Computational Complexity Questions Related to Finite Monoids and Semigroups

Pascal Tesson

School of Computer Science
McGill University, Montreal
February 2003

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Ph.D. Science.

Abstract

In this thesis, we address a number of issues pertaining to the computational power of monoids and semigroups as machines and to the computational complexity of problems whose difficulty is parametrized by an underlying semigroup or monoid and find that these two axes of research are deeply intertwined.

We first consider the "program over monoid" model of D. Barrington and D. Thérien [BT88] and set out to answer two fundamental questions: which monoids are rich enough to recognize arbitrary languages via programs of arbitrary length, and which monoids are so weak that any program over them has an equivalent of polynomial length? We find evidence that the two notions are dual and in particular prove that every monoid in **DS** has exactly one of these two properties. We also prove that for certain "weak" varieties of monoids, programs can only recognize those languages with a "neutral letter" that can be recognized via morphisms over that variety.

We then build an algebraic approach to communication complexity, a field which has been of great importance in the study of small complexity classes. Specifically, we consider the amount of communication that two players, Alice and Bob, need to exchange in order to compute the product $m_1 m_2 \ldots m_n$ of n elements of some fixed finite monoid M when Alice knows only the odd-indexed m_i and Bob knows the even-indexed m_i. We prove that every monoid has communication complexity $O(1)$, $\Theta(\log n)$ or $\Theta(n)$ in this model. We obtain similar classifications for the communication complexity of finite monoids in the probabilistic, simultaneous, probabilistic simultaneous and MOD_p-counting variants

of this two-party model and thus characterize the communication complexity (in a worst-case partition sense) of every regular language in these five models. Furthermore, we study the same questions in the Chandra-Furst-Lipton multi-party extension of the classical communication model and describe the variety of monoids which have bounded 3-party communication complexity and bounded k-party communication complexity for some k. We also show how these bounds can be used to establish computational limitations of programs over certain classes of monoids.

Finally, we consider the computational complexity of testing if an equation or a system of equations over some fixed finite monoid (or semigroup) has a solution. In the case of a single equation we extend the work of [GR99] by providing strong evidence that this problem cannot be resolved without answering questions about the expressive power of programs over that monoid. Most notably, we give a quasipolynomial-time upper bound for solving equations over a group which is known to require programs of exponential length in order to compute AND. We also give a number of upper bounds and hardness results for solving equations over monoids which are not groups and show that, in apparent contrast with the group case, the problem can be NP-complete over some M even if it is tractable over some N admitting M as a submonoid.

We find that testing the satisfiability of a system of equations over a finite monoid is either tractable or NP-complete depending on whether the monoid belongs to the class $\mathbf{J_1} \vee \mathbf{Ab}$ or not. For the restricted case when the right-hand side of the equations are constants, we show that a similar dichotomy holds for monoids and for regular semigroups. We also give a number of partial results for the general case of semigroups and relate this question with constraint-satisfaction problems.

Résumé

Nous étudions dans cette thèse des questions liées à la puissance de calcul des monoïdes et des semigroupes, lorsqu'ils sont considérés comme des machines, et à la complexité de problèmes dont la difficulté est paramétrisée par un semigroupe. Ces deux axes de recherche sont en fait intimement reliés.

Nous revoyons tout d'abord la notion de "programmes sur monoïdes" formalisée par D. Barrington et D. Thérien et tentons de répondre à deux questions fondamentales à propos de ce modèle: quels sont les monoïdes assez riches pour permettre la reconnaissance de langages arbitraires grâce à des programmes de longueur arbitraire et quels sont les monoïdes si faibles que tous leurs programmes ont un équivalent de longueur polynomiale? Nos résultats semblent indiquer que ces deux propriétés sont duales et démontrons qu'en particulier, tout monoïde dans la variété **DS** possède exactement l'une de ces propriétés. Nous démontrons également que pour certaines variétés, les programmes ne peuvent reconnaître un langage contenant une "lettre neutre" que si ce langage peut être reconnu grâce à un morphisme sur un monoïde de cette variété.

