the mapping $\mathtt{flat}$ from functional terms over $\mathcal{F}$ to applicative terms over the 'flattened version' of $\mathcal{F}$, notation $\mathcal{F}^{\mathtt{flat}}$, as follows:

$$\begin{array}{ll}
\mathtt{flat}(f_\sigma(s_1,\dots,s_n)) & = f_{\mathtt{flat}(\sigma)} \cdot \mathtt{flat}(s_1) \cdots \mathtt{flat}(s_n) \\
\mathtt{flat}(\lambda x.s) & = \lambda x.\mathtt{flat}(s) \\
\mathtt{flat}(s \cdot t) & = \mathtt{flat}(s) \cdot \mathtt{flat}(t) \\
\mathtt{flat}(x) & = x \quad (x \text{ a variable})
\end{array}$$

The flattened version $R^{\mathtt{flat}}$ of the set of rule schemes R consists of the rule scheme $\mathtt{flat}(l) \Rightarrow \mathtt{flat}(r)$ for every rule scheme $l \Rightarrow r$ in R.

Theorem 5. *$\Rightarrow_R$ is well-founded on terms over $\mathcal{F}$ if and only if $\Rightarrow_{R^{\mathtt{flat}}}$ is well-founded on terms over $\mathcal{F}^{\mathtt{flat}}$.*

Proof. It is easy to see that $s \Rightarrow_R t$ implies $\mathtt{flat}(s) \Rightarrow_{R^{\mathtt{flat}}} \mathtt{flat}(t)$ (with induction on the size of s, and a separate induction for topmost steps to see that flattening is preserved under substitution); this provides one direction. For the other, let $\mathtt{flat}^{-1}$ be the "inverse" transformation of $\mathtt{flat}$, which maps occurrences of $f \cdot s_1 \cdots s_k$ with $f_{(\sigma_1 \times \ldots \times \sigma_n) \longrightarrow \tau} \in \mathcal{F}$ to $\lambda x_{k+1} \ldots x_n.f(\mathtt{flat}^{-1}(s_1),\dots,\mathtt{flat}^{-1}(s_k), x_{k+1},\dots,x_n)$ if $k < n$ or to $f(\mathtt{flat}^{-1}(s_1),\dots,\mathtt{flat}^{-1}(s_n)) \cdot \mathtt{flat}^{-1}(s_{n+1}) \cdots \mathtt{flat}^{-1}(s_k)$ otherwise. It is not hard to see that $\mathtt{flat}^{-1}(s)[\boldsymbol{x} := \mathtt{flat}^{-1}(\boldsymbol{t})] \Rightarrow_\beta^* \mathtt{flat}^{-1}(s[\boldsymbol{x} := \boldsymbol{t}])$, and this $\Rightarrow_\beta^*$ is an equality if $HV(s) = \emptyset$. Therefore, and because $\mathtt{flat}^{-1}(R^{\mathtt{flat}})$ is exactly R, $\mathtt{flat}^{-1}(s) \Rightarrow_R^+ \mathtt{flat}^{-1}(t)$ holds if $s \Rightarrow_{R^{\mathtt{flat}}} t$.

Note the *if and only if* in Theorem 5. Because of this equivalence the theorem works in two ways. We can turn a functional system applicative, but also turn an applicative system functional, simply by taking the inverse of Transformation 7. For an applicative system, there are usually many sets of corresponding functional rules, all of which are equivalent for the purpose of termination.

Example 11. Our running example can be transformed into the applicative AFS:

$$\begin{array}{llll}
\mathtt{pow} \cdot F \cdot 0 & \Rightarrow \Lambda \cdot (\lambda x.x) & \Lambda \cdot (\lambda x.@ \cdot F \cdot x) & \Rightarrow F \\
\mathtt{pow} \cdot F \cdot (\mathtt{s} \cdot x) & \Rightarrow \mathtt{op} \cdot F \cdot (\mathtt{exp} \cdot F \cdot x) & \Lambda \cdot (\lambda x.\mathtt{op} \cdot F \cdot G \cdot x) & \Rightarrow \mathtt{op} \cdot F \cdot G \\
\mathtt{op} \cdot F \cdot G \cdot x & \Rightarrow @ \cdot F \cdot (@ \cdot G \cdot x) & @ \cdot F \cdot x & \Rightarrow F \cdot x \\
\mathtt{map} \cdot F \cdot \mathtt{nil} & \Rightarrow \mathtt{nil} & \Lambda \cdot F & \Rightarrow F \\
\mathtt{map} \cdot F \cdot (\mathtt{cons} \cdot x \cdot y) & \Rightarrow \mathtt{cons} \cdot (@ \cdot F \cdot x) \cdot (\mathtt{map} \cdot F \cdot y) & &
\end{array}$$

Related Work. In first-order rewriting, the question whether properties such as confluence and termination are preserved under currying or uncurrying is studied in [5,6,2]. In [6] a currying transformation from (functional) term rewriting systems (TRSs) into applicative term rewriting systems (ATRSs) is defined; a TRS is terminating if and only if its curried form is. In [2], an uncurrying transformation from ATRSs to TRSs is defined that can deal with partial application and leading variables, as long as they do not occur in the left-hand side of rewrite rules. This transformation is sound and complete with respect to termination.

However, these results do not apply to AFSs, both due to the presence of typing and because AFSs use a mixture of functional and applicative notation. We may for instance have terms of the form $f(x) \cdot y$, and currying might introduce new interactions via application.

8 η-Expansion

Finally, we consider η-expansion. It would often be convenient if we could assume that every term of some functional type $\sigma \rightarrow \tau$ has the form $\lambda x_\sigma.s$, which only reduces if its subterm s does. This is the case if we work modulo η, equating $s : \sigma \rightarrow \tau$, with s not an abstraction, to $\lambda x_\sigma.(s \cdot x_\sigma)$. As is well-known, simply working modulo η in the presence of β-reduction causes problems. Instead, we will limit reasoning to η-long terms.

Theorem 6. *Let $\mathcal{R}$ be a set of rules in restricted η-long form, that is, $l = l\uparrow^{\eta}_{FVar(l)}$ and $r = r\uparrow^{\eta}_{FVar(r)}$ for every rewrite rule $l \Rightarrow r$ in $\mathcal{R}$. Then the set of η-long terms is closed under rewriting. Moreover, the rewrite relation $\Rightarrow_{\mathcal{R}}$ is terminating on terms iff it is terminating on η-long terms.*

Proof. Evidently, if $\Rightarrow_{\mathcal{R}}$ is terminating then it is terminating on all η-long terms. For the less obvious direction, we see that $s \Rightarrow_{\mathcal{R}} t$ implies $s\uparrow^{\eta} \Rightarrow^{+}_{\mathcal{R}} t\uparrow^{\eta}$. Hence any infinite reduction can be transformed to an infinite reduction on η-long terms. Writing $\gamma^{\uparrow} := \{x \mapsto \gamma(x)\uparrow^{\eta} | x \in \text{dom}(\gamma)\}$, a simple inductive reasoning shows that $s\uparrow^{\eta}_{V}\gamma^{\uparrow} \Rightarrow^{*}_{\beta} s\gamma\uparrow^{\eta}$ if $V = \text{dom}(\gamma)$, and this is an equality if $HV(s) = \emptyset$. Thus, if $s \Rightarrow_{\mathcal{R}} t$ by a topmost reduction, then also $s\uparrow^{\eta} = l\gamma\uparrow^{\eta} = l\uparrow^{\eta}_{FVar(l)}\gamma^{\uparrow} = l\gamma^{\uparrow} \Rightarrow_{\mathcal{R}} r\gamma^{\uparrow} = r\uparrow^{\eta}_{FVar(r)}\gamma^{\uparrow} \Rightarrow_{\beta} r\gamma\uparrow^{\eta} = t\uparrow^{\eta}$. This forms the base case for an induction on s, which proves $s\uparrow^{\eta} \Rightarrow^{+}_{\mathcal{R}} t\uparrow^{\eta}$ whenever $s \Rightarrow_{\mathcal{R}} t$.

The requirement that the rules should be η-long is essential. Consider for example the system with a single rule $f_{o \rightarrow o} \cdot x_o \Rightarrow g_{(o \rightarrow o) \rightarrow o} \cdot f_{o \rightarrow o}$. The relation generated by this rule is terminating, but evidently the set of η-long terms is not closed under rewriting. The η-long variation of this rule, $f_{o \rightarrow o} \cdot x_o \Rightarrow g_{(o \rightarrow o) \rightarrow o} \cdot (\lambda y_o.f_{o \rightarrow o} \cdot y_o)$, is not terminating, as the left-hand side can be embedded in the right-hand side. This example is contrived, but it shows that we cannot be careless with η-expansion. However, when developing methods to prove termination of a system the most essential part of any transformation is to preserve *non-termination.* At the price of completeness, we can use Transformation 8:

Transformation 8 *(η-expanding rules).* Let $\mathcal{R}$ be a set of rules. Define $\mathcal{R}^{\uparrow}$ to be the set consisting of the rules $(l \cdot x^{1}_{\sigma_1} \cdots x^{n}_{\sigma_n})\uparrow^{\eta}_{V} \Rightarrow (r \cdot x^{1}_{\sigma_1} \cdots x^{n}_{\sigma_n})\uparrow^{\eta}_{V}$, for every rule $l \Rightarrow r$ in $\mathcal{R}$, with $l : \sigma_1 \rightarrow \ldots \sigma_n \rightarrow \iota$, all $x^{i}_{\sigma_i}$s fresh variables, and $V = FVar(l) \cup \{x^{1}_{\sigma_1}, \ldots, x^{n}_{\sigma_n}\}$.

The proof of the following theorem is a straightforward adaptation of the proof of Theorem 6.

Theorem 7. *If the rewrite relation generated by $\mathcal{R}^{\uparrow}$ is terminating on η-long terms, then the relation generated by $\mathcal{R}$ is terminating on the set of terms.*

Note that Theorems 6 and 7 concern *rules*, not rule schemes. The η-expansion of a terminating set of rule schemes may not be terminating, as demonstrated by the system with $R = \{f_{\alpha \rightarrow \alpha} \cdot g_\alpha \Rightarrow h_\alpha,\ h_{\mathtt{nat} \rightarrow \mathtt{nat}} \cdot 0_{\mathtt{nat}} \Rightarrow f_{(\mathtt{nat} \rightarrow \mathtt{nat}) \rightarrow \mathtt{nat} \rightarrow \mathtt{nat}} \cdot g_{\mathtt{nat} \rightarrow \mathtt{nat}} \cdot 0_{\mathtt{nat}}\}$. Thus, η-expansion is mainly useful on monomorphic systems, or for termination methods which, given rule schemes R, prove termination of $\mathcal{R}_R^{\uparrow}$.

9 Conclusions

We have seen various techniques to transform AFSs, essentially making it possible to pose restrictions on terms and rule schemes without losing generality. Considering existing results, this has various applications:

Applicative terms As mentioned before, most applicative systems cannot be dealt with directly. Consider for example the system with symbols $\mathtt{split}_{\mathtt{nat} \rightarrow \mathtt{tuple}}$ and $\mathtt{pair}_{\alpha \rightarrow \varepsilon \rightarrow \mathtt{tuple}}$ which has the following rule:

$$\mathtt{split} \cdot (x_{\mathtt{nat} \rightarrow \mathtt{nat}} \cdot y_{\mathtt{nat}}) \Rightarrow \mathtt{pair} \cdot x_{\mathtt{nat} \rightarrow \mathtt{nat}} \cdot y_{\mathtt{nat}}$$

Even the computability path ordering [1], which is the latest definition in a strengthening line of path orderings, cannot deal with this rule. However, using Transformations 1–4 we introduce $@_{(\mathtt{nat} \rightarrow \mathtt{nat} \times \mathtt{nat}) \longrightarrow \mathtt{nat}}$ and the system becomes:

$$\mathtt{split} \cdot @(x, y) \Rightarrow \mathtt{pair} \cdot x \cdot y \qquad @(x, y) \Rightarrow x \cdot y$$

This system has the same curried form as:

$$\mathtt{split}(@(x, y)) \Rightarrow \mathtt{pair}(x, y) \qquad @(x, y) \Rightarrow x \cdot y$$

Consequently, termination of one implies termination of the other by Theorem 5. But the latter is immediate with HORPO [4], using a precedence $@ >_{\mathcal{F}} \mathtt{pair}$.

CPO The latest recursive path ordering, CPO, is defined only for monotonic systems where all symbols have a data type as output type. It cannot, for instance, deal with a system with rules:

$$\begin{aligned} \mathtt{emap}(F, \mathtt{nil}) &\Rightarrow \mathtt{nil} \\ \mathtt{emap}(F, \mathtt{cons}(x, y)) &\Rightarrow \mathtt{cons}(F \cdot x, \mathtt{emap}(\mathtt{twice}(F), y)) \\ \mathtt{twice}(F) \cdot x &\Rightarrow F \cdot (F \cdot x) \end{aligned}$$

Here, $\mathtt{twice}$ has type declaration $(\mathtt{nat} \rightarrow \mathtt{nat}) \longrightarrow \mathtt{nat} \rightarrow \mathtt{nat}$. By Theorem 6 we can η-expand these rules, the result of which has the same curried form as:

$$\begin{aligned} \mathtt{emap}(F, \mathtt{nil}) &\Rightarrow \mathtt{nil} \\ \mathtt{emap}(F, \mathtt{cons}(x, y)) &\Rightarrow \mathtt{cons}(F \cdot x, \mathtt{emap}(\lambda z.\mathtt{twice}(F, z), y)) \\ \mathtt{twice}(F, x) &\Rightarrow F \cdot (F \cdot x) \end{aligned}$$

Thus, if this system can be proved terminating with CPO (which it can, if a reverse lexicographical ordering is used), the original system is terminating. CPO can be applied on any monomorphic system in this way, although the transformation may lose termination due to the η-expansion.

Dependency Pairs Since rules can be assumed to have a form $f(l_1, \ldots, l_n) \cdot l_{n+1} \cdots l_m$, the dependency pair method for AFSs in [8] is now applicable without restrictions other than monotonicity; a transformation tool which has Transformations 1–6 built into the input module could build around a dependency pair framework without losing out.

Summary and Future Work. In this paper we discussed transformations which simplify Algebraic Functional Systems significantly. We saw that polymorphism only has a function in defining rule schemes, that rule schemes can be assumed to be β-normal and that there is no need for leading free variables in the left-hand side of rules. We know that applicative and functional notation can be interchanged, and rule schemes can be assumed to have a form $f \cdot l_1 \cdots l_n \Rightarrow r$ with f a function symbol. Moreover, when we are interested only in proving termination, we may η-expand the rules and restrict attention to η-long terms.

A monomorphic version of the transformations given here was implemented in WANDA v1.0 [7], which participated in the Termination Competition 2010 [13].

In the future, we intend to look further into other formalisms, and give conditions and techniques to transfer results across.

Acknowledgement

We are indepted to the anonymous referees for their remarks which helped to improve the paper.

References

1. Blanqui, F., Jouannaud, J.-P., Rubio, A.: The computability path ordering: The end of a quest. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 1–14. Springer, Heidelberg (2008)
2. Hirokawa, N., Middeldorp, A., Zankl, H.: Uncurrying for termination. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 667–681. Springer, Heidelberg (2008)
3. Jouannaud, J.-P., Okada, M.: A computation model for executable higher-order algebraic specification languages. In: 6th IEEE Symposium on Logic In Computer Science, pp. 350–361. IEEE Press, Amsterdam (1991)
4. Jouannaud, J.-P., Rubio, A.: The higher-order recursive path ordering. In: 14th IEEE Symposium on Logic In Computer Science, pp. 402–411. IEEE Press, Los Alamitos (1999)
5. Kahrs, S.: Confluence of curried term-rewriting systems. Journal of Symbolic Computing 19, 601–623 (1995)
6. Kennaway, R., Klop, J.W., Sleep, M.R., de Vries, F.J.: Comparing curried and uncurried rewriting. Journal of Symbolic Computing 21(1), 15–39 (1996)
7. Kop, C.: Wanda, http://www.few.vu.nl/~kop/code.html
8. Kop, C., van Raamsdonk, F.: Higher order dependency pairs for algebraic functional systems. In: RTA 2011 (to appear, 2011)
9. Kusakari, K., Isogai, Y., Sakai, M., Blanqui, F.: Static dependency pair method based on strong computability for higher-order rewrite systems. IEICE Transactions on Information and Systems 92, 2007–2015 (2009)
10. Nipkow, T.: Higher-order critical pairs. In: 6th IEEE Symposium on Logic in Computer Science, pp. 342–349. IEEE Press, Amsterdam (1991)
11. van de Pol, J.C.: Termination of Higher-order Rewite Systems. PhD thesis, University of Utrecht (1996)
12. Sakai, M., Watanabe, Y., Sakabe, T.: An extension of the dependency pair method for proving termination of higher-order rewrite systems. IEICE Transactions on Information and Systems E84-D(8), 1025–1032 (2001)
13. Termination Portal (Wiki), http://www.termination-portal.org/

Hadamard Matrices, Designs and Their Secret-Sharing Schemes

Christos Koukouvinos[1], Dimitris E. Simos[1], and Zlatko Varbanov[2]

[1] Department of Mathematics, National Technical University of Athens, Zografou 15773, Athens, Greece
[2] Department of Information Technologies, University of Veliko Tarnovo, 2 T.Tarnovski Str, 5000, Bulgaria
{ckoukouv,dsimos}@math.ntua.gr,
zl.varbanov@uni-vt.bg

Abstract. In this paper, we give some methods to generate new secret-sharing schemes from Hadamard matrices derived through orthogonal 3-designs. A close connection of Hadamard designs and secret-sharing schemes is shown. In addition, we survey some of the most prolific construction methods for Hadamard matrices thus providing the necessary structures to describe a two-part secret-sharing scheme based on Hadamard designs. Furthermore, we exhibit how some algebraic aspects of secret-sharing cryptography are interpreted in terms of combinatorial design theory, such as the access structure and the security of the secret-sharing schemes.

Keywords: Hadamard matrices, Hadamard designs, construction, secret–sharing schemes.

1 Introduction – Preliminaries

A *Hadamard matrix* of order n, denoted by $H(n)$, is an $n \times n$ matrix with entries from $\{1, -1\}$ with the property

$$HH^T = nI_n$$

where H^T stands for the transpose matrix of H and I_n is the identity matrix of order n. The Hadamard property entails that the rows (and columns) of a Hadamard matrix are orthogonal. It is well known that if n is the order of a Hadamard matrix then n is necessarily $1, 2$ or a multiple of 4, see [23]. A Hadamard matrix is said to be *semi-normalized* if all entries in its first row are equal to 1, while *normalized* if all entries in both first row and column are equal to 1. Two Hadamard matrices are *equivalent* if one can be transformed into the other by a series of row or column permutations and negations. The Hadamard conjecture stipulates that there exists a Hadamard matrix of order $4m$, for every positive integer m. The Hadamard conjecture is one of the basic unsolved problems in Discrete Mathematics [12]. The smallest order n for which a Hadamard matrix is not known, is $n = 668$.

Hadamard matrices have important applications in Statistics, Coding Theory, Cryptography, Communication Systems and numerous other areas. The Computational Algebra System, MAGMA, maintains a database for inequivalent Hadamard matrices of small orders [3], while lower bounds for inequivalent Hadamard matrices of small orders can be reached in [18]. For authoritative information on Hadamard matrices and their applications we refer to [8], [12], [23], [28].

A $t-(v,k,\lambda)$ **design** is a pair $(\mathcal{P},\mathcal{B})$ where $\mathcal{P}$ is a set of v elements, called points, and $\mathcal{B}$ is a collection of distinct subsets of $\mathcal{P}$ of size k, called blocks, such that every subset of points of size t is contained in exactly λ blocks. Any $t-(v,k,\lambda)$ design is also an $s-(v,k,\lambda_s)$ design for $s \leq t$, where $\lambda_s = \frac{(v-s)}{(k-s)}\lambda_{s+1}$ and $\lambda_t = \lambda$. Moreover, if $(\mathcal{P},\mathcal{B})$ is a $t-(v,k,\lambda)$ design then $(\mathcal{P}\{x\}, DER_x(\mathcal{B}))$ is a $(t-1)-(v-1,k-1,\lambda)$ design, called the derived design for $(\mathcal{P},\mathcal{B})$, where $x \in \mathcal{P}$ and $DER_x(\mathcal{B}) = \{B\{x\} : x \in B \in \mathcal{B}\}$ [1]. Any 3-design with $v = 4m$, $k = 2m$, and $\lambda = m-1$, is called a Hadamard 3-design, because of the association with Hadamard matrices, as we will explain presently. Further, if $\mathcal{D}$ is a Hadamard 3-design, then each of its derived designs, $\mathcal{D}_P$, obtained by omitting a point P and all the blocks that are not incident with P, is symmetric. Both $\mathcal{D}_P$ and $\bar{\mathcal{D}}_P$, are called Hadamard 2-designs. Their parameters are, respectively, $(4m-1, 2m-1, m-1)$ and $(4m-1, 2m, m)$. For more details we refer to [6] and [1].