Nous développons ensuite une approche algébrique à la complexité de communication, un domaine d'une grande importance dans létude des petites classes de complexité. Nous étudions la quantité de communication que deux joueurs, Alice et Betrand, se doivent d'échanger pour calculer le produit de n éléments $m_1 m_2 \dots m_n$ d'un monoïde M lorsqu'Alice ne connaît que les m_i où i est pair et que Bertrand ne connaît que les m_i où i est impair. Nous montrons que tout monoïde a une complexité de communication $O(1)$, $\Theta(\log n)$ ou $\Theta(n)$ dans ce

modèle. Nous obtenons des classifications similaires dans les variantes probabiliste, simultanée, simultanée probabiliste et MOD_p de ce modèle et caractérisons ainsi la complexité de communication (par rapport à une partition pire-cas) de tous les langages réguliers. Nous étudions également ces questions dans l'extension multipartite du modèle à deux joueurs et obtenons une caractérisation des monoïdes ayant une complexité bornée pour le modèle à trois joueurs et pour le modèle a k joueurs pour un certain k. Nous montrons également comment ces résultats permettent d'établir les limites calculatoires des programmes sur certaines variétés de monoïdes.

Enfin, nous étudions la complexité de déterminer l'existence d'une solution à une équation ou à un système d'équations sur un monoïde (ou un semigroupe) donné. Dans le cas des équations, nos résultats complètent ceux de [GR99] et suggèrent fortement que la question ne peut être résolue sans comprendre les limites calculatoires des programmes sur ce monoïde. En particulier, nous décrivons un algorithme permettant de résoudre en temps quasi-polynomial une équation sur un groupe pour lequel les programmes calculant la fonction AND nécéssitent une longueur exponentielle. Nous établissons aussi quelques bornes inférieures et supérieures de la complexité de ce problème lorsque les équations sont sur un monoïde qui n'est pas un groupe. Contrairement au cas des groupes, nous montrons qu'il existe un M pour lequel ce problème est NP-complet bien qu'il soit calculable en temps polynomial pour un N dont M est un sous-monoïde.

Nous établissons aussi que le problème de la satisfaisabilité des systèmes d'équations est soit résoluble en temps polynomial sur un monoïde fini de la variété $\mathbf{J_1} \vee \mathbf{Ab}$ mais est NP-complet autrement. Nous démontrons une dichotomie semblable lorsque les moitiés droites de chaque équation ne sont que des constantes et plusieurs résultats dans le cas plus général des semigroupes. Nous relions aussi ces problèmes aux problèmes de satisfaisabilité de contraintes.

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my Ph.D. supervisor Denis Thérien. His enthusiasm, generosity and humanity have kept me motivated and eager to learn from his teaching. He has supported me financially over the course of my graduate studies at McGill and I am grateful for his confidence in my work. Above all, I want to thank him for supporting me through difficult times and periodically reminding me that these are "the best years of my life" (whether I like it or not).

I have been very fortunate to meet a number of Denis' collaborators over the last few years and I have greatly benefited from their experience, their insights and their willingness to patiently share them with me. In particular, the work in this thesis is for the most part based on papers coauthored by David Barrington, Ricard Gavaldà, Ondřej Klíma, Pierre McKenzie, Cris Moore and Jean-François Raymond (as well as Denis) [RTT98, BMM$^+$00, MTT01, GTT01, TT02a, TT03, KTT03] and it has been a great pleasure for me to work with each of them. I am also indebted to Stéphane Demri, Peter Kadau, Klaus-Jörn Lange, Clemens Lautemann, François Lemieux, Jean-Eric Pin, Pavel Pùdlak, Klaus Reinhardt, Alex Russell, Benjamin Steinberg and Howard Straubing for many helpful discussions that helped shape my understanding of the field.

I am very grateful to Klaus-Jörn Lange for welcoming me in Tübingen, and for generously sharing tea, beer and many ideas. My stay in Tübingen would not have been the same without the help, support and most importantly friendship of Peter, Jens, Klaus-Jörn, Klaus, Lyne and Denis (and their children!).