Let H be a Hadamard matrix of size $4m$ and let $r = (r_1, r_2, ..., r_{4m})$ be any row of H. Then for any other row s of H, $l_s = \{j | s_j = r_j\}$ is a $2m$-subset of $\mathcal{P} = \{1, 2, ..., 4m\}$, and the same is true for $\bar{l}_s = \mathcal{P} - l_s = \{j | s_j \neq r_j\}$. It is well known and elementary to verify (see [13]) that the collection

$$\mathcal{B}(H(r)) = \{l_s | s \neq r\} \cup \{\bar{l}_s | s \neq r\} \tag{1}$$

forms the block set of a Hadamard 3-design. To get a 2-design, we pick a point j, then retain the block l_s if $j \in l_s$, and $\bar{l}_s$ if not.

In the present work we study Hadamard matrices of order $n = 4m$ and their corresponding 3-designs. Using them, we describe a two-part secret-sharing scheme based on Hadamard 3-designs. A **secret-sharing scheme** is a way of sharing a secret among a finite set of people or entities such that only some distinguished subsets of these have access to the secret. The collection Γ of all such distinguished subsets is called the **access structure** of the scheme. A perfect secret-sharing scheme for Γ is a method by which the shares are distributed to the parties such that: (1) any subset in Γ can reconstruct the secret from its shares, and (2) any subset not in Γ can never reveal any partial information on the secret (in the information theoretic sense). Secret-sharing schemes were first introduced by Blakley [2] and Shamir [24] for the threshold case, that is, for the case where the subsets that can reconstruct the secret are all the sets whose cardinality is at least a certain threshold. In this work we consider some special properties of Hadamard matrices that play an important role in our study when enumerating access structures by size.

The paper is organized as follows. In Section 2 we consider fast constructions for Hadamard matrices. In Section 3 we describe the proposed two-part secret-sharing scheme and its access structure. In the last Section an algorithmic construction for secret-sharing schemes from Hadamard matrices is given.

2 Some Constructions for Hadamard Matrices

There is a large number of constructions for Hadamard matrices which can be roughly classified in three types:

- multiplication (recursion) theorems
- direct constructions
- "plug-in" methods

A good overview of the topic appears in [23]. The constructions given here by no means exhaust those known, but suffice to give a Hadamard matrix of each admissible order less up to 100.

2.1 Hadamard Matrices Obtained via Kronecker Product

The foundation of most multiplicative methods is the Kronecker product of two matrices. That is, if $A = (a_{ij})$ is a $m \times p$ matrix and $B = (b_{ij})$ is an $n \times q$ matrix, then the Kronecker product $A \otimes B$ is the $mn \times pq$ matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1p}B \\ a_{21}B & a_{22}B & \dots & a_{2p}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \vdots & a_{mp}B \end{bmatrix}.$$

Jacques Hadamard showed the following tensor (also called Kronecker) product construction.

Theorem 1 (Hadamard [11]). *If H_1 and H_2 are Hadamard matrices of orders m and n respectively, then the their Kronecker product $H_1 \otimes H_2$ is a Hadamard matrix of order mn.*

The first major family of Hadamard matrices however was found by Sylvester in his pioneering paper [25] for all orders 2^k, $k \geq 1$. In terms of the Kronecker product construction, his results can now be detailed by setting $S_1 = \begin{bmatrix} 1 & 1 \\ 1 & - \end{bmatrix}$ as the Kronecker product $S_1 \otimes H(n)$.

Lemma 1 (Sylvester [25]). *The Sylvester Hadamard matrices are the matrices in the family $\{S_k = \otimes^k S_1 : k \geq 1\}$.*

2.2 Paley Type Hadamard Matrices

Another major family of Hadamard matrices is the so called *Paley Type Hadamard matrices.* These families of Hadamard matrices were found by Paley's [22] direct construction using the quadratic residues (that is, the non-zero perfect squares) in a finite field $GF(q)$ of odd order. In the field $GF(q)$, half the non-zero elements are quadratic residues of squares and the other half are quadratic non-residues of non-squares. In particular, $+1$ is a square and -1 is a non-square only if $q \equiv 3 \pmod 4$.

The quadratic character of $GF(q)$ is the function χ given by

$$\chi(x) = \begin{cases} 0 & \text{if } x = 0; \\ +1 & \text{if } x \text{ is a quadratic residue;} \\ -1 & \text{if } x \text{ is a quadratic non-residue.} \end{cases}$$

Theorem 2 (Paley [22]). *For q an odd prime power, and an ordering $\{g_0 = 0, g_1, \ldots, g_{q-1}\}$ of $GF(q)$, set $Q = [\chi(g_i - g_j)]_{0 \le i,j < q}$ and set $S = \begin{bmatrix} 0 & \boldsymbol{1} \\ \boldsymbol{1}^T & Q \end{bmatrix}$, where $\boldsymbol{1}$ is the all-1s vector of length q.*

1. *(Paley Type I Hadamard matrix) If $q \equiv 3 \pmod 4$ then*

$$P_q = \begin{bmatrix} 1 & \boldsymbol{-1} \\ \boldsymbol{1}^T & Q + I_q \end{bmatrix}$$

is a Hadamard matrix of order $(q+1)$.

2. *(Paley Type II Hadamard matrix) If $q \equiv 1 \pmod 4$ then*

$$P_q^{'} = \begin{bmatrix} S + I_{q+1} & S - I_{q+1} \\ S - I_{q+1} & -S - I_{q+1} \end{bmatrix}$$

is a Hadamard matrix of order $2(q+1)$.

Remark 1. Paley Type I Hadamard matrices of order $q+1$ are strongly connected to $2-(q, \frac{q-1}{2}, \frac{q-3}{4})$ designs. For example, normalize the P_q; the resulting design can be described as follows: the point set is $GF(q)$; one block is the set J of non-zero squares (quadratic residues) in $GF(q)$, and the others are its translates $J + x = \{j + x : j \in J\}$ for $x \in GF(q)$. As noted earlier, these designs are also called Hadamard 2-designs.

2.3 Hadamard Matrices Obtained from Two Circulant Submatrices

We now present a method to "plug-in" matrices in a suitable array, firstly given by Yang [27].

Theorem 3 (Yang [27]). *If A and B are $n \times n$ circulant matrices with elements ± 1 that satisfy:*

$$AA^T + BB^T = 2nI_n \tag{2}$$

then the matrix $H = \begin{bmatrix} A & B \\ -B^T & A^T \end{bmatrix}$ is a Hadamard matrix of order $2n$.

In order to find suitable circulant submatrices that satisfy the additive property in (2) we use an important result that comes from sequences with zero non-periodic autocorrelation (NPAF), as outlined in [20].

Remark 2. If there are two sequences A and B of length n with entries from $\{\pm 1\}$ with zero non-periodic autocorrelation function, then these sequences can be used as the first rows of circulant matrices which can be used in the Yang array to form a Hadamard matrix of order $2n$.

We adopt the following definitions from [17].

Definition 1. *Let $A = [a_1, a_2, \dots, a_n]$ be a sequence of length n. The non-periodic autocorrelation function, NPAF, $N_A(s)$ is defined as:*

$$N_A(s) = \sum_{i=1}^{n-s} a_i a_{i+s} \ , s = 0, 1, \dots, n-1.$$

Definition 2. *Two sequences, $A = [a_1, \cdots, a_n]$ and $B = [b_1, \cdots, b_n]$, of length n are said to have zero NPAF, if $N_A(s) + N_B(s) = 0$ for $s = 1, \dots, n-1$.*

Definition 3. *Two sequences, $A = [a_1, \cdots, a_n]$ and $B = [b_1, \cdots, b_n]$, of length n with elements from $\{-1, +1\}$ are called Golay sequences, denoted by $GS(n)$, if they have zero NPAF, i.e. if $N_A(s) + N_B(s) = 0$ for $s = 1, \dots, n-1$.*

$GS(n)$ are known for lengths $n = 2$ and 10 ([9]), and for length 26 ([10]).

- $n = 2$: $[1, 1]$ and $[1, -1]$
- $n = 10$: $[1, -1, -1, 1, -1, 1, -1, -1, -1, 1]$ and $[1, -1, -1, -1, -1, -1, -1, 1, 1, -1]$
- $n = 26$: $[1, 1, 1, -1, -1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1, -1, -1]$ and $[-1, -1, -1, 1, 1, -1, -1, -1, 1, -1, 1, 1, -1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1, -1, -1]$

The basic properties of Golay sequences were studied by Golay in [9]. It is well known that $GS(n)$ exist for $n = 2^a \cdot 10^b \cdot 26^c$ where a, b, c are nonnegative integers. Infinitely many lengths of Golay sequences can be produced from a recursive construction due to Turyn [26]. Hence, an infinite family of Hadamard matrices can be obtained from Golay sequences (using Theorem 3 and Remark 2). In particular, the following family exists: $\{H(2n)$ is a Hadamard matrix of order $2n : \ n = 2^a \cdot 10^b \cdot 26^c, \ a, b, c$ nonnegative integers such as $GS(n)$ exist$\}$.

2.4 Hadamard Matrices Obtained from Other Multiplication Methods

A prolific method for constructing Hadamard matrices uses T-matrices or T-sequences. This method relies on Base sequences and Yang numbers as its main characteristics. We give the necessary definitions needed for establishing the theoretical background for multiplications theorem using T-sequences. For further details on these multiplication methods for T-sequences we refer the interested reader to [14,17].

Definition 4. *Four $(-1,1)$ sequences A, B, C, D of lengths $n+p, n+p, n, n$ are Base sequences, (abbreviated as $BS(n+p,n)$) if:*

1. $N_A(s)+N_B(s)+N_C(s)+N_D(s) = \begin{cases} 0, & s=1,\dots,n-1 \\ 4n+2p, & s=0 \end{cases}$
2. $N_A(s)+N_B(s)=0, s=n,\dots,n+p-1$

whereas with N_X we denote the non-periodic autocorrelation function of a sequence X.

Definition 5. *Four sequences $X=[x_1,\dots,x_n]$, $Y=[y_1,\dots,y_n]$, $Z=[z_1,\dots,z_n]$, $W=[w_1,\dots,w_n]$ of length n with entries $(-1,0,1)$ are called T-sequences, (abbreviated as $TS(n)$) if:*

1. $|x_i|+|y_i|+|z_i|+|w_i|=1,\ i=1,\dots,n.$
2. $N_X(s)+N_Y(s)+N_Z(s)+N_W(s) = \begin{cases} 0, & s=1,\dots,n-1 \\ n, & s=0 \end{cases}$

There is a close connection between Base and T-sequences as this can be seen from the following construction.

Theorem 4 (Seberry and Yamada [23]). *If A,B,C,D are $BS(n+1,n)$ then the sequences, $X=[\frac{1}{2}(A+B),0_n]$, $Y=[\frac{1}{2}(A-B),0_n]$, $Z=[0_{n+1},\frac{1}{2}(C+D)]$, $W=[0_{n+1},\frac{1}{2}(C-D)]$ are $TS(2n+1)$.*

Definition 6. *Four circulant matrices T_1, T_2, T_3, T_4 of order t with entries $(-1,0,1)$ are called T-matrices if:*

1. $T_i * T_j = 0, i \neq j$ *($*$ denotes the Hadamard product)*
2. $T_1T_1^T+T_2T_2^T+T_3T_3^T+T_4T_4^T=tI_t.$

We recall that T-sequences always yield T-matrices, since a T-sequence of length n can be used as the first row of a circulant matrix which results in a T-matrix of order n, but not conversely. If there is a multiplication method which uses suitable sequences of lengths $n+p, n+p, n, n$ to produce T-sequences of length $y(2n+p)$, then y is called a Yang number. These methods have been used to produce vast numbers of inequivalent Hadamard matrices of large orders in [15], [16]. It is well known that if there exist T-sequences of length t then there exists a Hadamard matrix of order $4t$.

Theorem 5 (Cooper and Wallis [5]). *Suppose there exist circulant T-matrices (or equivalent T-sequences) T_i, $i=1,\dots,4$ of order n. Then the matrices,*

$$\begin{aligned} A &= T_1+T_2+T_3+T_4 \\ B &= -T_1+T_2+T_3-T_4 \\ C &= -T_1-T_2+T_3+T_4 \\ D &= -T_1+T_2-T_3+T_4 \end{aligned}$$

can be used in the Goethals-Seidel array (see [8, page 107]) to obtain a Hadamard matrix of order $4n$.

2.5 List of Hadamard Matrices of Orders Up to 100

In the following Table we give a list of Hadamard matrices for orders up to 100, using only the construction methods given previously. Hence, these Hadamard matrices are easily constructed and thus can provide immediately the secret sharing schemes presented in this paper. For higher orders we refer to the respective Tables of [15] and [16]. Note that, for a permissible order of $H(n)$ there may be more than one construction methods available. We list only one of them for each case.

Table 1. Fast Construction Methods for $H(n)$, $4 \leq n \leq 100$

Order	Family	Construction	Order	Family	Construction
4	Sylvester	Lemma 1	8	$GS(4)$ exist	Theorem 3
12	Paley Type I	Theorem 2	16	$GS(8)$ exist	Theorem 3
20	$GS(10)$ exist	Theorem 3	24	Paley Type I	Theorem 2
28	Paley Type II	Theorem 2	32	Sylvester	Lemma 1
36	Paley Type II	Theorem 2	40	$GS(20)$ exist	Theorem 3
44	$TS(11)$ exist	Theorem 5	48	Paley Type I	Theorem 2
52	$GS(26)$ exist	Theorem 3	56	$H(2) \otimes H(28)$	Theorem 1
60	Paley Type II	Theorem 2	64	Sylvester	Lemma 1
68	Paley Type I	Theorem 2	72	Paley Type I	Theorem 2
76	Paley Type II	Theorem 2	80	$H(4) \otimes H(20)$	Theorem 1
84	Paley Type II	Theorem 2	88	$H(2) \otimes H(44)$	Theorem 1
92	$TS(23)$ exist	Theorem 5	96	$H(2) \otimes H(48)$	Theorem 1
100	$TS(25)$ exist	Theorem 5			

3 Hadamard 3-Designs and Secret-Sharing Schemes

Dougherty, Mesnager, and Solé [7] proposed the following secret-sharing scheme. A secret consisting of elements of F_q is split into its components. Let $s \in F_q$ be the secret we wish to share, and let G be a generator matrix for a code C of length n with columns $G_0, G_1, \ldots, G_{n-1}$. Let z be the information vector such that $zG_0 = s$, and $u = zG$. The corresponding coordinate u_i, $i = 1, 2, \ldots, n-1$, is assigned to each party. Assume that G_0 is a linear combination of the $n-1$ columns $G_1, \ldots, G_{n-1}$. The secret s is then determined by the set of shares $\{u_{i_1}, u_{i_2}, \ldots, u_{i_m}\}$, if and only if G_0 is a linear combination $G_0 = \sum_{j=1}^{m} x_j G_{i_j}$, where $1 \leq i_1 < \cdots < i_m \leq n-1$ and $m \leq n-1$. So by solving this linear equation, we find x_j and from then on the secret by $s = zG_0 = \sum_{j=1}^{m} x_j zG_{i_j} = \sum_{j=1}^{m} x_j u_{i_j}$. The set of m shares $\{u_{i_1}, u_{i_2}, \ldots, u_{i_m}\}$ determines the secret if and only if there is a codeword $(1, 0, ..., 0, c_{i_1}, 0, ..., 0, c_{i_m}, 0.., 0) \in C^{\perp}$, where $c_{i_j} \neq 0$ for at least one j [7] (see also [21] for descriptions of this technique). Let $\mathcal{P}$ be the set of parties involved in the secret sharing. In this case $\mathcal{P}$ is the set of coordinates except for the first one. The set Γ, called the **access structure** of the secret-sharing scheme, consists of subsets of $\mathcal{P}$ such that any element of Γ can uncover the secret.

In [4] the following two-part scheme is explained: Let the codewords of weight i of a given binary self-dual $[n, n/2]$ code C hold a $3-(v,k,\lambda)$ design D_i, where $v = n$ and $k = i$ (the codewords of weight i are the blocks of D_i). For the first part of the secret the distribution is the same as in the previous scheme. For the second part of the secret the first two participants should be removed, so only $n-3$ participants are involved in this part. The second part of the secret is $s' = s + zG_1 + zG_2 = z(G_0 + G_1 + G_2)$. Then s' can be determined by the set of shares $\{u_{i_3}, u_{i_4}, \dots, u_{i_m}\}$, if and only if $G_2 = G_0 + G_1 + \sum_{j=3}^{m} x_j G_{i_j}$ where $3 \le i_3 < \cdots < i_m \le n-1$ and $m \le n-1$. Hence s' can be determined by the set of shares $\{u_{i_3}, u_{i_4}, \dots, u_{i_m}\}$, if and only if there is a codeword $x \in C$ with $supp(x) = \{1, 2, 3, i_3, \dots, i_m\}$.

Briefly, the idea of the two-part scheme can be described as: there are two doors (one after another) and n users (any user has a part of the key) and small groups of users can unlock the "inner" door by combining their parts of the key. Just after that bigger groups of users can combine their parts of the key in order to unlock the "outer" door.

Here we propose the following scheme: Let $H(4m)$ be a Hadamard matrix of order $4m$ with all entries 1 in its first row. Using it, we construct a Hadamard $3-(4m, 2m, m-1)$ design $\mathcal{D}$ that has an orthogonality property. The scheme is based on the design structure and the orthogonality of any two rows. For the first part of the secret s, the access structure of this secret-sharing scheme is given by

$$\Gamma = \{A \mid A \text{ is the support of a block } B \in \mathcal{B} \text{ with } B_0 = 1\}. \tag{3}$$

We take the blocks that have 1 in the first position. It is easy to compute that there are $\frac{(v-1)(v-2)}{(k-1)(k-2)}\lambda$ such blocks where $v = 4m, k = 2m$, and $\lambda = m-1$ (then $\frac{(v-1)(v-2)}{(k-1)(k-2)}\lambda = 4m-1$). These blocks without the first point hold $2-(v-1, k-1, \lambda)$ design $\mathcal{D}'$. For the second part we take the $\frac{v-2}{k-2}\lambda$ $(= 2m-1)$ blocks of $\mathcal{D}'$ that have 1 in the first position. These blocks without the first point hold $1-(v-2, k-2, \lambda)$ design $\mathcal{D}''$. Then, for the second part of the secret, the access structure consists of λ groups of size $k-3$. To recover the two-part secret we should use the groups of size $k-3$ at first. They recover the second part of the secret. After that to recover the other part of the secret we use these groups (they are of size $k-2$ already) and the other $\frac{v-2}{k-2}\lambda - \lambda = \frac{v-k}{k-2}\lambda$ groups of size $k-2$. We add a new participant that has ones in these groups of size $k-2$ (the other values are 0). At last, we use the obtained $\frac{v-2}{k-2}\lambda$ groups of size $k-1$, and the other $\frac{(v-1)(v-2)}{(k-1)(k-2)}\lambda - \frac{v-2}{k-2}\lambda = \frac{(v-k)(v-2)}{(k-1)(k-2)}\lambda$ groups of the same size to recover the first part of the secret. Therefore, we obtain the following:

Theorem 6. *Let $H(4m)$ be a Hadamard matrix of order $4m$ in semi-normalized form. Then there exists a two-step secret-sharing scheme derived from a Hadamard $3-(4m, 2m, m-1)$ design with the following access structure:*

(i) $4m-1$ *groups of size* $2m-1$ *can recover the first part of the secret*
(ii) $m-1$ *groups of size* $2m-3$ *can recover the second part of the secret*

Recall from Section 2, that there exist infinite families of Hadamard matrices that arise from Golay sequences (in powers of $2, 10$ and 26), and of Sylvester (in powers of 2) or Paley Type (for prime numbers). Using this remark with the previous Theorem the following Corollary is immediate.

Corollary 1. *Let* $H(4m)$ *be a Hadamard matrix in semi-normalized form as above, constructed from Golay sequences, Paley or Sylvester Type, and their Kronecker product. Then there exists an infinite family of two-step secret-sharing schemes with access structure as in Theorem 6.*

Note that there are cases when the access structure for the second part of the scheme is not threshold (as in the given example in the next section). It can be useful for cases when some users (or special groups of users) have more privileges than the other users.

Example 1. Let $H(16)$ be a Hadamard matrix of order 16 (Sylvester Type or constructed from $GS(8)$). The corresponding Hadamard design is a $3-(16, 8, 3)$ design $\mathcal{D}$. For the first part of the secret s, the access structure of this secret-sharing scheme contains those 15 blocks of $\mathcal{D}$ that have 1 in the first position. Then, there are 15 groups of size 7. These blocks without the first point hold $2-(15, 7, 3)$ design $\mathcal{D}^{'}$. For the second part we take those 7 blocks of $\mathcal{D}^{'}$ that have 1 in the first position. These blocks without the first point hold $1-(14, 6, 3)$ design $\mathcal{D}^{''}$. Then, for the second part of the secret, the access structure consists of 3 groups of size 5. To recover the two-part secret we should use the groups of size 5 at first. They recover the second part of the secret. After that to recover the other part of the secret we use these groups (they are of size 6 already) and the other 4 groups of size 6. We add a new participant that has ones in these groups of size 6 (the other values are 0). At last, we use the obtained 15 groups of size 7, and the other 15 groups of the same size to recover the first part of the secret.