I feel privileged to have been a part of McGill's computer science department. I have greatly appreciated the administrative and academic staff's dedication and patience. I also want to thank my office mates Arkadev Chattopadhyay, Sylvie Hamel and Mark Mercer for their advice and support during the writing of my thesis.

There have been many times when I have needed the help of close friends and family and I would like to particularly thank my parents, my sister, Xavier, Véronique, Francis, Guillaume and Olivier for their kindness and support.

Last but not least, and from the bottom of my heart, I want to thank Marie-Claude for her love, her patience, her advice and her unwavering support throughout the past four years.

Contents

Chapter 1

Introduction

1.1 Finite Semigroups, Automata and Regular Languages

Despite their well-known limitations, finite automata have always been and will remain a fundamental model of computation and a starting point for investigations in theoretical aspects of computer science. Their study has also been motivated by applications to pattern matching, modeling of finite state systems and important links with logic and algebra.

It was noticed early on that algebra provided a most powerful framework to analyze and classify regular languages according to their combinatorial properties. The earlier results included the algebraic characterization of *star-free* languages proved by M. P. Schützenberger [Sch65] and the characterization of *piecewise testable* languages given by I. Simon [Sim75] before S. Eilenberg established a one-to-one correspondence between *varieties of semigroups* (classes of semigroups closed under direct product, morphic images and subsemigroups) and *varieties of languages* (classes of languages closed under quotients, Boolean operations and inverse morphisms from free semigroup to free semigroup), thus providing the precise framework for algebraic automata theory. Finding explicit algebraic descriptions of language varieties for which we are given a combinatorial description and, conversely, finding combinatorial descriptions of language varieties corresponding to natural algebraic varieties has led to important advances in both algebra and language theory. In fact, these elegant methods have been so successful that it is difficult to separate today automata theory from finite semigroup theory: their influence can be seen for instance on recent results concerning logic and in particular temporal logic [Str94, TW02b, TW98, TW02a, BMT99] and they are robust enough to be adapted to offer nice algebraic approaches to Büchi automata [PP03] and timed-automata [BPT01].

1.2 Monoids as Machines

Let us view a finite monoid M as a machine whose sole ability is to compute the product of a list of elements of M. How can we use this machine to recognize languages in, say, A^*? The classical mechanism is that of a morphism: each input letter is translated into a monoid element through some predetermined mapping $\phi : A \to M$ and the input is accepted if the product of these elements lies in some target set $F \subseteq M$. It is easy to see that this captures exactly the regular languages and this observation is the starting point of classical algebraic automata theory.

As pictured in Figure 1.1, we can consider more elaborate ways to use a monoid M as a language recognizer. We construct a machine that first preprocesses the input in A^* in some predetermined way thus translating it into a sequence of monoid elements. Our machine then accepts its input if the multiplication of these elements belongs to some previously chosen accepting subset of M. Clearly, the power of such machines depends both on the nature of this preprocessing and on the particular monoid used in the later step. Amazingly, well-known complexity classes can be characterized in this way.

The first such example stems from the "program over monoid" formalism introduced by D. Barrington and D. Thérien in the mid 80's: in this case, the

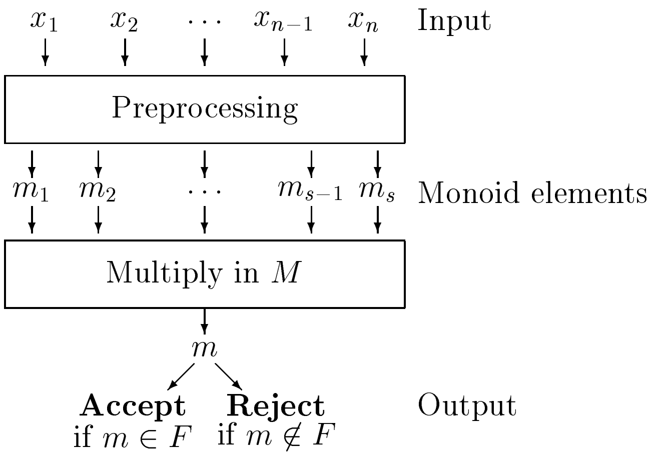


Figure 1.1: Monoids as machines

preprocessing produces a polynomial number of monoid elements, each of which is a function of exactly one input position. It was first shown that a language can be recognized by a polynomial length program over a monoid if and only if it belongs to the class NC^1 of languages recognizable by families of logarithmic depth Boolean circuits [Bar89]. Subsequently, algebraic characterizations of well-known subclasses of NC^1 were obtained when the underlying monoid was restricted to belong to some important varieties [BT88].

A few years later, similar techniques were used to show that using a preprocessing performed by a polynomial-time machine allowed an algebraic characterization of PSPACE [HLS$^+$93].

Such characterizations are interesting for different reasons. First of all, they automatically yield a new point of view on the corresponding complexity classes and give one the opportunity to use tools developed in algebraic automata theory to investigate the properties of the classes (see e.g. [MPT91]). They also suggest an interesting way of studying the structure of this class by examining the computational power of these machines when the monoid belongs to restricted classes. In the two examples just cited, many of the best known subclasses of NC^1 and PSPACE can be put in correspondence with well-studied varieties of monoids. Even more importantly, varieties allow one to algebraically define very fine parametrizations of the complexity classes.

In turn, algebraic characterizations of complexity classes underline the importance of questions about computational problems whose complexity is parametrized by an underlying finite semigroup or monoid such as the membership problem [Koz77, BLS87, BMT92] and some of its variants [BKLM01], equation satisfiability [GR99], monoidal circuit evaluation [BMPT97], learning an expression over a monoid [GTT01] among others. In many cases, questions about the complexity of these problems and questions about the computational limits of semigroups as language recognizers are closely linked and sometimes inseparable.

As we try to understand the computational power of monoids as machines (in various formalisms) and the computational complexity of algorithmic problems about monoids, we are thus simultaneously building an algebraic point of view on computation and a computational point of view on algebra. While for the most part, tools from algebra have resulted in advances in complexity theory, it is also the case that complexity questions have motivated advances in semigroup theory.

1.3 Our contributions

The work presented in this thesis is a contribution to this algebraic point of view on computational complexity. We prove a number of new results about the computational power of programs over monoids and explore new areas in which the semigroup/monoid approach is meaningful. We relate the results obtained in these different contexts with one another and with existing work.

We present in Chapter 2 the main tools and results from semigroup theory and algebraic automata theory which will be used in later chapters. In particular, we list varieties of monoids and semigroups which bear particular importance to our work and to many similar investigations. We also recall some basic notions of computational complexity theory, introduce Boolean circuit models, branching programs and briefly survey the current state of research in circuit complexity to outline some of the major open questions in the field. Chapters 3, 4, 5 form the bulk of our work; we have tried to make each of them as self-contained as possible although an interesting feature of our results is that some classes of monoids and semigroups play key roles in apparently unrelated problems.

1.3.1 Programs over Monoids

We begin Chapter 3 by reviewing the "program over monoid" formalism and its deep running link to Boolean circuits of shallow depth (and bounded-width branching programs). We then prove that some monoids are so weak as machines that *any* computation they can perform via programs can actually be achieved with programs of polynomial length. On the other hand, some monoids are rich enough that they can, via programs, recognize arbitrary languages provided that no restriction on program length is imposed. Surprisingly, we find some evidence that these two properties are dual and show that in the variety **DS** every monoid either has the above *polynomial length property* or is *universal*. We also present a number of results for monoids outside this class and argue in favor of a conjecture which would generalize the dichotomy observed in **DS**.

In order to understand the computational power of programs over given varieties of monoids, it is crucial to isolate so-called program-varieties $\mathbf{V}$, i.e. varieties such that any regular language which contains a so-called *neutral letter* and can be recognized by programs of polynomial length over some $M \in \mathbf{V}$ can in fact be recognized by some $N \in \mathbf{V}$ but using the more primitive notion of recognition via morphism. We show that for some varieties an even stronger statement is true: we say that $\mathbf{V}$ has the *Crane Beach property* if any language with a neutral letter that can be recognized by programs of polynomial length over some $M \in \mathbf{V}$ is in fact regular and can be recognized, via morphism, by some $N \in \mathbf{V}$. We show in particular that commutative monoids and *$\mathcal{J}$-trivial* monoids have this property while it has been shown not to hold for *aperiodic monoids* [BIL$^+$01].