Example 2. Let $H(56)$ be a Hadamard matrix of order 56. The corresponding Hadamard design is a $3-(56, 28, 13)$ design $\mathcal{D}$. In the same way, for the first part of the secret s, the access structure of this secret-sharing scheme contains those 55 blocks of $\mathcal{D}$ that have 1 in the first position. Then, there are 55 groups of size 27. These blocks without the first point hold $2-(55, 27, 13)$ design $\mathcal{D}^{'}$. For the second part we take those 27 blocks of $\mathcal{D}^{'}$ that have 1 in the first position. These blocks without the first point hold $1-(54, 26, 13)$ design $\mathcal{D}^{''}$. Then, for the second part of the secret, the access structure consists of 13 groups of size 25. To recover the two-part secret we should use the groups of size 25 at first. They recover the second part of the secret. After that to recover the other part of the secret we use these groups (they are of size 26 already) and the other groups of size 26. We add a new participant that has ones in these groups of size 26 (the other values are 0). At last, we use the obtained 55 groups of size 27, and the other 55 groups of the same size to recover the first part of the secret.

4 Algorithmic Construction of Secret-Sharing Schemes from Hadamard Matrices

Finally, we present an algorithmic construction of secret-sharing schemes from Hadamard matrices, based on the multiplication methods presented in Section 2.4, via Base sequences. Note that $BS(n+1, n)$ exist for each $n = 1, \ldots, 35$, and is one of the most promising methods to show the Hadamard conjecture. For more details on this matter see [14]. All $BS(n+1, n)$ for $n = 1, \ldots, 35$ can be found in [19].

Algorithm 1

```
  function BaseSeqs2SecretSharingScheme($B1, B2, B3, B4$)
Require: $BS(n+1, n)$ exist
    $X, Y, Z, W \leftarrow$ BaseSeqs2TSeqs($B1, B2, B3, B4$)      ▷ $X, Y, Z, W$ are $TS(2n+1)$
    $H(4(2n+1)) \leftarrow$ TSeqs2Hadamard($X, Y, Z, W$)          ▷ Theorem 5 Construction
    $H(4m(2n+1)) \leftarrow H(4(2n+1)) \otimes H(m)$               ▷ Theorem 1 Construction
    Hadamard3Design($4m(2n+1), 2m(2n+1), 2mn+m-1$) $\leftarrow$
    Hadamard2HadamardDesign($H(4m(2n+1))$)
    return (SecretSharingScheme)
  end function
```

We illustrate the execution of Algorithm 1 for $n = 1$, thus visualizing our proposed method for secret-sharing schemes in a small example (in order to save space).

Input. $BS(2,1)$: $B_1 = [1,1]$, $B_2 = [1,-1]$, $B_3 = [1]$, $B_4 = [1]$
Step 1. $TS(3)$: $X = [1,0,0]$, $Y = [0,1,0]$, $Z = [0,0,1]$, $W = [0,0,0]$
Step 2. $H(24)$: We consider the Kronecker product of the produced $H(12)$ from $TS(3)$ by $H(2)$, i.e. $H(24) = H(12) \otimes H(2)$
Step 3. We normalize the derived Hadamard matrix of order 24, $H(24)$:

$$H(24) = \begin{pmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&-&1&-&1&1&1&-&-&-&1&-&1&-&1&-&1&1&1&-&-&-&1&-\\
1&-&-&1&-&1&1&1&-&-&-&1&1&-&-&1&-&1&1&1&-&-&-&1\\
1&1&-&-&1&-&1&1&1&-&-&-&1&1&-&-&1&-&1&1&1&-&-&-\\
1&-&1&-&-&1&-&1&1&1&-&-&1&-&1&-&-&1&-&1&1&1&-&-\\
1&-&-&1&-&-&1&-&1&1&1&-&1&-&-&1&-&-&1&-&1&1&1&-\\
1&-&-&-&1&-&-&1&-&1&1&1&1&-&-&-&1&-&-&1&-&1&1&1\\
1&1&-&-&-&1&-&-&1&-&1&1&1&1&-&-&-&1&-&-&1&-&1&1\\
1&1&1&-&-&-&1&-&-&1&-&1&1&1&1&-&-&-&1&-&-&1&-&1\\
1&1&1&1&-&-&-&1&-&-&1&-&1&1&1&1&-&-&-&1&-&-&1&-\\
1&-&1&1&1&-&-&-&1&-&-&1&1&-&1&1&1&-&-&-&1&-&-&1\\
1&1&-&1&1&1&-&-&-&1&-&-&1&1&-&1&1&1&-&-&-&1&-&-\\
1&1&1&1&1&1&1&1&1&1&1&1&-&-&-&-&-&-&-&-&-&-&-&-\\
1&-&1&-&1&1&1&-&-&-&1&-&-&1&-&1&-&-&-&1&1&1&-&1\\
1&-&-&1&-&1&1&1&-&-&-&1&-&1&1&-&1&-&-&-&1&1&1&-\\
1&1&-&-&1&-&1&1&1&-&-&-&-&-&1&1&-&1&-&-&-&1&1&1\\
1&-&1&-&-&1&-&1&1&1&-&-&-&1&-&1&1&-&1&-&-&-&1&1\\
1&-&-&1&-&-&1&-&1&1&1&-&-&1&1&-&1&1&-&1&-&-&-&1\\
1&-&-&-&1&-&-&1&-&1&1&1&-&1&1&1&-&1&1&-&1&-&-&-\\
1&1&-&-&-&1&-&-&1&-&1&1&-&-&1&1&1&-&1&1&-&1&-&-\\
1&1&1&-&-&-&1&-&-&1&-&1&-&-&-&1&1&1&-&1&1&-&1&-\\
1&1&1&1&-&-&-&1&-&-&1&-&-&-&-&-&1&1&1&-&1&1&-&1\\
1&-&1&1&1&-&-&-&1&-&-&1&-&1&-&-&-&1&1&1&-&1&1&-\\
1&1&-&1&1&1&-&-&-&1&-&-&-&-&1&-&-&-&1&1&1&-&1&1
\end{pmatrix}$$

Step 4. The incidence matrix of the corresponding Hadamard 3-design is:

$$D = \begin{pmatrix}
101011100010101011100010 \\
010100011101010100011101 \\
100101110001100101110001 \\
011010001110011010001110 \\
110010111000110010111000 \\
001101000111001101000111 \\
101001011100101001011100 \\
010110100011010110100011 \\
100100101110100100101110 \\
011011010001011011010001 \\
100010010111100010010111 \\
011101101000011101101000 \\
110001001011110001001011 \\
001110110100001110110100 \\
111000100101111000100101 \\
000111011010000111011010 \\
111100010010111100010010 \\
000011101101000011101101 \\
101110001001101110001001 \\
010001110110010001110110 \\
110111000100110111000100 \\
001000111011001000111011 \\
111111111111000000000000 \\
000000000000111111111111 \\
101011100010010100011101 \\
010100011101101011100010 \\
100101110001011010001110 \\
011010001110100101110001 \\
110010111000001101000111 \\
001101000111110010111000 \\
101001011100010110100011 \\
010110100011101001011100 \\
100100101110011011010001 \\
011011010001100100101110 \\
100010010111011101101000 \\
011101101000100010010111 \\
110001001011001110110100 \\
001110110100110001001011 \\
111000100101000111011010 \\
000111011010111000100101 \\
111100010010000011101101 \\
000011101101111100010010 \\
101110001001010001110110 \\
010001110110101110001001 \\
110111000100001000111011 \\
001000111011110111000100
\end{pmatrix}$$

Step 5. The matrix D_1 represents the key (the first coordinate) for the first part of the secret. The blocks of D_1 without this coordinate form the groups of size 11 and hold $2-(23,11,5)$ design D'. The matrix D_2 contains the blocks of D' with 1 in the first coordinate. Without their first coordinate, the blocks of D' hold $1-(22,10,5)$ design. The matrix D_3 represents the key (the third coordinate) for the second part of the secret. These two rows (without the coordinates in black) form the groups of size 9.

$$D_1 = \begin{pmatrix}
\mathbf{1}01011100010101011100010 \\
\mathbf{1}00101110001100101110001 \\
\mathbf{1}10010111000110010111000 \\
\mathbf{1}01001011100101001011100 \\
\mathbf{1}00100101110100100101110 \\
\mathbf{1}00010010111100010010111 \\
\mathbf{1}10001001011110001001011 \\
\mathbf{1}11000100101111000100101 \\
\mathbf{1}11100010010111100010010 \\
\mathbf{1}01110001001101110001001 \\
\mathbf{1}10111000100110111000100 \\
\mathbf{1}11111111111000000000000 \\
\mathbf{1}01011100010010100011101 \\
\mathbf{1}00101110001011010001110 \\
\mathbf{1}10010111000001101000111 \\
\mathbf{1}01001011100010110100011 \\
\mathbf{1}00100101110011011010001 \\
\mathbf{1}00010010111011101101000 \\
\mathbf{1}10001001011001110110100 \\
\mathbf{1}11000100101000111011010 \\
\mathbf{1}11100010010000011101101 \\
\mathbf{1}01110001001010001110110 \\
\mathbf{1}10111000100001000111011
\end{pmatrix},$$

$$D_2 = \begin{pmatrix}
\mathbf{11}0010111000110010111000 \\
\mathbf{11}0001001011110001001011 \\
\mathbf{11}1000100101111000100101 \\
\mathbf{11}1100010010111100010010 \\
\mathbf{11}0111000100110111000100 \\
\mathbf{11}1111111111000000000000 \\
\mathbf{11}0010111000001101000111 \\
\mathbf{11}0001001011001110110100 \\
\mathbf{11}1000100101000111011010 \\
\mathbf{11}1100010010000011101101 \\
\mathbf{11}0111000100001000111011
\end{pmatrix}, \quad D_3 = \begin{pmatrix}
\mathbf{111}000100101111000100101 \\
\mathbf{111}100010010111100010010 \\
\mathbf{111}111111111000000000000 \\
\mathbf{111}000100101000111011010 \\
\mathbf{111}100010010000011101101
\end{pmatrix}$$

In this example the second part of the scheme is not threshold. Here the set of users $\{3, 4, 5, 6, 7, 8, 9, 10, 11\}$ can uncover the secret but also it can be uncovered by subsets of users $\{3, 4, 6\}$, $\{5, 7, 9\}$, or $\{8, 10, 11\}$. It is useful when special groups of users have more privileges. For example, if the users are managers, accountants, and guards in a branch bank then a given safe can be unlocked by all users or by special group manager-accountant-guard.

5 Conclusion

In this paper, we have considered some connections between Hadamard 3-designs and secret-sharing schemes. This has been achieved by considering some constructions for Hadamard matrices and Hadamard 3-designs. In the sequel, we describe a two-step secret-sharing scheme based on Hadamard matrices in semi-normalized form and their corresponding 3-designs.

Acknowledgments

The research of the second author was financially supported by a scholarship awarded by the Secretariat of the Research Committee of National Technical University of Athens. The work of the third author was partially supported within the project BG 051PO001-3.3.04/13 of the HR Development OP of the European Social Fund 2007-2013 and the project RD642-01/2010 of the University of Veliko Tarnovo Science Fund.

References

1. Assmus Jr., E.F., Key, J.D.: Designs and their codes. Cambridge University Press, Great Britain (1992)
2. Blakley, G.R.: Safeguarding cryptographic keys. In: Proc. of the 1979 AFIPS National Computer Conference, pp. 313–317 (1979)
3. Bosma, W., Cannon, J.: Handbook of Magma Functions, Version 2.9, Sydney (2002)
4. Bouyuklieva, S., Varbanov, Z.: Some connections between self-dual codes, combinatorial designs and secret-sharing schemes. Adv. Math. of Communications (to appear)
5. Cooper, J., Wallis, J.S.: A Construction for Hadamard Arrays, Bull. Austral. Math. Soc. 7, 269–278 (1972)
6. Craigen, R.: Hadamard Matrices and Designs. In: Colbourn, C.J., Dinitz, J.H. (eds.) The CRC Handbook of Combinatorial Designs, pp. 370–377. CRC Press, Boca Raton (1996)
7. Dougherty, S.T., Mesnager, S., Solé, P.: Secret-sharing schemes based on self-dual codes. In: Information Theory Workshop, Porto, May 5-9, pp. 338–342 (2008)
8. Geramita, A.V., Seberry, J.: Orthogonal Designs. In: Quadratic Forms and Hadamard Matrices. Lecture Notes in Pure and Applied Mathematics, vol. 45, Marcel Dekker, Inc., New York (1979)
9. Golay, M.J.E.: Complementary sequences. IRE Transactions on Information Theory 7, 82–87 (1961)
10. Golay, M. J. E.: Note on Complementary series. Proc. IRE (50), 84 (1962)
11. Hadamard, J.: Resolution d'une question relative aux determinants. Bull. des. Sci. Math. 17, 240–246 (1893)
12. Horadam, K.J.: Hadamard matrices and their applications. Princeton University Press, Princeton (2007)
13. Hughes, D.R., Piper, F.C.: Design theory. Cambridge Univ. Press, Cambridge (1985)
14. Kharaghani, H., Koukouvinos, C.: Complementary, Base and Turyn Sequences. In: Colbourn, C.J., Dinitz, J.H. (eds.) Handbook of Combinatorial Designs, 2nd edn., pp. 317–321. Chapman and Hall/CRC Press, Boca Raton, Fla (2006)
15. Kotsireas, I.S., Koukouvinos, C., Simos, D.E.: Inequivalent Hadamard matrices from base sequences. Util. Math. 78, 3–9 (2009)
16. Kotsireas, I.S., Koukouvinos, C., Simos, D.E.: Inequivalent Hadamard matrices from near normal sequences. J. Combin. Math. Combin. Comput. 75, 105–115 (2010)
17. Koukouvinos, C.: Sequences with Zero Autocorrelation. In: Colbourn, C.J., Dinitz, J.H. (eds.) The CRC Handbook of Combinatorial Designs, pp. 452–456. CRC Press, Boca Raton (1996)
18. Koukouvinos, C.: Undecided cases for D-optimal designs of order $n \equiv 0 \pmod 4$, http://www.math.ntua.gr/~ckoukouv
19. Koukouvinos, C.: Base sequences $BS(n+1, n)$, http://www.math.ntua.gr/~ckoukouv

20. Koukouvinos, C., Seberry, J.: New weighing matrices and orthogonal designs constructed using two sequences with zero autocorrelation function - a review. J. Statist. Plann. Inference 81, 153–182 (1999)
21. Massey, J.L.: Some applications of coding theory in cryptography. In: Farrell, P.G. (ed.) Codes and Ciphers, Cryptography and Coding IV, pp. 33–47. Formara Lt, Esses (1995)
22. Paley, R.E.A.C.: On orthogonal matrices. Journal of Mathematics and Physics 12, 311–320 (1933)
23. Seberry, J., Yamada, M.: Hadamard Matrices, Sequences and Block Designs. In: Dinitz, J.H., Stinson, D.R. (eds.) Contemporary Design Theory: A Collection of Surveys, pp. 431–560. John Wiley & Sons, New York (1992)
24. Shamir, A.: How to share a secret. Communications of the ACM 22, 612–613 (1979)
25. Sylvester, J.J.: Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tesselated pavements in two or more colours, with applications to Newton's rule, ornamental tilework, and the theory of numbers. Phil. Mag. 34, 461–475 (1867)
26. Turyn, R.J.: Hadamard matrices, Baumert-Hall units, four-symbol sequences, pulse compression, and surface wave encoding. J. Combin. Theory A 16, 313–333 (1974)
27. Yang, C.H.: On hadamard matrices constructible by two circulant submatrices. Math. Comp. 25, 181–186 (1971)
28. Wallis, W.D., Street, A.P.: Combinatorics: Room Squares, Sum-Free Sets, Hadamard matrices. Lecture Notes in Mathematics, vol. 292. Springer, Heidelberg (1972)

I-RiSC: An SMT-Compliant Solver for the Existential Fragment of Real Algebra

Ulrich Loup and Erika Ábrahám

RWTH Aachen University, Germany
{loup,abraham}@cs.rwth-aachen.de

Abstract. This paper connects research in computer science in the field of SAT-modulo-theories (SMT) solving and research in mathematics on decision procedures for real algebra. We consider a real algebraic decision procedure computing all realizable sign conditions of a set of polynomials. We modify this procedure so that it satisfies certain requirements needed for the embedding into an SMT-solver.

Keywords: SMT Solving, Real Algebra, I-RiSC, FO Logic, DPLL(T).

1 Introduction

Though the propositional satisfiability problem (SAT), where the variables range over the values 1 (true) and 0 (false), is NP-complete, SAT-solvers are quite efficient in practice due to a vast progress in SAT-solving over the last years. In particular, the DPLL algorithm [12] and its recent improvements such as clause learning or sophisticated decision heuristics made SAT-solving highly efficient for practical problems, what led to a break-through of SAT-solving also in industry.

SAT-modulo-theories (SMT) solving aims at embedding decision procedures for various first-order theories into the SAT-solving context [1,15] This combination yields highly efficient solvers, which are frequently applied, for example, in the formal analysis, verification, and synthesis of systems, even over a continuous domain. For the domain of the real numbers, research in the area of SMT solving concentrates on linear real arithmetic. Prominent examples of SMT-solvers for this logic are `Z3` [14], `Yices` [9], `MathSAT` [7] and `OpenSMT` [6]. Less emphasis is put on *real algebra*, the first-order logic with addition and multiplication over the reals, which we address in this paper. However, there is a growing interest in SMT-solving for real algebra. This drift is reflected, for instance, by 2010's SMT-competition [16], where the non-linear real arithmetic (NRA) division was introduced for the first time. The few existing SMT-solvers supporting non-linear real algebraic constraints are incomplete. For example, `Z3`, `CVC3` [2], `MiniSMT` [18] and `ABsolver` [4] can handle only fragments of real algebra, whereas the solver `iSAT` [10], based on interval arithmetic, allows even trigonometrical expressions but may terminate with the answer "unknown".

Several decision procedures were developed for real algebra since the 1940s, which are currently operational in some computer algebra systems. The most well-known approaches are the CAD method [8] and Gröbner bases computations. The textbook [3] comprises the state of the art for algorithms in real algebraic geometry. This book is also available online at

http://perso.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted2.pdf

However, the employment of these procedures in an SMT-solver is not straightforward, because SMT-solvers impose some requirements on the embedded decision procedures for the approach to be feasible in practice [1, 26.4.1]:

– First of all, for efficient SMT-solving we need decision procedures that work *incrementally*. That is, after the consistency check of a set of real algebraic constraints the procedure should be able to extend the set by adding new constraints, and reuse the previous computations for the check of the extended set.
– For an unsatisfiable set of constraints the decision procedure should be able to determine a *minimal infeasible subset*, i.e., an unsatisfiable subset which is minimal in the sense that removing any constraint makes it satisfiable.
– The ability to *backtrack* should allow to remove previously added constraints.

Unfortunately, current decision procedures for real algebra do not support the above functionalities. In this paper we describe how a method from [3], based on computing realizable sign conditions, can be modified to satisfy the requirements for SMT-solving. We call the modified new method I-RiSC (Incremental Realization of Sign Conditions).

2 Preliminaries

2.1 SMT-Solving

DPLL-based decision procedures [12] are also applicable to logics richer than propositional logic, by abstracting all non-propositional atomic constraints by propositional variables. This approach is called lazy *SAT-modulo-theories* (*SMT*) solving (cf. Fig. 1).

Full lazy (*off-line*) SMT-solvers first create a Boolean skeleton of the input formula, replacing all theory constraints by fresh Boolean variables. The resulting Boolean formula is passed to a SAT-solver, which searches for a satisfying assignment. If it does not succeed, the formula is unsatisfiable. Otherwise, the assignment found corresponds to certain truth values for the theory constraints and has to be verified by the theory solver. If the constraints are satisfiable, then the original formula is satisfiable. Otherwise, if the theory solver detects that the conjunction of the corresponding theory constraints is unsatisfiable, it then hands over a reason for the unsatisfiability, a *minimal infeasible subset* of the theory constraints, to the SAT-solver. The SAT-solver uses this piece of information to exclude the detected conflict from further search. Afterwards,

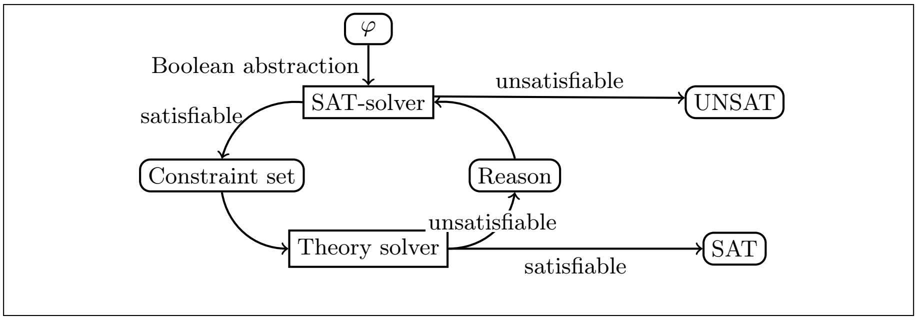


Fig. 1. The basic scheme of DPLL(T)-based SMT-solving

the SAT-solver computes again an assignment for the refined Boolean problem, which in turn has to be verified by the theory solver. Continuing this iteration in the end decides the satisfiability of the input formula.