1.3.2 Communication Complexity

In Chapter 4, we propose an algebraic approach to communication complexity, a field which, over the last twenty years, has been at the heart of many investigations in complexity theory [KN97], most significantly in the study of shallow Boolean circuits and branching programs.

We look at the amount of communication that k parties need to exchange in order to evaluate the product of n elements of a finite monoid M when the access to the inputs is distributed among the different parties in the worst possible way. We prove that the *two-party communication complexity of a finite monoid* is either constant, $\Theta(\log n)$ or $\Theta(n)$ in the standard two-party deterministic model of A. Yao [Yao79] and give algebraic descriptions of all three cases. We obtain similar classifications for the two-party simultaneous, probabilistic, probabilistic simultaneous and Mod_p-counting communication complexity of a finite monoid. As a corollary, we are able to give, up to a constant, the communication complexity, in a worst-case partition sense, of *any* regular language in all five of these models. Some of our results highlight and explain the central importance of certain regular languages in communication complexity theory.

We also look at the communication complexity of regular languages and monoids in the multiparty model of A. Chandra, M. Furst and R. Lipton [CFL83]. We prove algebraic characterizations for monoids and regular languages which have bounded 3-party communication complexity and those which have bounded k-party communication complexity for some fixed k. Our algebraic approach isolates natural examples of languages for which precise multiparty communication complexity bounds would constitute fundamental progress in our understanding of this tricky model. We are also led to conjecture a multiparty generalization of Szegedy's algebraic characterization of languages with bounded two-party communication complexity.

We apply our communication complexity results to identify program-varieties and to obtain length lower bounds for programs computing some explicit function over certain classes of monoids. While most of the lower bounds are corollaries or only slight improvements of previously known results, our techniques are quite different.

1.3.3 Equations over Semigroups

In Chapter 5, we try to understand how the algebraic structure of a finite monoid or semigroup affects the complexity of solving equations over that fixed semigroup. Our work complements the results of M. Goldmann and A. Russell who had obtained results in the group case [GR99].

We first look at the complexity of testing if a given equation over the monoid M:

$$c_0 x_{i_1} c_1 \dots x_{i_k} c_k = d_0 x_{j_1} d_1 \dots x_{j_t} d_t$$

where $c_i, d_i \in M$ are constants and x's are variables, can be satisfied. That is if variables can be assigned values in M so that the right-hand and left-hand side of the equation multiply out to the same value in M. This problem, denoted EQN_M, had been shown NP-complete for non-solvable groups and in P for nilpotent groups. The latter upper bound was in fact obtained for the related problem P-SAT_M of testing whether a given M-program has some input on which it outputs a specified target. We prove that the complexity of P-SAT_G and EQN_G when the underlying monoid is a solvable but non-nilpotent groups G is tightly connected to well-known open problems on the expressivity of bounded-depth modular circuits: in particular, we obtain a quasi-polynomial time upper bound for both P-SAT$_G$ and EQN$_G$ when exponential lower bounds are known on the length of G-programs computing AND. When the underlying M is aperiodic, we surprisingly show that solving an equation over M is in some cases easier than solving an equation over some *divisor* N of M and that P-SAT$_M$ can be strictly harder than EQN$_M$. We further prove that P-SAT$_M$ lies in the very simple complexity class AC^0 if M lies in the variety $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ but is NP-complete for any aperiodic monoid not lying in this class.

We also look at the complexity of EQN$^*_M$, the problem of testing the satisfiability of a system of equations over M (or more generally of a semigroup S) and also consider a restricted version of the problem T-EQN$^*_M$ when the right-hand side of each equation contains no variables. We prove sharp dichotomies for the complexity of both problems which depend on the algebraic properties of the underlying monoid. We show that EQN$^*_M$ lies in P if M is a monoid in $\mathbf{J_1} \vee \mathbf{Ab}$ and is NP-complete otherwise and similarly show that T-EQN$^*_M$ lies in P if M is a monoid in $\mathbf{RB} \vee \mathbf{Ab}$ and is NP-complete otherwise. We also consider the case of systems over a *regular semigroup* and in particular obtain a similar dichotomy for the complexity of T-EQN$^*_S$. We also establish an intriguing connection between our methods and universal algebra methods used in the study of *constraint satisfaction problems*.

Chapter 2

Background

This chapter gives a quick technical introduction to algebraic automata theory and complexity theory which are the bases of our discussion. We review a number of definitions and important results in the field as well as set notation. We assume that the reader is familiar with the basic notions of relations, congruences, morphisms, solvable groups and Time/Space complexity of a Turing machine.

2.1 Algebraic Automata Theory

2.1.1 Semigroups and Automata

The theory of finite semigroups and its applications to formal languages have been the subject of extensive work since the 50's. We suggest as reference the book of J.E. Pin [Pin86] and his more recent comprehensive survey on syntactic semigroups [Pin97] although some of the more technical results can only be found in less accessible books such as [Eil76] or [How76].

A *semigroup* is a set S together with a binary, associative operation (which we usually denote multiplicatively). We further say that S is a *monoid* if there exists an *identity element* 1_S in S such that $1_S \cdot t = t \cdot 1_S = t$ for all $t \in S$. The multiplication of a semigroup defines a canonical surjective morphism $eval_S$:

$S^* \to S$ by

$$eval_S(s_1 s_2 \ldots s_k) = s_1 \cdot s_2 \cdot \ldots \cdot s_k.$$

We will sometimes refer to the languages in S^+ of the form $\{w | eval_S(w) \in F\}$ for some $F \subseteq S$ as the *word problems* of S. In the case of monoids, $eval_M$ is defined as a function from M^* to M and, similarly, word problems are subsets of M^*.

With the exceptions of the free semigroup A^+ and the free monoid A^*, all semigroups considered in this thesis will be finite and in the rest of this Chapter, S and M will respectively denote a finite semigroup and a finite monoid.

We want to view finite semigroups as language recognizers akin to finite automata (see [Sip97, HU79]). Formally, we say that a language $L \subseteq A^+$ (resp. $L \subseteq A^*$) can be *recognized* by the semigroup S (resp. the monoid M) if there exists a morphism $\phi : A^+ \to S$ (resp. $\phi : A^* \to M$) and an *accepting subset* $T \subseteq S$ (resp. $T \subseteq M$) such that $L = \phi^{-1}(T)$.

The algebraic theory of automata and regular languages is affected, sometimes at quite a deep level, by whether languages are defined to be subsets of the free monoid A^* (finite words over the alphabet A *including* the empty word) or subsets of the free semigroup A^+ (finite words over the alphabet A *excluding* the empty word). Because of this, two parallel theories presenting only slight, but occasionally crucial, differences have to be constructed. This is only a concern in a few occasions in this work and we will for the most part try to avoid the problem. In particular, many of the definitions stated below cover the monoid case although the reader should keep in mind that distinctions with the semigroup case might exist.

For a finite automaton $\mathcal{M}$ with state set Q, every word $w \in A^*$ defines a transformation $Q \to Q$. This set of mappings forms a monoid (under composition) which we call $\mathcal{M}$'s *transformation monoid.* One can easily show that any language recognized by $\mathcal{M}$ can be recognized by $\mathcal{M}$'s transformation monoid.

Moreover, the Cayley graph of a finite monoid M can be viewed as a finite automaton, allowing one to prove:

Proposition 2.1 *A language $L \subseteq A^*$ can be recognized by a finite automaton if and only if it can be recognized by a finite monoid if and only if it is regular.*

The last part of this statement is a trivial reformulation of Kleene's Theorem. The main objective of algebraic automata theory is to refine Kleene's Theorem: once it is established that languages recognized by finite monoids have a nice combinatorial characterization (i.e. they can be described by regular expressions) it is natural to ask whether subclasses of regular languages can be put in similar correspondence with subclasses of monoids.