Such full lazy check is often disadvantageous, since the SAT-solver may do a lot of needless work by extending an already (in the theory domain) contradictory partial assignment. *Less lazy* (*on-line*) DPLL(T) variants of the procedure call the theory solver more often, handing over constraints corresponding to partial assignments. To do so efficiently, the theory solver should accept constraints in an *incremental* fashion, where computation results of previous steps can be reused. In case of a conflict the theory solver should also be able to *backtrack*, i.e., remove the last asserted constraints.

Note that we strictly separate the satisfiability checks in the Boolean and in the theory domains, that means, we do not consider theory propagation embedded in the DPLL search like, e.g., Yices [9] does.

2.2 Real Algebra

In SMT-solving we consider only a fragment of real algebra, containing existentially quantified conjunctions of real algebraic constraints c, which compare *polynomials* p to zero:

$$\begin{array}{lcccccc} p & ::= & 0 & \mid\; 1 & \mid\; x & \mid\; (p+p) & \mid\; (p \cdot p) \\ c & ::= & p = 0 & \mid\; p < 0 & \mid\; p > 0 & & \end{array}$$

The operators $+$ and $\cdot$ have the standard semantics of addition and multiplication. We stick to a more algebraic point of view and refer to [13] or [3] for basic notions on real algebra.

Let $n \in \mathbb{N}$ with $n \geq 1$ and $\mathbb{Z}[x_1, \ldots, x_n]$ be the set of all polynomials in the real-valued variables $x_1, \ldots, x_n$, called the polynomial ring of multivariate polynomials with integer coefficients. For a polynomial $p \in \mathbb{Z}[x_1, \ldots, x_n]$ the x_i-*degree of* p, written $\deg_{x_i}(p)$, is the highest exponent at x_i in p, and $[x_i^d]p$ denotes the *coefficient of* x_i^d *in* p, which is a polynomial in $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. A polynomial p is called a *monomial* if $p = \prod_{i=1}^{n} x_i^{e_i}$ with $e_i \in \mathbb{N}$; and in this case we define its degree as $\deg(p) := \sum_{i=1}^{n} e_i$. For arbitrary multivariate polynomials p we set $\deg(p) := \max\{\deg(q) \mid q$ monomial of $p\}$. The power set of S is denoted by 2^S, and $\binom{S}{k} := \{S' \in 2^S \mid |S'| = k\}$.

A *system of m real algebraic constraints* is a sequence $p_1 \sim_1 0, \ldots, p_m \sim_m 0$ of real algebraic constraints with $m \geq 1$ and $p_i \in \mathbb{Z}[x_1, \ldots, x_n]$, $\sim_i \in \{=, <, >\}$ for $1 \leq i \leq m$. We define the *signs* $\sigma = (\sigma_1, \ldots, \sigma_m) \in \{-1, 0, 1\}^m$ of the polynomials $P = (p_1, \ldots, p_m)$ by $\operatorname{sgn}(p_i) = \sigma_i$ for all $1 \leq i \leq m$. We call (P, σ) a *sign condition on P*. The set

$$\Re eali_\sigma(P) = \{(a_1, \ldots, a_n) \in \mathbb{R}^n \mid \operatorname{sgn}(p_i(a_1, \ldots, a_n)) = \sigma_i, 1 \leq i \leq m\}$$

of real solutions to the sign condition (P, σ) is called the *realization of* (P, σ). Note that $\Re eali_\sigma(P)$ is also a special *semi-algebraic set*. A sign condition (P, σ) is called *realizable* if $\Re eali_\sigma(P) \neq \emptyset$ (cf. [3, Def. 2.25]). The sign condition $(P, \{0\}^m)$ is said to be *algebraic*.

The satisfiability problem for conjunctions of real algebraic constraints, which we address in this paper, can be formulated in an algebraic context as follows:

Problem 1. Satisfiability problem of real algebraic conjunctions

Input:	$n, m \in \mathbb{N}$, $m, n \geq 1$, $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$, $\sigma \in \{-1, 0, 1\}^m$.
Problem:	Determine whether $\Re eali_\sigma(P) = \emptyset$ and compute an $a \in \Re eali_\sigma(P)$ in case the set is not empty.

A realization of a sign condition can be composed of several sign-invariant subsets which do not have to be connected among each other. The connected sign-invariant sets are defined as follows.

Definition 1 (Sign-invariance, region)
Let $n, m \in \mathbb{N}$, $m, n \geq 1$, $R \subseteq \mathbb{R}^n$, $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$, and $\sigma \in \{-1, 0, 1\}^m$.

- *R is said to be* σ-invariant over P *if $R \subseteq \Re eali_\sigma(P)$.*
- *R is said to be* sign-invariant over P *or* P-sign invariant *if there is a $\sigma \in \{-1, 0, 1\}^m$ such that $R \subseteq \Re eali_\sigma(P)$.*
- *R is called a* region *if $R \neq \emptyset$ and R is connected, i.e., $R \neq (A \cap R) \cup (B \cap R)$ with $A \cap R \neq \emptyset$ and $B \cap R \neq \emptyset$ for any open, nonempty sets $A, B \subseteq \mathbb{R}^n$ with $A \cap B = \emptyset$.*

Considering the real line, the maximal sign-invariant regions have a simple structure: The P-sign invariant regions are confined by the roots and intersection points of the polynomials of P. This results in a decomposition of $\mathbb{R}$ into these points and the open intervals between the points. All points which can be solutions to a sign condition are called samples.

Definition 2 (Sample)
A set $S \subseteq \mathbb{R}^n$ is called set of samples for the sign condition (P, σ) *if $S \cap R \neq \emptyset$ for each σ-invariant region $R \subseteq \Re eali_\sigma(P)$.*

A natural approach to determine a real-valued point which satisfies a given semi-algebraic system of polynomials is to construct sample points for each maximal sign-invariant region and to search in the set of sample points for a solution point. The crucial part is the construction of the sample points. The CAD-method (partial-CAD), for example, tackles this problem by iteratively projecting the input set of polynomials until only univariate polynomials are left. Their roots, computed as algebraic numbers, and all points between the roots as well as one point below the smallest and one point above the largest root are the possible values for the current component of the sample points for the sign-invariant regions in a CAD.

If the set of samples for a given sign condition (P, σ) is empty, the sign condition is not realizable. In terms of SMT-solving, (P, σ) is then called an infeasible subset, which the SAT-solver gets back as a reason of unsatisfiability. We now define the notion of minimal infeasible subset precisely.

Definition 3 (Minimal infeasible subset)
Let $n, m \in \mathbb{N}$, $m, n \geq 1$, $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$ and $\sigma \in \{-1, 0, 1\}^m$. The sign condition

$$((P_{i_1}, \sigma_{i_1}), \ldots, (P_{i_k}, \sigma_{i_k})), \qquad \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, m\}, 1 \leq k \leq m$$

is called infeasible subset *if $((P_{i_1}, \sigma_{i_1}), \ldots, (P_{i_k}, \sigma_{i_k}))$ is not realizable. If in this case*

$$((P_{j_1}, \sigma_{j_1}), \ldots, (P_{j_{k-1}}, \sigma_{j_{k-1}}))$$

is realizable for all $\{j_1, \ldots, j_{k-1}\} \subsetneqq \{i_1, \ldots, i_k\}$ with $\{j_1, \ldots, j_{k-1}\} \neq \emptyset$, then $((P_{j_1}, \sigma_{j_1}), \ldots, (P_{j_k}, \sigma_{j_k}))$ is said to be minimal.

For the explanation of a recent approach for the sample construction in Section 3, we introduce algebraic computations with infinitesimal values. These are needed for several transformations of the input polynomials in order to obtain a set of polynomials whose realizable sign conditions can be computed much more efficiently than with the CAD-method.

Puiseux series. Since the abstraction to general fields is not needed in this paper, we introduce the notion of Puiseux series over the real field $\mathbb{R}$. Let ε be an infinitesimal element and $j, k \in \mathbb{Z}$ with $j > 0$. We call a series

$$a = \sum_{i \in \mathbb{Z}, i \geq k} a_i \varepsilon^{\frac{i}{j}}$$

with $a_i \in \mathbb{R}$ a *Puiseux series in ε*. Puiseux series are Laurent series in $\varepsilon^{\frac{1}{j}}$. If, for example, $j = 1$ and $k = 0$ then the Puiseux series a is nothing else than a Taylor series in ε. The additional parameters in a Puiseux series allow for the representation of elements *algebraic* over the *field of rational functions* $\mathbb{R}(\varepsilon)$, i.e. the representation of roots of polynomials in $\mathbb{R}(\varepsilon)[x]$. We denote the field of algebraic Puiseux series $\mathbb{R}\langle\varepsilon\rangle$. Given several infinitesimal elements $\varepsilon_1 < \ldots < \varepsilon_l$, we identify $\mathbb{R}\langle\varepsilon_1, \ldots, \varepsilon_l\rangle$ with $\mathbb{R}\langle\varepsilon_1\rangle \cdots \langle\varepsilon_l\rangle$. For $a \in \mathbb{R}\langle\varepsilon_1, \ldots, \varepsilon_l\rangle$ we denote $\lim_{\varepsilon_1, \ldots, \varepsilon_l} a = [\varepsilon_1^0 \cdots \varepsilon_l^0]a$, i.e. the part of a which is constant in $\varepsilon_1, \ldots, \varepsilon_l$.

3 Computing Realizable Sign Conditions

Let $m, n \in \mathbb{N}$ with $n \geq 1$ and $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$ throughout this section.

We describe a method from [3] for determining all realizable sign conditions over P by computing samples for them. After presenting a first approach to solve this problem, in a second step we give an improved variant utilizing some state-of-the-art optimizations. Apart from these, both algorithms have a similar structure: They compute all or possibly a bounded number of subsets of the possibly modified input polynomials, and for each of these subsets they construct the samples for a specific algebraic sign condition. The union of all of these samples builds the samples for the original sign condition. The different subsets of polynomials need to be considered in order to transform strict sign conditions to just equations (see [3, Prop. 13.1]).

In this paper, we solely concentrate on the search structure of the procedure. Therefore, we decouple the subset computation from the construction of samples. Moreover, we make the sample construction a black box represented by the subprocedure `samples(`Q`,`$(x_1 \ldots x_n)$`)` where Q is a tuple of $k \geq 1$ polynomials with coefficients possibly using infinitesimal elements δ and γ.

The method `samples(`Q`,`$(x_1 \ldots x_n)$`)` combines [3, Alg. 12.17] and [3, Alg. 13.2]. For readers familiar with real algebra, we give an intuitive description of how `samples(`Q`,`$(x_1 \ldots x_n)$`)` works in Table 1. We use the term `samples(`Q`,`$(x_1 \ldots x_n)$`)` to refer to both, the method and its output. `samples(`Q`,` $(x_1, \ldots, x_n)$`)` has the time complexity $d^{\mathcal{O}(k)}$ where $d = \max\{\deg(Q_i) \mid 1 \leq i \leq k\}$ (cf. [3, p. 512]).

Listing 1 shows the first variant of the algorithm for determining all realizable sign conditions over P suggested in [3, Rem. 13.3], combined with [3, Alg. 13.2].

Notation for lists. Empty lists are denoted by () and list concatenations by $\oplus$. We use L_i to refer to the ith element of a list L; the same notation is used for tuples.

Listing 1. First algorithm for computing realizable sign conditions

```
1  Input: $m, n \in \mathbb{N}$, $n \geq 1$, $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$
2  Output: samples $S \subseteq \mathbb{R}^n$ for every sign condition $(P, \sigma)$
3
4  $S$ := $\emptyset$;
5  for $1 \leq i \leq m$:
6    for $\{j_1, \ldots, j_i\} \in \binom{\{1,\ldots,m\}}{i}$:
7      $S$ := $S \cup$ samples(($P_{j_1}, \ldots, P_{j_i}$), ($x_1, \ldots, x_n$));
8  return $S$;
```

The correctness of this algorithm follows from [3, Prop. 13.2].

Taking the complexity of the samples computation into account, the number of steps performed by Listing 1 is $2^m d^{\mathcal{O}(k)}$, because lines 5 and 6 define a search through every subset of the given m polynomials.

An optimized version of Listing 1 needs $m^{n+1} d^{\mathcal{O}(k)}$ steps to compute all realizable sign conditions. This method is given by [3, Alg. 13.1] combined with [3, Alg.

Table 1. Description of the method `samples(`Q`,`$(x_1 \dots x_n)$`)`

`Input:` $k, n \in \mathbb{N}$, $k, n \geq 1$, $Q \in \mathcal{R}[x_1, \dots, x_n]^k$ with $\mathcal{R} \in \{\mathbb{Z}, \mathbb{Z}[\delta, \gamma]\}$
`Output:` set of samples in $\mathbb{R}^k$ for $(Q_1^2 + \cdots + Q_k^2 + (\varepsilon(x_1, \dots, x_n, y) - 1)^2, \{0\}^k)$

(1) Define $q := Q_1^2 + \cdots + Q_k^2 + (\varepsilon(x_1, \dots, x_n, y) - 1)^2$. In Listing 1, the polynomials $Q_1, \dots, Q_k$ are a selection of the input polynomials $P_1, \dots, P_m$. However in the improved Listing 2, $Q_1, \dots, Q_k$ comprise a subset of perturbed versions of the input polynomials. We will give some more details on the perturbation below.

Squaring the polynomials $Q_1, \dots, Q_k$ and adding the term $(\varepsilon(x_1, \dots, x_n, y) - 1)^2$ applies a perturbation of the given set of polynomials. The common roots of the transformed set of polynomials are bounded. This is achieved by intersecting the cylinders based on the extension of sign-invariant regions of Q to $\mathbb{R}\langle\varepsilon\rangle$ with the k-dimensional sphere with center 0 and radius $\frac{1}{\varepsilon}$, as given by $(\varepsilon(x_1, \dots, x_n, y) - 1)^2$ and an appropriate projection eliminating y (see [3, 12.6]).

(2) Generate, based on q, a *special Gröbner basis* G containing n elements, by applying additional perturbations to q utilizing a fresh infinitesimal element ζ (see [3, Not. 12.46]). The special structure of this Gröbner basis assures a finite number of common roots with multiplicity 1 of the polynomials in G (see [3, Lem. 12.45]). In particular, the remainders modulo the ideal generated by G can be computed using a finite multiplication table; and, in addition to it, each of them represents exactly one common root of G.

(3) Apply [3, Alg. 12.9] on the input G to obtain the finite, special multiplication tables mentioned in the previous step.

(4) Apply [3, Alg. 12.14] (performing $\lim_\zeta$ or $\lim_{\gamma,\zeta}$) using the special multiplication tables to compute *univariate representations* (see [3, p. 465]) for the roots of q. Note that this representation can still contain infinitesimal elements.

(5) Apply [3, Alg. 11.20] (performing $\lim_\varepsilon$ or $\lim_{\delta,\varepsilon}$) to remove the remaining infinitesimal elements. Multiplication with the main denominator results in univariate representations for the roots of q in $\mathbb{R}$.

13.2]. We now describe some details on this optimization and give the improved algorithm in Listing 2 by making use of the black box `samples(`Q`, `$(x_1, \dots, x_n)$`)` as introduced above.

Let ε, δ, γ be infinitesimal elements with $\varepsilon > \delta > \gamma > 0$.

Definition 4 (Perturbed general position polynomial) *Let $d, i, n \in \mathbb{N}$ with $n \geq 1$ and $1 \leq i \leq n$, $p \in \mathbb{Z}[x_1, \dots, x_n]$, and $\mu \in \{-1, 1, -\gamma, \gamma\}$ be a perturbation value, then*

$$\mathrm{PG}^{\mu}_{n,i,d}(p) := (1 - \delta)p + \delta\mu(1 + \sum_{1 \leq j \leq n} i^j x_j^d)$$

denotes the perturbed general position polynomial *of p w.r.t. n, i, d, and μ.*

This perturbation of P enables that not more than n polynomials need to be combined in order to compute the realizable sign conditions over P. More precisely, let $d = \max\{\deg(p_i) \mid 1 \leq i \leq m\}$ and for $1 \leq i \leq m$

$$\Gamma_i := \{\mathrm{PG}^{1}_{n,i,d}(p_i), \mathrm{PG}^{-1}_{n,i,d}(p_i), \mathrm{PG}^{\gamma}_{n,i,d}(p_i), \mathrm{PG}^{-\gamma}_{n,i,d}(p_i)\},$$

then any n polynomials from different Γ_i have at most a finite number of common roots; in particular, no $n+1$ polynomials from different Γ_i have a common root (see [3, Prop. 13.6]).

Note that there are only $\sum_{j=0}^{n} \binom{m}{j} 4^j = m^{n+1}$ combinations to consider, in contrast to 2^m of the first approach.

Listing 2. Improved algorithm for computing realizable sign conditions

```
1  Input: $m, n \in \mathbb{N}$, $n \geq 1$, $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$
2  Output: samples $S \subseteq \mathbb{R}^n$ for every sign condition $(P, \sigma)$
3
4  $S$ := $\emptyset$;
5  $\Gamma$ := $()$;
6  for $1 \leq i \leq m$:
7    $\Gamma$ := $\Gamma \oplus (\emptyset)$;
8  $d$ := $\max\{\deg(p_i) \mid 1 \leq i \leq m\}$;
9  for $1 \leq i \leq m$:
10   $\Gamma_i$ := $\{\mathrm{PG}^{1}_{n,i,d}(p_i), \mathrm{PG}^{-1}_{n,i,d}(p_i), \mathrm{PG}^{\gamma}_{n,i,d}(p_i), \mathrm{PG}^{-\gamma}_{n,i,d}(p_i)\}$;
11 for $1 \leq i \leq n$:
12   for $\{j_1, \ldots, j_i\} \in \binom{\{1,\ldots,m\}}{i}$:
13     for $(p_{j_1}, \ldots, p_{j_i}) \in \Gamma_{j_1} \times \cdots \times \Gamma_{j_i}$:
14       $S$ := $S$ $\cup$ samples($(p_{j_1}, \ldots, p_{j_i})$, $(x_1, \ldots, x_n)$);
15 return $S$;
```

Observe that Listing 2 is similar to Listing 1, only the perturbation performed in lines 8 to 10 and the additional loop over the combinations of perturbed general position polynomials in line 13 are new.

The correctness of the improved algorithm follows from the correctness of the algorithm presented in 1 as well as [3, Cor. 13.8], which states that the sample computation also works for the perturbed general position polynomials.

Both algorithms can be implemented using only a polynomial amount of space [3, Rem. 13.10].

4 The I-RiSC Solver

In this section, we present an SMT-compliant solver for the satisfiability problem of real algebraic conjunctions, which especially supports adding sign conditions incrementally, providing minimal infeasible subsets as reason for unsatisfiability or an assignment as proof of satisfiability, and the possibility to backtrack the search by a given number of steps. We call this solver I-RiSC referring to Incremental computation of Realizable Sign Conditions.

We pursue a class-based design of I-RiSC as it will be implemented in `C++` using `GiNaCRA`, which is a library providing real algebraic data structures [11].

4.1 Data Structure

Let $m, n \in \mathbb{N}$ with $n \geq 1$, $P \in \mathbb{Z}[x_1, \ldots, x_n]^m$, and $\sigma \in \{-1, 0, 1\}^m$. When adding a sign condition, the I-RiSC solver first chooses at most n of the sets of perturbed general position polynomials $\Gamma_1, \ldots, \Gamma_m$. The search tree for these selections can be represented as a binary tree of depth m where each inner node characterizes one decision whether to take a specific Γ_k or not. The left edge of an inner node labeled by 0 indicates that a specific Γ_k was not selected, the right edge labeled by 1 indicates that it was. The leaves of the tree are labeled with the bit words $w \in \{0,1\}^m$ representing the path to the leaf, as shown in Fig. 2 for $m = 2$. A $w \in \{0,1\}^m$ induces a choice of polynomials, denoted by

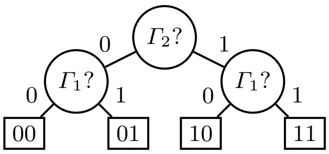


Fig. 2. I-RiSC search tree for two sets of polynomials Γ_1 and Γ_2

$P_w = (P_i \mid w_i = 1)$, a choice of sets of perturbed general position polynomials, denoted by $\Gamma_w = (\Gamma_i \mid w_i = 1)$, and a choice of corresponding sign conditions, denoted by $\sigma_w = (\sigma_i \mid w_i = 1)$. The samples are generated only at the leafs of the search tree. The mapping $\alpha : \{0,1\}^m \to 2^{\mathbb{R}^n}$ assigns a set of samples $\alpha(w)$ to each leaf $w \in \{0,1\}^m$. We use the notation $|w|_1$ for the number of 1s in $w \in \{0,1\}^m$; the empty word is denoted by ϵ. In order to traverse the search tree we use an order $\prec$ on bit words, defined as

$$w_1 \prec w_2 :\Leftrightarrow |w_1|_1 < |w_2|_1 \text{ or } (|w_1|_1 = |w_2|_1 \text{ and } w_1 <_{\text{lex}} w_2)$$

where $w_1 <_{\text{lex}} w_2$ compares the words lexicographically. For example, $00 <_{\text{lex}} 01 <_{\text{lex}} 10 <_{\text{lex}} 11$. Note that $\prec$ defines a strict total ordering on $\{0,1\}^m$. This enables the definition of the $\text{next}_\prec$ operator providing the next element $\text{next}_\prec(w)$ of w. Moreover, we write $\prod(Q_1, \ldots, Q_k)$ for the product $Q_1 \times \cdots \times Q_k$ of the sets $Q_1, \ldots, Q_k$.

We now define a notion useful for the analysis of the I-RiSC methods, especially the method for adding a new sign condition.

Definition 5 (Admissible and inadmissible leaves)
Let $w \in \{0,1\}^m$ be a leaf of the I-RiSC search tree. Let

$$S(w) := \{a \in \texttt{samples}(P_w, \sigma_w) \mid a \text{ is a sample for } (P_w, \sigma_w)\},$$

then w is called admissible *if $S(w) \neq \emptyset$, and* inadmissible *if $S(w) = \emptyset$.*

If a leaf $w \in \{0,1\}^m$ is admissible, then the set $S(w)$ can be reused in a later stage of the search. If, for example, the search is backtracked and the last i sign conditions, which did not contribute to the sample construction in w, are deleted, then $S(w) = S(w_{m-i+1} \ldots w_m)$. Thus, we do not need to recompute the samples for $w_{m-i+1} \ldots w_m$.

In the same fashion, we can reuse the result that $S(w)$ is empty. In particular, $S(w) = \emptyset$ entails that (P_w, σ_w) is an infeasible subset.

Lemma 1 (Minimal infeasible subset characterization)
Let $w \in \{0,1\}^m$ be a leaf of the I-RiSC search tree.
Then (P_w, σ_w) is a minimal infeasible subset iff w is inadmissible and for all $v \in \{0,1\}^m$ with $0 < |v|_1 < |w|_1$ it holds that v is admissible.

Proof. By Definition 3, (P_w, σ_w) is a minimal infeasible subset iff (P_v, σ_v) is realizable where $v = v_1 \ldots v_{|w|}$ with $v_i \in \{w_i, 0\}$ for $1 \leq i \leq |w|$ such that not all v_i are 0 and not all v_i are w_i. Because of Definition 5, this is equivalent to v being admissible with $0 < |v|_1 < |w|_1$. $\square$

Note that if all leaves of the I-RiSC search tree are traversed in order of $\prec$, then the first leaf w which proves to be inadmissible represents a minimal infeasible subset (P_w, σ_w), because only leaves with at least $|w|_1$ 1s would follow. The method `IRiSC.add` proceeds in this manner.

4.2 Class Design

In the following listing we specify the class containing the I-RiSC data structure and methods for the interaction with the DPLL-based SAT-solver.

Table 2 contains annotations to the variables defined in Listing 3. Note that the type of the variables sometimes depends on m or n, which are instantiated at runtime. But this does not affect the realizability of the class in `C++` because the type can be implemented by means of dynamic data structures.

The methods in Listing 3 are described in Table 3.

4.3 Methods

The methods `IRiSC.add` and `IRiSC.backjump` are presented in Listings 4 and 5.

`IRiSC.add`. In lines 3 to 12, the new sign condition (p, ς) is stored in P and σ, the corresponding set of perturbed general position polynomials is attached to Γ, and w and α are adapted to the new search tree. In particular, the arriving sign condition is the new root of the search tree, and the old search tree is the new left successor of the root. This is illustrated in Fig. 3.

In case the sign conditions added so far are not realizable, the arriving sign condition does not change that result. Thus, the solver does not have to construct new samples and may abort the search immediately, copying the previous entries of W, S, R, and K. This case is captured by lines 13 to 16.

Table 2. Annotations to the variables in Listing 3

Variable	*Initial value*	*Description*
d	fixed	$d \geq \max\{\deg(q) \mid q \in P\}$ (preselected large enough)
m	0	current number of polynomials
n	fixed	dimension, i.e., the number of variables
P	()	current list of polynomials
σ	()	current list of signs
Γ	()	current list of sets of perturbed general position polynomials
W	()	list of search tree leafs such that W_k is the leaf where the search for the first k polynomials stopped
w	ϵ	current leaf of the search tree
α	$(\epsilon \mapsto \emptyset)$	set of samples not yet considered at the leaves of the search tree
S	()	list of m samples where the kth sample satisfies the first k sign conditions, in case they were satisfiable
r	1	flag determining the current state of w being a candidate for representing the reason of unsatisfiability ($r = 1$ means that w is still a candidate)
R	()	list of reasons for the unsatisfiability results where each reason represents a conjunction of constraints (() in case of satisfiability)
K	()	list of satisfiability results such that W_k is the state of satisfiability (1 means satisfiable, 0 not) when the search for the first k polynomials stopped

Table 3. Annotations to the methods in Listing 3

`IRiSC`	Constructor fixing the dimension, the maximal degree of input polynomials, and the initial values for the variables as given in Table 2.
`add`	Adds a constraint in the form of a sign condition (p, σ) to the system of polynomial equations and inequalities.
`backjump`	Jumps back to a previous state of the search, by undoing the last j additions of sign conditions.
`satisfiable`	Asks whether the current sign condition is satisfiable.
`reason`	Returns a reason of unsatisfiability as list of sign conditions if appropriate.
`assignment`	Returns a satisfying assignment if appropriate.

Listing 3. The class IRiSC

```
class IRiSC
{
  const VAR $\delta$, $\gamma$, $\epsilon$;
  $d, m, n \in \mathbb{N}$;
  $P \in \mathbb{Z}[x_1, \dots, x_n]^m$;
  $\sigma \in \{-1, 0, 1\}^m$;
  $\Gamma \in (2^{\mathbb{Z}}[x_1, \dots, x_n])^m$;
  $W \in (\{0, 1\}^m)^m$;
  $w \in \{0, 1\}^m$;
  $\alpha : \{0, 1\}^m \to 2^{\mathbb{R}^n}$;
  $S \in (\mathbb{R}^n)^m$;
  bool $r$;
  $R \in 2^{\mathbb{Z}[x_1, \dots, x_n]}$;
  $K \in \{0, 1\}^m$;

  IRiSC($d, n \in \mathbb{N}$);

  void add($p \in \mathbb{Z}[x_1, \dots, x_n]$, $\varsigma \in \{-1, 0, 1\}$);
  void backjump($i \in \mathbb{N} : 1 \leq i \leq m$);
  bool satisfiable(){ return $K_m$; };
 $2^{\mathbb{Z}[x_1, \dots, x_n]}$ reason(){ return $\{(R_m)_j \mid 1 \leq j \leq |R_m|\}$; };
 $\{x_1, \dots, x_n\} \to \mathbb{R}$ assignment() { return $(x_j \mapsto (S_m)_j \mid 1 \leq j \leq n)$; };
}
```

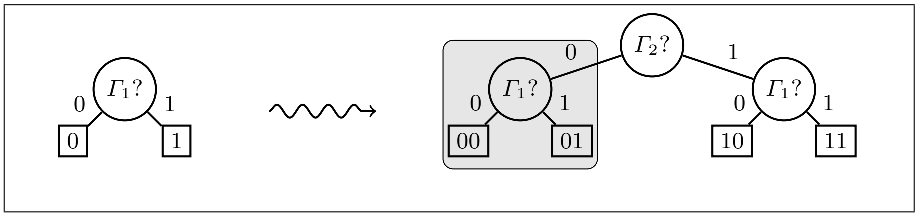


Fig. 3. Evolution of the I-RiSC search tree when adding a new sign condition with its corresponding set of perturbed general position polynomials Γ_2

If the previously added sign conditions are realizable, the search for a sample satisfying all sign conditions continues at the currently smallest leaf w.r.t. $\prec$ which has an empty sample set. Since only those leaves w matter, which choose at least one polynomial for the sample construction ($|w|_1 > 0$), we must exclude the cases where $|w|_1 = 0$ (lines 17 and 18). Lines 19 to 45 comprise the outer loop, enumerating all leaves $w \in \{0, 1\}^m$ with $|w|_1 > 0$ in order of $\prec$.

This loop can be exited in two cases, each of which also terminates: firstly, in case a sample satisfying *all* sign conditions was found; secondly, if w is inadmissible. The samples are constructed for all combinations polynomials of different

Listing 4. IRiSC.add

```
void IRiSC.add($p \in \mathbb{Z}[x_1,\ldots,x_n]$, $\varsigma \in \{-1,0,1\}$)
{
  $m$ := $m+1$;
  $P$ := $P \oplus (p)$; $\sigma$ := $\sigma \oplus (\varsigma)$;
  $\Gamma$ := $\Gamma \oplus (\{\mathrm{PG}^1_{n,k,d}(p), \mathrm{PG}^{-1}_{n,k,d}(p), \mathrm{PG}^{\gamma}_{n,k,d}(p), \mathrm{PG}^{-\gamma}_{n,k,d}(p)\})$;
  $\widetilde{\alpha} : \{0,1\}^m \to 2^{\mathbb{R}^n}$;
  for $v \in \{0,1\}^{m-1}$: {
    $\widetilde{\alpha}(0v)$ := $\alpha(v)$;
    $\widetilde{\alpha}(1v)$ := $\emptyset$;
  }
  $\alpha$ := $\widetilde{\alpha}$;
  $w$ := $\min_{\prec}($ $w$, $\min_{\prec}\{v \in \{0,1\}^m \mid \alpha(v) = \emptyset\}$ $)$;
  if $K_{m-1} = 0$: {
    $W$ := $W \oplus (w)$; $S$ := $S \oplus (1)$; $R$ := $R \oplus (R_{m-1})$; $K$
 := $K \oplus (0)$;
    return;
  }
  if $|w|_1 = 0$:
    $w$ := $\mathrm{next}_{\prec}(w)$;
  while true: {
    if $\alpha(w) = \emptyset$: {
      $\alpha(w)$ := $\bigcup_{Q \in \prod \Gamma_w}$ samples($Q$, $(x_1,\ldots,x_n)$);
      $r$ := true;
    }
    else:
      $r$ := false;
    $S'$ := $\emptyset$;
    for $a \in \alpha(w)$: {
      if $a \in \Re eali_{\sigma_w}(P_w)$: {
        $r$ := false;
        $S'$ := $S' \cup \{a\}$;
        if $a \in \Re eali_{\sigma}(P)$: {
          $W$ := $W \oplus (w)$; $S$ := $S \oplus (a)$; $R$ := $R \oplus (())$;
          $K$ := $K \oplus (1)$; $\alpha(w)$ := $\alpha(w) \cup S'$;
          return;
        }
      }
      $\alpha(w)$ := $\alpha(w) \setminus \{a\}$;
    }
    if $r$ = true: {
     $W$ := $W \oplus (w)$;$S$ := $S \oplus (1)$;$R$ := $R \oplus ((P_w, \sigma_w))$;$K$ := $K \oplus (0)$;
      return;
    }
    $\alpha(w)$ := $S'$;
    $w$ := $\mathrm{next}_{\prec}(w)$;
  }
}
```

sets of perturbed general position polynomials in line 21. This construction is only performed if no samples were constructed for this node beforehand. The inner loop in lines 27 to 38 then iterates over all samples for the current leaf w. The test of satisfiability is two-fold: first, the current sample is checked for satisfying (P_w, σ_w) in line 28; second, it is tested against the complete sign condition (P, σ) in line 31. This proceeding enables on the one hand the storage of the set $S(w)$, by the variable S' which later is assigned to $\alpha(w)$ in line 43. On the other hand, it allows for recognizing an inadmissible leaf, by using the variable r: r is only set to `true` in line 22 where w's samples are constructed newly, and r is switched to `false` immediately in case one sample satisfies the current local sign condition (P_w, σ_w), i.e., if w proves to be admissible. If otherwise all checks against (P_w, σ_w) fail, then w must be inadmissible; and by Lemma 1, (P_w, σ_w) is a minimal infeasible subset, which is stored as such in line 40. Note that r is set to `false` in line 25, where w is identified to be admissible.

IRiSC.backjump. The method `IRiSC.backjump(`$i \in \mathbb{N} : 1 \leq i \leq m$`)` removes the last i entries from P, σ, Γ, W, S, R, and K. It also updates the mapping α.

Listing 5. IRiSC.backjump

```
void IRiSC.backjump($i \in \mathbb{N} : 1 \leq i \leq m$)
{
  $\widetilde{\alpha} : \{0,1\}^{m-i} \to 2^{\mathbb{R}^n}$;
  for $w \in \{0,1\}^m$:
    if $w = 0 \dots 0\, w_{i+1} \dots w_m$:
      $\widetilde{\alpha}(w_{i+1} \dots w_m)$ := $\alpha(w)$;
  $\alpha$ := $\widetilde{\alpha}$; $\widetilde{P}$ := (); $\widetilde{\sigma}$ := $\epsilon$; $\widetilde{\Gamma}$ := ();
  $\widetilde{W}$ := (); $\widetilde{S}$ := (); $\widetilde{R}$ := (); $\widetilde{K}$ := ();
  for $1 \leq k \leq m - i$: {
    $\widetilde{P}$ := $\widetilde{P} \oplus (P_k)$; $\widetilde{\sigma}$ := $\widetilde{\sigma} \oplus (\sigma_k)$; $\widetilde{\Gamma}$ := $\widetilde{\Gamma} \oplus (\Gamma_k)$;
   $\widetilde{W}$ := $\widetilde{W} \oplus (W_k)$; $\widetilde{S}$ := $\widetilde{S} \oplus (S_k)$; $\widetilde{R}$ := $\widetilde{R} \oplus (R_k)$; $\widetilde{K}$ := $\widetilde{K} \oplus (K_k)$;
  }
  $P$ := $\widetilde{P}$; $\sigma$ := $\widetilde{\sigma}$; $\Gamma$ := $\widetilde{\Gamma}$;
  $W$ := $\widetilde{W}$; $S$ := $\widetilde{S}$; $R$ := $\widetilde{R}$; $K$ := $\widetilde{K}$; $m$ := $m - i$; $w$ := $W_m$;
}
```

Fig. 4 shows the execution of `IRiSC.backjump(1)` using the example search tree of Fig. 3.

4.4 Example

We explain the functioning of I-RiSC by an example, illustrating how the I-RiSC search tree evolves when adding the sign conditions $((p_1), (\sigma_1))$, $((p_2), (\sigma_2))$, $((p_3), (\sigma_3))$ one by one, then backtracking 2 steps, and finally add a last sign condition $((p_4), (\sigma_4))$.

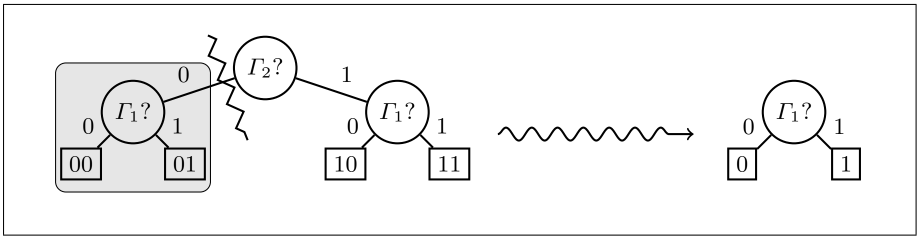
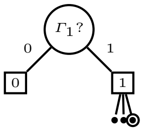


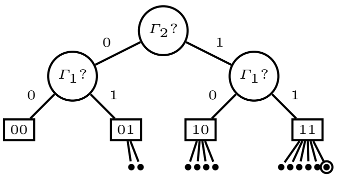
Fig. 4. Evolution of the I-RiSC search tree when backtracking, undoing the last step

We assume $((p_1, p_2), (\sigma_1, \sigma_2))$ and $((p_1, p_3), (\sigma_1, \sigma_3))$ being satisfiable, but $((p_2, p_3), (\sigma_2, \sigma_3))$ is not. We depict sample points constructed for one combination of a leaf by •, and circle the sample satisfying the current sign condition.

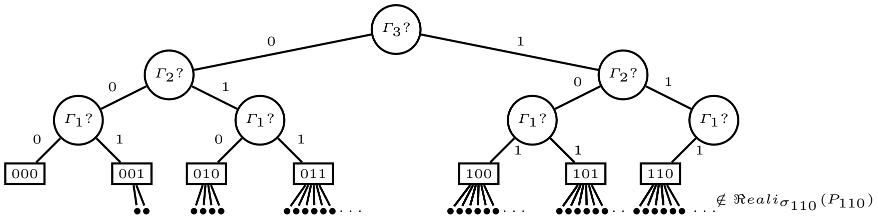
1. Search tree after adding (p_1, σ_1):



2. Search tree after adding (p_2, σ_2):



3. Search tree after adding (p_3, σ_3):

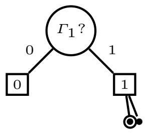


In the first step, I-RiSC constructs the set Γ_1 of perturbed general position polynomials for p_1 and, using Γ_1, constructs three samples for the given sign condition, where the third sample satisfies it. The second condition is added by first constructing the node "Γ_2", which represents the choice of Γ_2, and second, appending the previous search tree as left successor of the new root node "Γ_2". The samples of Γ_1 are discarded, except for two of the two of them, which satisfy (p_1, σ_1) but not $((p_1, p_2), (\sigma_1, \sigma_2))$. I-RiSC proceeds by trying samples for Γ_2 and for (Γ_1, Γ_2). The latter yields a satisfying sample after computing 6 of the possible 16 choices of sets of general position polynomials. In the last step, (p_3, σ_3) is added. Now, following $\prec$ back to the next leaf without samples, Γ_3 is used for sample construction. In this example, the samples generated at the leaves 100, 011, or 101 satisfy their respective sign conditions (P_{100}, σ_{100}), (P_{011}, σ_{011}), or (P_{101}, σ_{101}), but not all sign conditions. The next condition (P_{110}, σ_{110}) is satisfied by none of its respective samples. This allows I-RiSC to stop the search process and store the sign condition (P_{110}, σ_{110}) as reason of unsatisfiability.

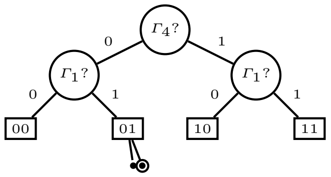
This reason is now removed by performing `IRiSC.backjump(2)`. Thereby, the search tree of the very first step is restored, still containing some of the samples computed for Γ_1. The final step of this example shows the adding of $((p_4), (\sigma_4))$, which here is satisfied already by the next sample, computed for Γ_1.

The resulting search trees may look as follows.

1. Search tree after backtracking 2 steps:



2. Search tree after adding (p_4, σ_4):



4.5 Optimizations

There are several ways to optimize the implementation of I-RiSC. Here, we list three of them.

– The method `IRiSC.add` does not have to compute all the complete set $\prod \Gamma_w$ and all samples for a given w in advance, but could store the current position in the $\prod \Gamma_w$ and the corresponding sample computation. This would at best avoid $4^{|w|_1} - 1$ calls of the method `IRiSC.samples` as well as save the memory consumed by the samples and the tuples of $\prod \Gamma_w$.
– I-RiSC does not yet support theory propagation. As discussed in [15], theory propagation can considerably speed-up the solving procedure. We plan first to implement the I-RiSC solver as proposed in this paper, before further improving it by adding theory propagation.
– Last but not least, since good search heuristics are very important for the practical applicability of I-RiSC, we will investigate them in the near future.

5 Conclusion

In this paper we introduced a modification of an existing decision procedure for real algebra from [3]. Our modified algorithm I-RiSC satisfies the requirement to be embedded into an efficient SMT-solver.

Currently, we work on the implementation of the I-RiSC method. The implementation, based on the `C++` library `GiNaCRA` [11], will allow to develop a complete SMT-solver for real algebra.

There are several points for further optimization of the proposed algorithm. In addition, I-RiSC can be combined with techniques applied in other decision procedures for real algebra. One example is the reduction of the dimension of the input polynomials by variable elimination techniques. E.g., if the multivariate input polynomials are at most quadratic in one variable, then solution formulas can be used for variable elimination, as done by the virtual substitution [17].

References

1. Barrett, C., Sebastiani, R., Seshia, S., Tinelli, C.: Satisfiability Modulo Theories, ch. 26. Frontiers in Artificial Intelligence and Applications [5], vol. 185, pp. 825–885 (2009)
2. Barrett, C., Tinelli, C.: CVC3. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 298–302. Springer, Heidelberg (2007)
3. Basu, S., Pollack, R., Roy, M.: Algorithms in Real Algebraic Geometry. Springer, Heidelberg (2010)
4. Bauer, A., Pister, M., Tautschnig, M.: Tool-support for the analysis of hybrid systems and models. In: DATE 2007, pp. 924–929. European Design and Automation Association (2007)
5. Biere, A., Heule, M., van Maaren, H., Walsh, T.: Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Amsterdam (2009)
6. Bruttomesso, R., Pek, E., Sharygina, N., Tsitovich, A.: The openSMT solver. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 150–153. Springer, Heidelberg (2010)
7. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R.: The mathSAT 4SMT solver. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 299–303. Springer, Heidelberg (2008)
8. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 134–183. Springer, Heidelberg (1975)
9. Dutertre, B., Moura, L.D.: A fast linear-arithmetic solver for DPLL(T). In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 81–94. Springer, Heidelberg (2006)
10. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. Journal on Satisfiability, Boolean Modeling and Computation 1(3-4), 209–236 (2007)
11. Loup, U., Ábrahám, E.: GiNaCRA: A C++ library for real algebraic computations. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 512–517. Springer, Heidelberg (2011)
12. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-Driven Clause Learning SAT Solvers, ch. 4. Frontiers in Artificial Intelligence and Applications [5], vol. 85, pp. 131–153 (2009)
13. Mishra, B.: Algorithmic Algebra. Texts and Monographs in Computer Science. Springer, Heidelberg (1993)
14. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
15. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). Journal of the ACM 53, 937–977 (2006)
16. SMT competition 2010, http://www.smtcomp.org/2010/
17. Weispfenning, V.: Quantifier elimination for real algebra – The quadratic case and beyond. Applicable Algebra in Engineering, Communication and Computing 8(2), 85–101 (1997)
18. Zankl, H., Middeldorp, A.: Satisfiability of non-linear (Ir)rational arithmetic. In: Clarke, E.M., Voronkov, A. (eds.) LPAR-16 2010. LNCS, vol. 6355, pp. 481–500. Springer, Heidelberg (2010)

Variable Tree Automata over Infinite Ranked Alphabets[⋆]

Irini-Eleftheria Mens and George Rahonis

Department of Mathematics
Aristotle University of Thessaloniki
54124, Thessaloniki, Greece
{emens,grahonis}@math.auth.gr

Abstract. We introduce variable tree automata with infinite input ranked alphabets. Our model is based on an underlying bottom-up tree automaton over a finite ranked alphabet containing variable symbols. The underlying tree automaton computes its tree language, and then replaces the variable symbols with symbols from the infinite alphabet following certain rules. We show that the class of recognizable tree languages over infinite ranked alphabets is closed under union and intersection but not under complementation. The emptiness problem is decidable, and the equivalence problem is decidable within special subclasses of variable tree automata. The universality problem is also decidable in a subclass of variable tree automata. We demonstrate the robustness of our model by connecting it to variable finite automata and indicating several characterizations of recognizable tree languages over infinite ranked alphabets.

Keywords: Infinite ranked alphabets, relabelings, variable bottom-up tree automata.

1 Introduction

The automata-theoretic approach, based on finite automata, has been successfully applied so far, in automated reasoning for finite state systems. Nevertheless, a corresponding approach for reasoning on infinite state systems with finite control and infinite data, requires automata working over infinite alphabets. Several authors considered such automata models, namely *register* (cf. [14,16,17,19]), *pebble* (cf. [16,17]), and *data automata* (cf. [1,2]). Recently in [10] (cf. also [11]), the authors considered a new model of finite automaton, the *variable finite automaton* consuming words over infinite alphabets. It is based on an underlying finite automaton with input finite alphabet which consists of a constant alphabet (subalphabet of the infinite alphabet) and variable symbols. The variable symbols are of two types, namely *bounded* and *free* with the restriction that

[⋆] Research of the first author was supported by the RISC-Linz Transnational Access Programme, project SCIEnce (contract No. 026133) of the European Commission FP6 for Integrated Infrastructures Initiatives.

there exists only one free variable symbol. The automaton accepts a language as follows. Firstly, it computes the language of the underlying automaton. Then, it substitutes the variable symbols with letters from the infinite alphabet. For this substitution concrete requirements are imposed. It is shown that variable finite automata have nice properties, but their main advantage is the simplicity of their operation in comparison to the other models of automata over infinite alphabets.

The processing of XML documents led to the investigation of unranked trees and especially to data trees, cf. for instance [2,3,13] and the references in these papers. Important results have been established for XML reasoning using tree automata on data trees. Nevertheless, those automata are quite complicated according to implementation and application. In this paper, we introduce a simple tree automaton model accepting trees from an infinite ranked alphabet with bounded rank. Our model is based on an underlying bottom-up tree automaton with finite input ranked alphabet, consisting of a subalphabet of the infinite alphabet, and two variable ranked alphabets, namely *bounded* and *free*. Firstly, we compute the tree language of the underlying automaton, and then we apply certain relabelings on the variable symbols assigning letters (of the same rank) from the infinite alphabet. In this way we get the tree language accepted by the tree automaton. We call our model a *variable tree automaton*. We show that the class of tree languages accepted by variable tree automata is closed under union and intersection. The equivalence problem is decidable in two subclasses of variable tree automata. Our formalism allows further characterizations of the recognizable tree languages as well as the consideration of top-down tree automata which are shown to be equivalent to the bottom-up models. Furthermore, this formalism clearly indicates the definition of tree automata over infinite trees, not considered in this paper. This shows the robustness of our model; moreover there is a natural connection of our tree automata, via monadic alphabets, with the variable finite automata of [10,11]. In this way, we show for instance, that the equivalence is decidable in two subclasses of variable finite automata. Furthermore, we show that our recognizable tree languages are not closed under complementation, using a corresponding result from the string case. One more well-known result from classical tree language theory fails in our setup, namely the class of the branches of our recognizable tree languages does not coincide with the recognizable languages over infinite alphabets. Several problems remain open, the most important may be, the determinization of variable tree automata and the closure under tree homomorphisms and tree substitutions.

2 Preliminaries

For a set X, we let X^* for the free monoid generated by X, and ε for the unit element of X^* which is called the *empty word* if we consider X as an alphabet. We shall denote by $\mathbb{N}$ the set of natural numbers and let $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. The *prefix relation* $\leq$ over $\mathbb{N}^*$ is a partial order defined in the usual way: for every $w, v \in \mathbb{N}^*$, $w \leq v$ iff there exists $u \in \mathbb{N}^*$ such that $wu = v$. A set $A \subseteq \mathbb{N}^*$ is called *prefix-closed* if $wu \in A$ implies $w \in A$.

A (finite) *ranked alphabet* Σ is a pair (Σ, rk) (simply denoted by Σ) where Σ is a finite set and $rk : \Sigma \to \mathbb{N}$. As usually, we set $\Sigma_k = \{\sigma \in \Sigma \mid rk(\sigma) = k\}$, $k \geq 0$ and $\deg(\Sigma) = \max\{k \in \mathbb{N} \mid \Sigma_k \neq \emptyset\}$ is the *degree of* Σ.

The union $\Sigma \cup \Gamma$, the intersection $\Sigma \cap \Gamma$, the difference $\Sigma \setminus \Gamma$, and the cartesian product $\Sigma \times \Gamma$ of two ranked alphabets Σ and Γ are ranked alphabets defined, respectively, by $(\Sigma \cup \Gamma)_k = \Sigma_k \cup \Gamma_k$, $(\Sigma \cap \Gamma)_k = \Sigma_k \cap \Gamma_k$, $(\Sigma \setminus \Gamma)_k = \Sigma_k \setminus \Gamma_k$, and $(\Sigma \times \Gamma)_k = \Sigma_k \times \Gamma_k$, for every $k \geq 0$.

A *finite tree t over Σ* is a partial mapping $t : \mathbb{N}_+^* \to \Sigma$ such that the domain $dom(t)$ is a nonempty prefix-closed set, and if $t(w) \in \Sigma_k$, $k \geq 0$, then for $i \in \mathbb{N}_+$, $wi \in dom(t)$ iff $1 \leq i \leq k$. The set $dom(t)$ is the set of *nodes of* t, and for every $w \in dom(t)$ the symbol $t(w)$ is the *label of t at w*. Hence, $t(dom(t))$ is the set of all labels of t. We denote by T_Σ the set of all finite trees over Σ. Alternatively, the set T_Σ is defined to be the smallest set T such that if $k \geq 0$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T$, then $\sigma(t_1, \ldots, t_k) \in T$. If $\sigma \in \Sigma_0$, then we write just σ for $\sigma(\,)$. By definition $T_\Sigma \neq \emptyset$ iff $\Sigma_0 \neq \emptyset$, thus we assume that every ranked alphabet we consider in this paper has at least one nullary symbol. In the following, we recall tree automata (cf. [4,8,9]).

A (nondeterministic) bottom-up tree automaton (buta for short) is a quadruple $M = (Q, \Sigma, \Delta, F)$ where Q is the finite state set, Σ is the input ranked alphabet, $\Delta = (\Delta_k)_{k \geq 0}$ is the family of transitions with $\Delta_k \subseteq Q^k \times \Sigma_k \times Q$ for every $k \geq 0$, and $F \subseteq Q$ is the final state set.

Let $t \in T_\Sigma$. A run of M over t is a partial mapping $r_t : \mathbb{N}_+^* \to Q$ with $dom(r_t) = dom(t)$, such that for every $w \in dom(t)$ it holds $((r_t(w1), \ldots, r_t(w \centerdot rk(t(w)))), t(w), r_t(w)) \in \Delta_{rk(t(w))}$. The run r_t is called successful if $r_t(\varepsilon) \in F$. The tree t is recognized (or accepted) by M if there is a successful run of M over t. The tree language of M consists of all trees accepted by M and is denoted by $L(M)$.

Dually, a (nondeterministic) top-down tree automaton (tdta for short) is a quadruple $N = (Q, \Sigma, Q_0, \Delta)$ where Q is the finite state set, Σ is the input ranked alphabet, $Q_0 \subseteq Q$ is the set of initial states, and $\Delta = (\Delta_k)_{k \geq 0}$ is the family of transitions with $\Delta_k \subseteq Q \times \Sigma_k \times Q^k$ for every $k \geq 0$.

Given a tree $t \in T_\Sigma$, a run of N over t is a partial mapping $r_t : \mathbb{N}_+^* \to Q$ with $dom(r_t) = dom(t)$, such that for every $w \in dom(t)$, it holds $(r_t(w), t(w), (r_t(w1), \ldots, r_t(w \centerdot rk(t(w))))) \in \Delta_{rk(t(w))}$. Now the run r_t is called successful if $r_t(\varepsilon) \in Q_0$. The tree t is recognized (or accepted) by N if there is a successful run of N over t. The tree language of N consists of all trees accepted by N and is denoted by $L(N)$. It is well-known that the classes of tree languages accepted by all buta and all tdta coincide. This common class is the class of recognizable tree languages and is denoted by REC. If we need to restrict REC over a specific ranked alphabet Σ, then we write $REC(\Sigma)$.

3 Variable Tree Automata

In this section, we consider tree automata whose input ranked alphabet is infinite. More precisely, a ranked alphabet (Σ, rk) is infinite if Σ is infinite but still its degree is a finite number. The set T_Σ of finite trees over Σ is defined in the same way as for finite ranked alphabets. Again, we will assume that every infinite ranked alphabet we shall use in this paper has at least one nullary symbol.

Let Σ and Σ' be (infinite) ranked alphabets. A *relabeling from* Σ *to* Σ' is a family of mappings $(h_k)_{k\geq 0}$ such that for every $k \geq 0$, $h_k : \Sigma_k \to \mathcal{P}(\Sigma'_k)$. The relabeling $(h_k)_{k\geq 0}$ induces a mapping $h : T_\Sigma \to \mathcal{P}(T_{\Sigma'})$ which is defined inductively in the following way. For every $t \in T_\Sigma$ with $t = \sigma(t_1, \ldots, t_k)$ we let $h(t) = \{\sigma'(t'_1, \ldots, t'_k) \mid \sigma' \in h_k(\sigma) \text{ and } t'_i \in h(t_i), 1 \leq i \leq k\}$. Note that $dom(s) = dom(t)$ for every $s \in h(t)$. For simplicity, sometimes we write just h for h_k, $k \geq 0$.

Next let Z and Y be two nonempty finite ranked alphabets with $\deg(Z) \leq \deg(\Sigma)$, $\deg(Y) \leq \deg(\Sigma)$, and $card(Y_k) \leq 1$ for every $k \geq 0$. Moreover, we assume that Σ, Z, and Y are pairwise disjoint and $Z_k = Y_k = \emptyset$ if $\Sigma_k = \emptyset$ for every $0 \leq k \leq \deg(\Sigma)$. For reasons explained in the sequel, we call the elements of Z *bounded variable symbols* (or *bounded variables*) and the elements of Y *free variable symbols* (or *free variables*). Let $A \subseteq \Sigma$ be a finite subalphabet of Σ and set $\Gamma = A \cup Z \cup Y$. A relabeling h from Γ to Σ is called *valid on* Γ if

(i) it is the identity on A,[1]
(ii) $card(h_k(z)) = 1$ for every $k \geq 0$ and $z \in Z_k$,
(iii) h is injective on Z and $A_k \cap h_k(Z_k) = \emptyset$ for every $k \geq 0$,
(iv) $h_k(y) = \Sigma_k \setminus (A_k \cup h_k(Z_k))$ for every $k \geq 0$ and $y \in Y_k$.

Thus, the application of h on a tree t over Γ, assigns to every occurrence of a symbol $z \in Z$ in t the same label from Σ, but it is possible to assign different labels to different occurrences of the same symbol $y \in Y$ in t. This justifies the names bounded and free for the set of variables Z and Y, respectively. Clearly, it is sufficient to define a valid relabeling on Γ only on Z. We denote by $VR(\Gamma)$ the class of all valid relabelings on Γ. Clearly $VR(\Gamma)$ is infinite. Now we are ready to introduce the main concept of the paper.

A *variable bottom-up tree automaton* (*vbuta* for short) is a pair $\mathcal{M} = \langle \Sigma, M \rangle$ where Σ is an infinite ranked alphabet and $M = (Q, \Gamma_M, \Delta, F)$ is a buta with input alphabet $\Gamma_M = \Sigma_M \cup Z \cup Y$ where $\Sigma_M \subseteq \Sigma$ is a finite ranked subalphabet, Z is a finite ranked alphabet of bounded variables, and Y is a finite ranked alphabet of free variables. The *tree language* $L(\mathcal{M})$ *of* $\mathcal{M}$ is defined by $L(\mathcal{M}) = \bigcup_{h \in VR(\Gamma_M)} h(L(M))$. Then, we say that the tree language $L(\mathcal{M})$ is *accepted* (or *recognized*) by $\mathcal{M}$. We denote by $VREC(\Sigma)$ the class of tree languages over Σ accepted by all vbuta with input ranked alphabet Σ. Moreover, we denote by $VREC$ the class of all recognizable tree languages over all infinite ranked alphabets.

Example 1. Consider the vbuta $\mathcal{M} = \langle \Sigma, M \rangle$ with $M = (Q, \Gamma_M, \Delta, F)$, where $Q = \{q, q'\}$, $Z = Z_2 = \{z\}$, $Y = \{y_0, y_2\}$ with $rk(y_0) = 0, rk(y_2) = 2$, $F = \{q'\}$, and $\Delta = \{(y_0, q), ((q, q), y_2, q), ((q, q), z, q')\}$. Then, $L(\mathcal{M})$ consists of all binary trees t over the infinite alphabet Σ, such that the label of the root(=$t(\varepsilon)$) of t is different from the label of every other node of t.

[1] Abusing notations we identify $\{a\}$ with a, for every $a \in A$.

Example 2. Let $\mathcal{M} = \langle \Sigma, M \rangle$ with $M = (\{q\}, \Gamma_M, \Delta, \{q\})$, where $\Sigma_M = \emptyset$, $Y_k = \{y_k\}$ and $\Delta_k = \{((q, \dots, q), y_k, q)\}$ for every $0 \le k \le \deg(\Sigma)$ with $\Sigma_k \neq \emptyset$. Trivially, the vbuta $\mathcal{M}$ accepts every tree in T_Σ, hence $L(\mathcal{M}) = T_\Sigma$.

Example 3. For every recognizable tree language L over a finite ranked alphabet $\Sigma' \subseteq \Sigma$, we have $L \in VREC(\Sigma)$. Indeed, a buta M accepting L is a vbuta $\mathcal{M} = \langle \Sigma, M \rangle$ whose transitions do not contain any variable.

From the above Example 3, we immediately obtain the subsequent result.

Proposition 1. *$REC \subseteq VREC$.*

In the rest of the paper Σ will denote an infinite ranked alphabet.

Variable top-down tree automata (*vtdta* for short) are defined analogously. More precisely, a vtdta is a pair $\mathcal{M} = \langle \Sigma, M \rangle$ where $M = (Q, \Gamma_M, Q_0, \Delta)$ is a tdta. The tree language of $\mathcal{M}$ is defined by $L(\mathcal{M}) = \bigcup_{h \in VR(\Gamma_M)} h(L(M))$. Using the equivalence of buta and tdta, we trivially get the next equality result.

Proposition 2. *The class of tree languages accepted by all vtdta with input ranked alphabet Σ coincides with $VREC(\Sigma)$.*

In the sequel, we intend to investigate closure properties of the class $VREC(\Sigma)$. Nevertheless, standard constructions on tree automata over finite ranked alphabets cannot be applied in a straightforward way in our setup. For instance, consider two vbuta $\mathcal{M}^{(i)} = \left\langle \Sigma, M^{(i)} \right\rangle$ where $M^{(i)} = (Q^{(i)}, \Gamma^{(i)}, \Delta^{(i)}, F^{(i)})$ with $\Gamma^{(i)} = \Sigma^{(i)} \cup Z^{(i)} \cup Y^{(i)}$, $i = 1, 2$. Let M be the well-known *disjoint union* (cf. [4]) of $M^{(1)}$ and $M^{(2)}$ and $h^{(1)} \in VR(\Gamma^{(1)})$ such that $h^{(1)}(z) \in \Sigma^{(2)}$ for some $z \in Z^{(1)}$. Then $h^{(1)}$ cannot be anymore a valid relabeling on $\Gamma^{(1)} \cup \Gamma^{(2)}$ which implies that if $t \in h^{(1)}(L(M^{(1)}))$ and there is a node of t with label $h^{(1)}(z)$, then it is not guaranteed that $t \in L(\mathcal{M})$, where $\mathcal{M} = \langle \Sigma, M \rangle$. The subsequent Lemma 1 will be crucial for our constructions. For its proof we need some preliminary matter.

Let $\mathcal{M} = \langle \Sigma, M \rangle$ be a vbuta, $h \in VR(\Gamma_M)$, and $\Sigma' \subseteq \Sigma$ a finite subalphabet with $\Sigma' \setminus \Sigma_M \neq \emptyset$. We consider the buta $M_h = (Q_h, \Gamma_{M_h}, \Delta_h, F_h)$ where Q_h is a copy of Q, i.e., $Q_h = \{q^h \mid q \in Q\}$ and $F_h = \{q^h \in Q_h \mid q \in F\}$. The input alphabet Γ_{M_h} is defined by $\Gamma_{M_h} = \Sigma_M \cup (h(Z \cup Y) \cap \Sigma') \cup (Z \setminus Z_h) \cup Y$ where $Z_h = \{z \in Z \mid h(z) \in \Sigma' \setminus \Sigma_M\}$. The family Δ_h of transitions is determined as follows. For every $k \geq 0$ and $z \in (Z_h)_k$ we replace every transition $((q_1, \dots, q_k), z, q) \in \Delta_k$ with the new transition $((q_1^h, \dots, q_k^h), h_k(z), q^h)$. We repeat the same procedure in case $h_k(y) \cap (\Sigma' \setminus \Sigma_M)_k \neq \emptyset$, hence we replace every transition $((q_1, \dots, q_k), y, q) \in \Delta_k$ with the transitions of the form $((q_1^h, \dots, q_k^h), \sigma', q^h)$ for every $\sigma' \in h_k(y) \cap (\Sigma' \setminus \Sigma_M)_k$. Formally, for every $k \geq 0$ we set

$$\begin{aligned}(\Delta_h)_k = &\{\left(\left(q_1^h, \dots, q_k^h\right), \sigma, q^h\right) \mid \left(\left(q_1, \dots, q_k\right), \sigma, q\right) \in \Delta_k, \sigma \in \left(\Sigma_M\right)_k \cup (Z \setminus Z^h)_k \cup Y_k\} \\ &\cup \left\{\left(\left(q_1^h, \dots, q_k^h\right), h_k(z), q^h\right) \mid \left(\left(q_1, \dots, q_k\right), z, q\right) \in \Delta_k, z \in \left(Z_h\right)_k\right\} \\ &\cup \left\{\left(\left(q_1^h, \dots, q_k^h\right), \sigma', q^h\right) \mid \left(\left(q_1, \dots, q_k\right), y, q\right) \in \Delta_k, \sigma' \in h_k(y) \cap (\Sigma' \setminus \Sigma_M)_k\right\}.\end{aligned}$$

The alphabets Σ_M, Z, Y, and Σ' are finite, thus by applying the above procedure for every $h \in VR(\Gamma_M)$ we get a finite number of buta modulo the identification of the sets of states. Let V be a finite subset of $VR(\Gamma_M)$ determining this finite class of tree automata and $M_g = (Q_g, \Gamma_{M_g}, \Delta_g, F_g)$ for every $g \in V$. Without any loss, we assume the sets Q_g, $g \in V$, to be pairwise disjoint. Now we consider the buta $M_{(\Sigma',V)} = (Q_V, \Gamma_V, \Delta_V, F_V)$ with $\Gamma_V = \Sigma_M \cup \Sigma' \cup Z \cup Y$, $Q_V = \bigcup_{g\in V} Q_g$, $F_V = \bigcup_{g\in V} F_g$, and $\Delta_V = \bigcup_{g\in V} \Delta_g$. Clearly $L\left(M_{(\Sigma',V)}\right) = \bigcup_{g\in V} L(M_g)$ and thus $L(\mathcal{M}_{(\Sigma',V)}) = \bigcup_{h\in VR(\Gamma_V)} h\left(\bigcup_{g\in V} L(M_g)\right)$.

Lemma 1. $L(\mathcal{M}) = L(\mathcal{M}_{(\Sigma',V)})$.

Proof. Let $t \in T_\Sigma$ and assume firstly that $t \in L(\mathcal{M})$. By definition, there is a tree $s \in T_{\Gamma_M}$ and a valid relabeling $h \in VR(\Gamma_M)$ such that $t \in h(s)$. By construction of the vbuta $\mathcal{M}_{(\Sigma',V)}$ there is a valid relabeling $g \in V$ such that $g(z) = h(z)$ for every $z \in Z_h$ and $g(y) \cap \Sigma' = h(y) \cap \Sigma'$ for every $y \in Y$. We consider the tree s' with $dom(s') = dom(s)$, defined as follows. For every $w \in dom(s')$ we let

$$s'(w) = \begin{cases} s(w) \text{ if } (s(w) \in \Sigma_M \cup (Z \setminus Z_h)) \text{ or } (s(w) = y \text{ and } t(w) \notin \Sigma' \setminus \Sigma_M) \\ t(w) \text{ if } (s(w) \in Z_h) \text{ or } (s(w) = y \text{ and } t(w) \in \Sigma' \setminus \Sigma_M). \end{cases}$$

Clearly $s' \in L(M_g)$. We consider a valid relabeling $h' \in VR(\Gamma_V)$ defined in the following way. For every $z \in (Z \setminus Z_h)_k$, $k \geq 0$, we let $h'_k(z) = h_k(z)$, for every $z \in (Z_h)_k$, $k \geq 0$, we (nondeterministically) let $h'_k(z) \in (\Sigma \setminus (\Sigma_M \cup \Sigma' \cup h(Z \setminus Z_h)))_k$. Thus, for every $k \geq 0$ we have $h'_k(y) = (\Sigma \setminus (\Sigma_M \cup \Sigma' \cup h'(Z)))_k$. Trivially $t \in h'(s')$, hence $t \in h'(L(M_g))$.

Conversely, assume that $t \in L(\mathcal{M}_{(\Sigma',V)})$. Then, there is a tree $s \in L(M_{(\Sigma',V)})$ and a valid relabeling $h \in VR(\Gamma_V)$ such that $t \in h(s)$. This in turn implies that there is a $g \in V$ such that $s \in L(M_g)$ hence $t \in h(L(M_g))$. By construction of M_g, there is a tree $s' \in L(M)$ such that $s \in g'(s')$ where $g' : \Gamma_M \to \mathcal{P}(\Sigma) \cup Z \cup Y$ is a mapping which coincides with g on $\Sigma_M \cup Z_g$, it is the identity on $Z \setminus Z_g$, and $g'(y) = g(y) \cup \{y\}$ for every $y \in Y$. Now we consider the relabeling h' from Γ_M to Σ defined in the following way. It is the identity on Σ_M, and for every $k \geq 0$, $h'_k(z) = h_k(z)$ for every $z \in (Z \setminus Z_g)_k$, $h'_k(z) = g_k(z)$ for every $z \in (Z_g)_k$, and $h'_k(y) = h_k(y) \cup \left((g_k(y) \cap \Sigma'_k) \setminus g_k\left((Z_g)_k\right)\right)$ for $y \in Y_k$ (in fact $(g_k(y) \cap \Sigma'_k) \cap g_k\left((Z_g)_k\right) = \emptyset$ since g is a valid relabeling on Γ_M). Trivially h' is a valid relabeling on Γ_M and $t \in h'(s')$. We conclude that $t \in L(\mathcal{M})$ and our proof is completed.

Proposition 3. *The class $VREC(\Sigma)$ is closed under union.*

Proof. Let $\mathcal{M}^{(i)} = \left\langle \Sigma, M^{(i)} \right\rangle$, $i = 1, 2$, be two vbuta where $M^{(i)} = (Q^{(i)}, \Gamma^{(i)}, \Delta^{(i)}, F^{(i)})$ with $\Gamma^{(i)} = \Sigma^{(i)} \cup Z^{(i)} \cup Y^{(i)}$. Without any loss, we assume $Q^{(1)} \cap Q^{(2)} = \emptyset$ and the variable sets $Z^{(1)}$, $Z^{(2)}$, $Y^{(1)}$, and $Y^{(2)}$ to be pairwise disjoint. We consider the buta $M^{(1)}_{(\Sigma^{(2)},V_1)} = \left(Q^{(1)}_{V_1}, \Gamma^{(1)} \cup \Sigma^{(2)}, \Delta^{(1)}_{V_1}, F^{(1)}_{V_1}\right)$ and $M^{(2)}_{(\Sigma^{(1)},V_2)} = \left(Q^{(2)}_{V_2}, \Gamma^{(2)} \cup \Sigma^{(1)}, \Delta^{(2)}_{V_2}, F^{(2)}_{V_2}\right)$ obtained by the procedure described before Lemma 1. Again without any loss, we can assume that $Q^{(1)}_{V_1} \cap Q^{(2)}_{V_2} = \emptyset$.

We consider the buta $M = (Q_{V_1}^{(1)} \cup Q_{V_2}^{(2)}, \Gamma, \overline{\Delta}_{V_1}^{(1)} \cup \overline{\Delta}_{V_2}^{(2)}, F_{V_1}^{(1)} \cup F_{V_2}^{(2)})$ which is obtained by the disjoint union of $M^{(1)}_{(\Sigma^{(2)},V_1)}$ and $M^{(2)}_{(\Sigma^{(1)},V_2)}$ by letting $\Gamma = \Sigma^{(1)} \cup \Sigma^{(2)} \cup Z^{(1)} \cup Z^{(2)} \cup Y$. The ranked alphabet Y of free variables is assumed disjoint from $Y^{(1)}$ and $Y^{(2)}$ with $card(Y_k) = \max\{card(Y_k^{(1)}), card(Y_k^{(2)})\}$ for every $k \geq 0$. The transition set $\overline{\Delta}_{V_i}^{(i)}$ is obtained from $\Delta_{V_i}^{(i)}$, $i = 1, 2$, by replacing every occurrence of $y^{(i)} \in Y_k^{(i)}$ with the new symbol $y \in Y_k$, $k \geq 0$. We will show that $L(\mathcal{M}) = L\left(\mathcal{M}^{(1)}\right) \cup L\left(\mathcal{M}^{(2)}\right)$ where $\mathcal{M} = \langle \Sigma, M \rangle$. For this, let $t \in L(\mathcal{M})$. Then, there exists a tree $s \in L(M)$ and a valid relabeling $h \in VR(\Gamma)$ such that $t \in h(s)$. Furthermore, there is a tree $s' \in L\left(M^{(1)}_{(\Sigma^{(2)},V_1)}\right) \cup L\left(M^{(2)}_{(\Sigma^{(1)},V_2)}\right)$ such that s is obtained from s' by replacing, for every $k \geq 0$, every occurrence of $y^{(1)} \in Y_k^{(1)}$ and $y^{(2)} \in Y_k^{(2)}$ with $y \in Y_k$ (in fact, by construction of the finite automaton M, only free variables from $Y^{(1)}$ or $Y^{(2)}$ occur in s'). Assume firstly that $s' \in L\left(M^{(1)}_{(\Sigma^{(2)},V_1)}\right)$. We consider the relabeling h' from $\Gamma^{(1)} \cup \Sigma^{(2)}$ to Σ which coincides with h on $\Sigma^{(1)} \cup \Sigma^{(2)} \cup Z^{(1)}$, and $h'_k(y^{(1)}) = h_k(y) \cup h_k\left(Z_k^{(2)}\right)$ for every $k \geq 0$ and $y^{(1)} \in Y_k^{(1)}$. Trivially $h' \in VR\left(\Gamma^{(1)} \cup \Sigma^{(2)}\right)$ and $t \in h'(s')$, hence $t \in L\left(\mathcal{M}^{(1)}_{(\Sigma^{(2)},V_1)}\right)$. Thus, by Lemma 1 we get $t \in L\left(\mathcal{M}^{(1)}\right)$. The case $s' \in L\left(M^{(2)}_{(\Sigma^{(1)},V_2)}\right)$ is treated similarly.

Conversely, let $t \in L\left(\mathcal{M}^{(1)}\right) \cup L\left(\mathcal{M}^{(2)}\right)$ and assume that $t \in L\left(\mathcal{M}^{(1)}\right)$. By Lemma 1, we get that $t \in L\left(\mathcal{M}^{(1)}_{(\Sigma^{(2)},V_1)}\right)$ which implies that there is a tree $s \in L\left(M^{(1)}_{(\Sigma^{(2)},V_1)}\right)$ and a valid relabeling $h \in VR(\Gamma^{(1)} \cup \Sigma^{(2)})$ such that $t \in h(s)$. We consider a valid relabeling h' from Γ to Σ determined as follows. It coincides with h on $\Sigma^{(1)} \cup \Sigma^{(2)} \cup Z^{(1)}$ and we let (nondeterministically) $h'_k(z) \in \Sigma_k \setminus \left(\Sigma^{(1)} \cup \Sigma^{(2)} \cup h(Z^{(1)}) \cup \Sigma''\right)_k$ for every $k \geq 0$ and $z \in Z_k^{(2)}$, where Σ'' is set of all labels of t which are obtained by replacing (via h) all occurrences of free variables in s. Let s' be the tree obtained from s by replacing, for every $k \geq 0$, every occurrence of $y^{(1)} \in Y_k^{(1)}$ with $y \in Y_k$. Clearly, $s' \in L(M)$ and $t \in h'(s')$, hence $t \in L(\mathcal{M})$. The case $t \in L\left(\mathcal{M}^{(2)}\right)$ is treated similarly, and we are done.

Proposition 4. *The class $VREC(\Sigma)$ is closed under intersection.*

Proof. Let $\mathcal{M}^{(i)} = \langle \Sigma, M^{(i)} \rangle$, $i = 1, 2$, be two vbuta where $M^{(i)} = (Q^{(i)}, \Gamma^{(i)}, \Delta^{(i)}, F^{(i)})$ with $\Gamma^{(i)} = \Sigma^{(i)} \cup Z^{(i)} \cup Y^{(i)}$. Without any loss, we assume the sets $Z^{(1)}$, $Z^{(2)}$, $Y^{(1)}$, and $Y^{(2)}$ to be pairwise disjoint. Consider the buta $M^{(1)}_{(\Sigma^{(2)},V_1)} = (Q_{V_1}^{(1)}, \Gamma^{(1)} \cup \Sigma^{(2)}, \Delta_{V_1}^{(1)}, F_{V_1}^{(1)})$ and $M^{(2)}_{(\Sigma^{(1)},V_2)} = \left(Q_{V_2}^{(2)}, \Gamma^{(2)} \cup \Sigma^{(1)}, \Delta_{V_2}^{(2)}, F_{V_2}^{(2)}\right)$ obtained by the procedure described before Lemma 1. We consider also the ranked alphabet $((Z^{(1)} \cup Y^{(1)}) \times (Z^{(2)} \cup Y^{(2)})) \setminus Y$, where $Y = Y^{(1)} \times Y^{(2)}$, and a maximal subalphabet $R \subseteq ((Z^{(1)} \cup Y^{(1)}) \times (Z^{(2)} \cup Y^{(2)})) \setminus Y$ satisfying the next condition: every element of $Z^{(1)}$ (resp. $Z^{(2)}$) occurs as a left (resp. right) coordinate in at most one pair in R. Let $R_1, \ldots, R_m$ be all such alphabets. For every

$1 \leq j \leq m$, we consider the buta $M_{R_j} = \left(Q^{(1)}_{V_1} \times Q^{(2)}_{V_2}, \Gamma_{R_j}, \Delta_{R_j}, F^{(1)}_{V_1} \times F^{(2)}_{V_2}\right)$ with $\Gamma_{R_j} = \Sigma^{(1)} \cup \Sigma^{(2)} \cup R_j \cup Y$, where R_j is the set of bounded variables and Y is the set of free variables. The set Δ_{R_j} of transitions is defined, for every $k \geq 0$, in the following way.

$$\begin{aligned}(\Delta_{R_j})_k = \{&(((q_1^{(1)}, q_1^{(2)}), \ldots, (q_k^{(1)}, q_k^{(2)})), \sigma, (q^{(1)}, q^{(2)})) \\ &\quad \mid ((q_1^{(i)}, \ldots, q_k^{(i)}), \sigma, q^{(i)}) \in \Delta^{(i)}_{V_i},\ i = 1,2,\ \sigma \in \Sigma^{(1)}_k \cup \Sigma^{(2)}_k\} \\ \cup\{&(((q_1^{(1)}, q_1^{(2)}), \ldots, (q_k^{(1)}, q_k^{(2)})), (x^{(1)}, x^{(2)}), (q^{(1)}, q^{(2)})) \\ &\quad \mid ((q_1^{(i)}, \ldots, q_k^{(i)}), x^{(i)}, q^{(i)}) \in \Delta^{(i)}_{V_i},\ i = 1,2,\ (x^{(1)}, x^{(2)}) \in (R_j)_k \cup Y_k\}.\end{aligned}$$

We will show that $L\left(\mathcal{M}^{(1)}\right) \cap L\left(\mathcal{M}^{(2)}\right) = L(\mathcal{M}_{R_1}) \cup \ldots \cup L(\mathcal{M}_{R_m})$ where $\mathcal{M}_{R_j} = \langle \Sigma, M_{R_j} \rangle$ for every $1 \leq j \leq m$. To this end, let $t \in L\left(\mathcal{M}^{(1)}\right) \cap L\left(\mathcal{M}^{(2)}\right)$. Then, by Lemma 1, there are trees $s_1 \in L\left(M^{(1)}_{(\Sigma^{(2)}, V_1)}\right)$, $s_2 \in L\left(M^{(2)}_{(\Sigma^{(1)}, V_2)}\right)$ and valid relabelings $h^{(1)} \in VR(\Gamma^{(1)} \cup \Sigma^{(2)})$, $h^{(2)} \in VR(\Gamma^{(2)} \cup \Sigma^{(1)})$ such that $t \in h^{(1)}(s_1) \cap h^{(2)}(s_2)$. We get $dom(s_1) = dom(s_2) = dom(t)$. Moreover, for every $w \in dom(t)$ we have either $t(w) \in \Sigma^{(1)} \cup \Sigma^{(2)}$, hence $s_1(w) = s_2(w) = t(w)$ or $t(w) \in \Sigma \setminus (\Sigma^{(1)} \cup \Sigma^{(2)})$. The latter case implies:

– there are bounded variables $z^{(1)} \in Z^{(1)}, z^{(2)} \in Z^{(2)}$ such that $s_1(w) = z^{(1)}, s_2(w) = z^{(2)}$ and $h^{(1)}(z^{(1)}) = h^{(2)}(z^{(2)}) = t(w)$, or
– there is a bounded variable $z^{(1)} \in Z^{(1)}$ such that $s_1(w) = z^{(1)}, s_2(w) = y^{(2)} \in Y^{(2)}$ and $h^{(1)}(z^{(1)}) = t(w) \in h^{(2)}(y^{(2)})$, or
– there is a bounded variable $z^{(2)} \in Z^{(2)}$ such that $s_2(w) = z^{(2)}, s_1(w) = y^{(1)} \in Y^{(1)}$ and $h^{(2)}(z^{(2)}) = t(w) \in h^{(1)}(y^{(1)})$, or
– $s_1(w) = y^{(1)} \in Y^{(1)}$, $s_2(w) = y^{(2)} \in Y^{(2)}$, and $t(w) \in h^{(1)}(y^{(1)}) \cap h^{(2)}(y^{(2)})$.

By definition of the alphabets $R_1, \ldots, R_m$, there is an index $1 \leq j \leq m$ such that $\{(s_1(w), s_2(w)) \mid w \in dom(t) \text{ and } t(w) \in \Sigma \setminus (\Sigma^{(1)} \cup \Sigma^{(2)})\} \subseteq R_j \cup Y$. We define a valid relabeling $h \in VR(\Gamma_{R_j})$ by letting $h((x^{(1)}, x^{(2)})) = h^{(1)}(x^{(1)})$ for every $(x^{(1)}, x^{(2)}) \in \left(Z^{(1)} \times (Z^{(2)} \cup Y^{(2)})\right) \cap R_j$ and $h((x^{(1)}, x^{(2)})) = h^{(2)}(x^{(2)})$ for every $(x^{(1)}, x^{(2)}) \in \left(Y^{(1)} \times Z^{(2)}\right) \cap R_j$. Consider the tree s, with $dom(s) = dom(t)$ defined by $s(w) = t(w)$ if $t(w) \in \Sigma^{(1)} \cup \Sigma^{(2)}$, and $s(w) = (s_1(w), s_2(w))$ otherwise. Trivially $s \in L(M_{R_j})$, and we get $t \in h(s)$ which in turn implies that $t \in L(\mathcal{M}_{R_j})$.

Conversely, let $1 \leq j \leq m$ and $t \in L(\mathcal{M}_{R_j})$. Then there is an $s \in L(M_{R_j})$ and a valid relabeling $h \in VR(\Gamma_{R_j})$ such that $t \in h(s)$. We consider the valid relabelings $h^{(1)} \in VR(\Gamma^{(1)} \cup \Sigma^{(2)})$ and $h^{(2)} \in VR(\Gamma^{(2)} \cup \Sigma^{(1)})$ defined as follows. For every $z^{(1)} \in Z^{(1)}$ such that there is an $x^{(2)} \in Z^{(2)} \cup Y^{(2)}$ with $(z^{(1)}, x^{(2)}) \in R_j$ we let $h^{(1)}(z^{(1)}) = h((z^{(1)}, x^{(2)}))$, for every $z^{(2)} \in Z^{(2)}$ such that there is an $x^{(1)} \in Z^{(1)} \cup Y^{(1)}$ with $(x^{(1)}, z^{(2)}) \in R_j$ we let $h^{(2)}(z^{(2)}) = h((x^{(1)}, z^{(2)}))$. Finally for every $k \geq 0$ and every remaining bounded variable $z_k^{(i)} \in Z_k^{(i)}$ $(i = 1, 2)$ we define (nondeterministically) $h_k^{(i)}(z_k^{(i)}) \in \Sigma_k \setminus \left(\Sigma_k^{(1)} \cup \Sigma_k^{(2)} \cup h_k((R_j)_k) \cup (h_k(Y_k) \cap s(dom(s)))\right)$. Let s_1 and s_2 be the projections of s on $\Gamma^{(1)} \cup \Sigma^{(2)}$ and $\Gamma^{(2)} \cup \Sigma^{(1)}$, respectively, defined in the obvious way. By construction of the buta M_{R_j}, we get that $s_1 \in L\left(M^{(1)}_{(\Sigma^{(2)}, V_1)}\right)$ and $s_2 \in L\left(M^{(2)}_{(\Sigma^{(1)}, V_2)}\right)$. Moreover, by definition of $h^{(1)}$ and

$h^{(2)}$ we get $t \in h^{(1)}(s_1) \cap h^{(2)}(s_2)$, i.e., $t \in L\left(\mathcal{M}^{(1)}\right) \cap L\left(\mathcal{M}^{(2)}\right)$. We complete our proof by taking into account Proposition 3.

It is well-known (cf. [5]) that the *branches* of a recognizable tree language constitute a recognizable string language. Next, we show that this result fails when we consider recognizability over infinite alphabets. We need some preliminary matter. For variable finite automata considered in the sequel, we refer the reader to [10,11].

To every ranked alphabet Σ we assign a monadic alphabet $br(\Sigma)$ determined as follows:

- $br(\Sigma)_0 = \Sigma_0$, and
- $br(\Sigma)_1 = \{[\sigma, i] \mid \sigma \in \Sigma_k, k \geq 1 \text{ and } 1 \leq i \leq k\}$.

Then the branches of a tree $t \in T_\Sigma$ is the set of strings $br(t) \subseteq (br(\Sigma)_1)^* \, br(\Sigma)_0$ defined inductively by

- $br(t) = \{t\}$ if $t \in \Sigma_0$, and
- $br(t) = \bigcup_{1 \leq i \leq k} [\sigma, i](br(t_i))$ if $t = \sigma(t_1, \ldots, t_k)$ with $\sigma \in \Sigma_k$, $k \geq 1$, and $t_1, \ldots, t_k \in T_\Sigma$.

For a tree language $L \subseteq T_\Sigma$ we set $br(L) = \bigcup_{t \in L} br(t)$.

Now we are ready to prove our claim. Consider a letter $a \in \Sigma_0$ and the tree language $L = \{\sigma(\sigma(a,a), \sigma(a,a)) \mid \sigma \in \Sigma_2\}$. Then $L \in VREC(\Sigma)$. Indeed, L is accepted by a vbuta $\mathcal{M} = \langle \Sigma, M \rangle$ with $M = (Q, \Gamma_M, \Delta, F)$ where $\Gamma_M = \{a\} \cup Z \cup Y$, $z \in Z_2$, and $L(M) = \{z(z(a,a), z(a,a))\}$. Then $br(L) = \{[\sigma, i][\sigma, j]a \mid \sigma \in \Sigma_2,$ and $i, j = 1, 2\}$. Let us assume that $br(L)$ is a recognizable word language over the infinite alphabet $br(\Sigma)$ in the sense of [10,11]. Hence there is a variable finite automaton $\mathcal{A} = \langle br(\Sigma), A \rangle$ such that $br(L) = L(\mathcal{A})$. Since the letters $[\sigma, i], [\sigma, j]$, for $\sigma \in \Sigma_2$ and $i, j = 1, 2$, are infinitely many, we get that there is a word $x_1 x_2 a \in L(A)$ where x_1 and x_2 are both bounded variables with $x_1 \neq x_2$ or at least one of them is the free variable of A. This in turn implies that the words $[\sigma, i][\sigma', j]a \in L(\mathcal{A})$, where $\sigma, \sigma' \in \Sigma_2$ and $\sigma \neq \sigma'$, which is a contradiction.

Proposition 5. *The emptiness problem is decidable in $VREC(\Sigma)$.*

Proof. Let $L \in VREC(\Sigma)$. Then $L = L(\mathcal{M})$ where $\mathcal{M} = \langle \Sigma, M \rangle$ is a vbuta. Moreover, $L = \emptyset$ iff $L(M) = \emptyset$. Since the emptiness problem is decidable for recognizable tree languages (cf. [4]), we are done.

Our next task, is to show that the equivalence problem is decidable for two subclasses of vbuta. In fact, we will prove that the equivalence problem is decidable for vbuta whose transitions contain only one type of variables, bounded or free, but not both. Let $\mathcal{M}^{(i)} = \left\langle \Sigma, M^{(i)} \right\rangle$, $i = 1, 2$, be two vbuta where $M^{(i)} = (Q^{(i)}, \Gamma^{(i)}, \Delta^{(i)}, F^{(i)})$ with $\Gamma^{(i)} = \Sigma^{(i)} \cup Z^{(i)} \cup Y^{(i)}$. Without any loss, we assume the sets of variables $Z^{(1)}$, $Z^{(2)}$, $Y^{(1)}$, and $Y^{(2)}$ to be pairwise disjoint.

Case (i): Assume that the transitions of $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$ do not contain any free variable. Consider the vbuta $M^{(1)}_{(\Sigma^{(2)},V_1)} = \left(Q^{(1)}_{V_1}, \Gamma^{(1)} \cup \Sigma^{(2)}, \Delta^{(1)}_{V_1}, F^{(1)}_{V_1}\right)$ and $M^{(2)}_{(\Sigma^{(1)},V_2)} = \left(Q^{(2)}_{V_2}, \Gamma^{(2)} \cup \Sigma^{(1)}, \Delta^{(2)}_{V_2}, F^{(2)}_{V_2}\right)$ obtained, respectively, from $M^{(1)}$ and $M^{(2)}$ by the procedure described before Lemma 1. We consider the ranked alphabet $\Theta = (\Sigma^{(1)} \cup \Sigma^{(2)}) \cup (Z^{(1)} \times Z^{(2)})$ and the projection relabelings pr_i, $i = 1, 2$, from Θ to $(\Sigma^{(1)} \cup \Sigma^{(2)}) \cup Z^{(i)}$ defined in the obvious way. We set $L_1 = L\left(M^{(1)}_{(\Sigma^{(2)},V_1)}\right)$, $L_2 = \left(M^{(2)}_{(\Sigma^{(1)},V_2)}\right)$, $L_1' = pr_1^{-1}(L_1)$, and $L_2' = pr_2^{-1}(L_2)$. Let G be a maximal subalphabet of Θ satisfying the following condition. Every element of $Z^{(1)}$ (resp. $Z^{(2)}$) occurs as a left (resp. right) coordinate in at most one pair in G. Let $G_1, \dots, G_m$ be an enumeration of all such subalphabets of Θ. Finally, let $L = (L_1' \cap L_2') \cap (T_{G_1} \cup \dots \cup T_{G_m})$.

Lemma 2. *The following statements are equivalent*

(i) $pr_1(L) = L_1$ *and* $pr_2(L) = L_2$.
(ii) $L(\mathcal{M}^{(1)}) = L\left(\mathcal{M}^{(2)}\right)$.

Proof. Assume firstly that (i) holds and let $t \in L(\mathcal{M}^{(1)})$. Then, there is a tree $s_1 \in L_1$ and a valid relabeling $h^{(1)} \in VR(\Gamma^{(1)} \cup \Sigma^{(2)})$ such that $t \in h^{(1)}(s_1)$. Let $s \in L$ such that $pr_1(s) = s_1$ and let $pr_2(s) = s_2$. We have $dom(s_1) = dom(s) = dom(s_2)$. Let also $1 \leq j \leq m$ such that $s \in T_{G_j}$. We define a valid relabeling $h^{(2)} \in VR(\Gamma^{(2)} \cup \Sigma^{(1)})$ as follows. For every node $w \in dom(s)$ with $s(w) = (z^{(1)}, z^{(2)})$, $z^{(i)} \in Z^{(i)}$ $i = 1, 2$, we let $h^{(2)}(z^{(2)}) = t(w)$ (in fact, we have $t(w) = h^{(1)}(z^{(1)})$. It should be clear that such a valid relabeling exists by our assumption for the ranked alphabet G_j. Now we easily obtain that $t \in h^{(2)}(s_2)$ and since $s_2 \in L_2$, we get $t \in L(\mathcal{M}^{(2)})$. Thus $L(\mathcal{M}^{(1)}) \subseteq L(\mathcal{M}^{(2)})$. In a similar way, we show that $L(\mathcal{M}^{(2)}) \subseteq L(\mathcal{M}^{(1)})$, hence $L(\mathcal{M}^{(1)}) = L(\mathcal{M}^{(2)})$.

Assume next that (ii) holds. We will show that $pr_1(L) = L_1$ and $pr_2(L) = L_2$. Since $pr_1(L) \subseteq L_1$ and $pr_2(L) \subseteq L_2$, it remains to show the converse inclusions. For this, let $s_1 \in L_1$, and consider an $h^{(1)} \in VR(\Gamma^{(1)} \cup \Sigma^{(2)})$ and a tree $t \in h^{(1)}(s_1)$. Then $t \in L(\mathcal{M}^{(1)})$, hence $t \in L(\mathcal{M}^{(2)})$. This implies that there is a tree $s_2 \in L_2$ and a valid relabeling $h^{(2)} \in VR(\Gamma^{(2)} \cup \Sigma^{(1)})$ such that $t \in h^{(2)}(s_2)$. We claim that there exists a tree $s \in L_1' \cap L_2'$ such that $pr_1(s) = s_1$ and $pr_2(s) = s_2$. Indeed, if this is not the case, then there is a node $w \in dom(s_1) = dom(s_2)$ such that $s_1(w) \in \Sigma^{(1)} \cup \Sigma^{(2)}$ and $s_2(w) \in Z^{(2)}$ or $s_1(w) \in Z^{(1)}$ and $s_2(w) \in \Sigma^{(1)} \cup \Sigma^{(2)}$. But both of these conditions contradict our assumuption that $t \in h^{(1)}(s_1) \cap h^{(2)}(s_2)$. Now assume that $s \notin T_{G_1} \cup \dots \cup T_{G_m}$. This implies that there are two nodes $w, u \in dom(s)$ and bounded variables $z^{(1)}, \overline{z}^{(1)} \in Z^{(1)}$, $z^{(2)}, \overline{z}^{(2)} \in Z^{(2)}$ such that

(1) $s(w) = (z^{(1)}, z^{(2)})$ and $s(u) = (z^{(1)}, \overline{z}^{(2)})$, or
(2) $s(w) = (z^{(1)}, z^{(2)})$ and $s(u) = (\overline{z}^{(1)}, z^{(2)})$.

Assume that (1) holds. Then $s_1(w) = s_1(u) = z^{(1)}$, and $s_2(w) = z^{(2)}$, $s_2(u) = \overline{z}^{(2)}$ which in turn implies that $h^{(1)}(s_1(w)) = h^{(1)}(s_1(u))$ and $h^{(2)}(s_2(w)) \neq$

$h^{(2)}(s_2(u))$. But this is a contradiction since, by our assumption, $t \in h^{(1)}(s_1) \cap h^{(2)}(s_2)$. The case (2) contradicts to our assumption, similarly. Thus, we get $s \in L$ which implies $L_1 \subseteq pr_1(L)$. The inclusion $L_2 \subseteq pr_2(L)$ is shown by similar arguments, and our proof is completed.

Case (ii): Assume that the transitions of $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$ do not contain any bounded variable. Without any loss, we suppose that $\Sigma^{(1)} = \Sigma^{(2)}$. Indeed, if this not the case, then in the vbuta $\mathcal{M}^{(1)}$, for every $k \geq 0$ and every transition $((q_1, \ldots, q_k), y_k, q) \in \Delta_k^{(1)}$ we add the transition $((q_1, \ldots, q_k), \sigma, q)$ for every $\sigma \in (\Sigma^{(2)} \setminus \Sigma^{(1)})_k$; we make a similar modification on the vbuta $\mathcal{M}^{(2)}$. Now, we consider the ranked alphabet $\Theta = \Sigma^{(1)} \cup (Y^{(1)} \times Y^{(2)})$ and the projection relabelings pr_i, $i = 1, 2$, from Θ to $\Sigma^{(1)} \cup Y^{(i)}$ defined in the obvious way. We set $L_1 = L\left(M^{(1)}\right)$, $L_2 = \left(M^{(2)}\right)$, $L_1' = pr_1^{-1}(L_1)$, $L_2' = pr_2^{-1}(L_2)$, and $L' = L_1' \cap L_2'$.

Lemma 3. *The following statements are equivalent*

(i) $pr_1(L') = L_1$ *and* $pr_2(L') = L_2$.
(ii) $L(\mathcal{M}^{(1)}) = L\left(\mathcal{M}^{(2)}\right)$.

Proof. Assume firstly that (i) holds and let $t \in L(\mathcal{M}^{(1)})$. Then, there is a tree $s_1 \in L_1$ and a valid relabeling $h^{(1)}$ on $\Gamma^{(1)}$ such that $t \in h^{(1)}(s_1)$. Let $s \in L'$ such that $pr_1(s) = s_1$ and let $pr_2(s) = s_2$. We have $dom(s_1) = dom(s) = dom(s_2)$. We define a valid relabeling $h^{(2)}$ on $\Gamma^{(2)}$ as follows. For every node $w \in dom(s)$ with $s(w) = (y^{(1)}, y^{(2)})$, $y^{(i)} \in Y^{(i)}$ $i = 1, 2$, we let $h^{(2)}(y^{(2)}) = h^{(1)}(y^{(1)})$. Clearly, such a valid relabeling exists by our assumption that $\Sigma^{(1)} = \Sigma^{(2)}$. Trivially $t \in h^{(2)}(s_2)$ and since $s_2 \in L_2$, we get $t \in L(\mathcal{M}^{(2)})$. Thus $L(\mathcal{M}^{(1)}) \subseteq L(\mathcal{M}^{(2)})$. In a similar way, we show that $L(\mathcal{M}^{(2)}) \subseteq L(\mathcal{M}^{(1)})$, hence $L(\mathcal{M}^{(1)}) = L(\mathcal{M}^{(2)})$.

Next, assume that (ii) holds. We will show that $pr_1(L') = L_1$ and $pr_2(L') = L_2$. For this, let $s_1 \in L_1$, and consider an $h^{(1)} \in VR(\Gamma^{(1)})$ and a tree $t \in h^{(1)}(s_1)$. Then $t \in L(\mathcal{M}^{(1)})$, hence $t \in L(\mathcal{M}^{(2)})$. This implies that there is a tree $s_2 \in L_2$ and a valid relabeling $h^{(2)} \in VR(\Gamma^{(2)})$ such that $t \in h^{(2)}(s_2)$. We claim that there exists a tree $s \in L'$ such that $pr_1(s) = s_1$ and $pr_2(s) = s_2$. Indeed, if this is not the case, then there is a node $w \in dom(s_1) = dom(s_2)$ such that $s_1(w) \in \Sigma^{(1)}$ and $s_2(w) \in Y^{(2)}$ or $s_1(w) \in Y^{(1)}$ and $s_2(w) \in \Sigma^{(1)}$. But both of these conditions contradict our assumption that $t \in h^{(1)}(s_1) \cap h^{(2)}(s_2)$. Hence, $L_1 \subseteq pr_1^{-1}(L')$. The inclusion $L_2 \subseteq pr_2^{-1}(L')$ is shown similarly, and we are done.

Now we are ready to show the following important result.

Theorem 1. *The equivalence problem is decidable for*

(i) vbuta whose transitions do not contain any free variable
(ii) vbuta whose transitions do not contain any bounded variable.

Proof. (i) We follow the notations of Lemma 2. Let $\mathcal{M}^{(1)} = \left\langle \Sigma, M^{(1)} \right\rangle$ and $\mathcal{M}^{(2)} = \left\langle \Sigma, M^{(2)} \right\rangle$ be two vbuta. The tree languages (over finite ranked alphabets) L_1, L_2 are recognizable and thus L_1', L_2' are recognizable, too (cf. [4]).

Moreover, the tree language $T_{G_1} \cup \ldots \cup T_{G_m}$ is recognizable, hence L is recognizable. Finally, $pr_1(L), pr_2(L)$ are recognizable tree languages (cf. [4]), and since the equality of recognizable tree languages is decidable (cf. for instance Thm. 10.3, page 110, in [8]), the equalities $pr_1(L) = L_1$ and $pr_2(L) = L_2$ are decidable. Hence, by Lemma 2 we conclude our proof.

(ii) We use the notations of Lemma 3 and similar arguments with the proof of Statement (i) in this theorem.

Theorem 2. *The universality problem is decidable for vbuta whose transitions do not contain any bounded variable.*

Proof. We combine Theorem 1(ii) and Example 2.

As an application of Theorem 1, we will get the decidability of the equivalence in two subclasses of variable finite automata. For this, we need the subsequent discussion.

Let A be an infinite alphabet and e a new symbol not belonging to A. Then, by standard arguments on finite automata, we can show that a language L over the infinite alphabet A is recognizable iff the language Le over the infinite alphabet $A \cup \{e\}$ is recognizable. Furthermore, for two languages L_1, L_2 over A we have $L_1 = L_2$ iff $L_1 e = L_2 e$. Next, we define an infinite monadic ranked alphabet Σ, by $\Sigma_1 = A$ and $\Sigma_0 = \{e\}$. It should be clear that the string language Le is recognizable iff it is a recognizable tree language over Σ (cf. [7]). Hence, taking into account Theorem 1, we obtain the subsequent result.

Corollary 1. *The equivalence problem is decidable for*

(i) variable finite automata whose transitions do not contain any free variable
(ii) variable finite automata whose transitions do not contain any bounded variable.

Using the above formalism and a corresponding result in [10,11], we can show that the class $VREC(\Sigma)$ is not closed under complementation. Indeed assume the contrary, and let L be a recognizable language over the infinite alphabet A whose complement is not recognizable (cf. Thm. 3 in [11]). Then, keeping the above notations, Le is a recognizable word language and a recognizable tree language. Then, by our assumption the complement tree language $\overline{Le}$ is recognizable. Since our ranked alphabet has only one nullary symbol we get $\overline{Le} = \overline{L}e$. Hence, $\overline{L}$ is a recognizable word language, which is a contradiction.

Corollary 2. *The class $VREC(\Sigma)$ is not closed under complementation.*

4 Conclusions and Future Work

We introduced variable bottom-up tree automata accepting tree languages over infinite ranked alphabets. Our model works in the same way as variable finite automata which were investigated in [10,11]. It is based on a classical bottom-up tree automaton over a finite ranked alphabet which contains constant and variable symbols. The variable symbols are of two types, namely bounded and free. Firstly, we compute the tree language accepted by the automaton, and then we apply on the variable symbols relabelings satisfying certain requirements, the so-called valid relabelings. The image of the original recognizable tree language under all valid relabelings constitutes the tree language accepted by our new model. We proved that the class of recognizable tree languages over infinite ranked alphabets is closed under union and intersection, and the emptiness problem is decidable. We showed that the equivalence problem is decidable within two subclasses of vbuta. As a corollary, we obtained the decidability of the equivalence of two subclasses of variable finite automata of [10,11]. Still interesting problems remain open, like the determinization of vbuta, the closure under (inverse) tree homomorphisms and tree substitutions, and the existence of a pumping lemma. Especially, we conjecture that the equivalence of arbitrary vbuta is decidable. Our model, and accordingly the string model of [10,11], permits further characterizations of recognizable tree languages over infinite alphabets. For instance, instead of the underlying buta M one can consider an MSO-sentence, a regular tree grammar, a rational tree expression, or even a system of tree equations over the finite alphabet Γ_M, and then consider the image of the obtained tree language under the valid relabelings. In this way, we introduce MSO-definable, regular, rational, and equational tree languages respectively, over infinite ranked alphabets. Using the corresponding equivalences for tree languages over finite ranked alphabets (cf. [4,6,7,8,9,15,20], we get for a tree language $L \subseteq T_\Sigma$:

$$\begin{gathered} L \text{ is recognizable iff } L \text{ is } MSO\text{-definable iff } L \text{ is regular} \\ \text{iff } L \text{ is rational iff } L \text{ is } OI\text{-equational.} \end{gathered}$$

Furthermore, our model allows the investigation of other classes of tree languages over infinite ranked alphabets. More precisely, we can define synchronized or context-free tree languages if the underlying tree automaton is synchronized (cf. [18]) or pushdown (cf. [12]), respectively. Another important extension is the study of recognizability for infinitary tree languages over infinite ranked alphabets. This is done in a future paper.

References

1. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on words with data. In: Proceedings of the Symposium on Logic in Computer Science (LICS 2006), pp. 7–16. IEEE Computer Society Press, Los Alamitos (2006)
2. Bojańczyk, M., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and xml reasoning. J. ACM 56, 1–48 (2009)
3. Carme, J., Niehren, J., Tommasi, M.: Querying unranked trees with stepwise tree automata. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 105–118. Springer, Heidelberg (2004)
4. Comon, H., Dauchet, M., Gilleron, R., Jacquema, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications, http://www.grappa.univ-lille3.fr/tata

5. Damm, W.: The IO and OI-hierarchies. Theoret. Comput. Sci. 20, 95–207 (1982)
6. Doner, J.: Tree acceptors and some of their applications. J. Comput. System Sci. 4, 406–451 (1970)
7. Engelfriet, J.: Tree automata and tree grammars. DAIMI FN-10 (Lecture Notes), Aarhus University (1975)
8. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
9. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. III, pp. 1–68. Springer, Heidelberg (1997)
10. Grumberg, O., Kupferman, O., Sheinvald, S.: Variable automata over infinite alphabets. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 561–572. Springer, Heidelberg (2010)
11. Grumberg, O., Kupferman, O., Sheinvald, S.: Variable automata over infinite alphabets, `http://www.cs.huji.ac.il/\char126\relaxornak/publications/lata10.pdf`
12. Guessarian, I.: Pushdown tree automata. Math. Systems Theory 16, 237–263 (1983)
13. Jurdziński, M., Lazić, R.: Alternation-free modal mu-calculus for data trees. In: Proceedings of LICS 2007, pp. 131–140. IEEE Computer Society Press, Los Alamitos (2007)
14. Kaminski, M., Francez, N.: Finite-memory automata. Theoret. Comput. Sci. 134, 329–363 (1994)
15. Mezei, J., Wright, J.B.: Algebraic automata and context-free sets. Inform. Control 11, 3–29 (1967)
16. Neven, F., Schwentick, T., Vianu, V.: Towards regular languages over infinite alphabets. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 560–572. Springer, Heidelberg (2001)
17. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. 5, 403–435 (2004)
18. Salomaa, K.: Synchronized tree automata. Theoret. Comput. Sci. 127, 25–51 (1994)
19. Shemesh, Y., Francez, N.: Finite-state unification automata and relational languages. Infom. and Comput. 114, 192–213 (1994)
20. Thatcher, J.W., Wright, J.B.: Generalized finite automata theory with application to a decision problem of second-order logic. Math. Systems Theory 2, 57–81 (1968)

Author Index