2.1.2 The Variety Theorem

We say that a monoid N *divides* M and write $N \prec M$ if there exists a surjective morphism from a submonoid T of M onto N. It is easy to check that $\prec$ is a well-defined partial order (up to isomorphism) on finite monoids and that any language that can be recognized by N can also be recognized by any M with $N \prec M$. A class $\mathbf{V}$ of monoids is a (pseudo)-*variety*[1] if it is closed under direct product and division.

For $L \subseteq A^*$, we define the *syntactic congruence*, denoted $\equiv_L$, by letting $x \equiv_L y$ if and only if for all $u, v \in A^*$ we have $uxv \in L$ if and only if $uyv \in L$. The *syntactic monoid* of L, denoted $M(L)$ is $A^*/\equiv_L$. One can think of $M(L)$ as the "minimal recognizer" of L since it is not hard to show that $M(L)$ itself recognizes L and divides any other monoid that also recognizes L. It should be noted also that the construction of $M(L)$ is very similar to the automaton minimization process *à la* Myhill-Nerode. Of course, $M(L)$ is finite if and only if L is regular.

[1]Strictly speaking, a variety is a class of monoids closed under arbitrary direct product whereas pseudo-varieties only require closure under finite direct product. Because we only look at classes of finite monoids, we will ignore this distinction.

For $u \in A^*$ a string and $L \subseteq A^*$ a language, the right (resp. left) *quotient* of L by u is the set $Lu^{-1} = \{w : w \in A^* \text{ and } wu \in L\}$ (resp. $u^{-1}L = \{w : w \in A^* \text{ and } uw \in L\}$). A class $\mathcal{L}$ of languages is a *language-variety* if and only if it is closed under Boolean operations, left and right quotients and inverse morphisms from one free monoid to another (i.e. if $L \in A^*$ is in $\mathcal{L}$ and $\phi : B^* \to A^*$ is a morphism, then $\phi^{-1}(L)$ is also in $\mathcal{L}$).

Eilenberg's variety theorem links varieties of monoids and varieties of languages:

Theorem 2.2 (Variety Theorem) *There is a natural bijection between varieties of languages and varieties of monoids: if* $\mathbf{V}$ *is a class of monoids and* $\mathcal{L}$ *is the class of languages over any finite alphabet that are recognized by a monoid in* $\mathbf{V}$ *then* $\mathbf{V}$ *is a variety of monoids only if* $\mathcal{L}$ *is a variety of languages and is, in this case, generated by the set* $\{M(L) : L \in \mathcal{L}\}$.

Varieties are thus the natural unit to classify monoids in terms of their computational power and one can hope to make explicit the correspondence between an algebraic description of a variety $\mathbf{V}$ and a combinatorial description of languages in the associated language-variety. We will give many examples of such results.

2.1.3 The Structure of Finite Semigroups

For any monoid M, we introduce five equivalence relations known as *Green's relations* which describe whether two elements generate the same ideals in M. Formally:

- $x \mathcal{J} y$ if $MxM = MyM$;
- $x \,\mathcal{L}\, y$ if $Mx = My$;
- $x \,\mathcal{R}\, y$ if $xM = yM$;

- $x\,\mathcal{H}\,y$ if both $x\,\mathcal{R}\,y$ and $x\,\mathcal{L}\,y$;
- $x\mathcal{D}y$ if $x\,\mathcal{R}\circ\mathcal{L}\,y$, that is there exists z such that $x\,\mathcal{R}\,z$ and $z\,\mathcal{L}\,y$.

In a semigroup S, Green's relations are defined using ideals in S^1, the monoid obtained from S by adding an identity element if there is none in S. It can be shown that $\mathcal{R}$ is a left-congruence (i.e. $x\,\mathcal{R}\,y$ implies $cx\,\mathcal{R}\,cy$ for all c) and that $\mathcal{L}$ is a right-congruence. Moreover $\mathcal{R}$ and $\mathcal{L}$ commute (i.e. $\mathcal{D}=\mathcal{R}\circ\mathcal{L}=\mathcal{L}\circ\mathcal{R}$) and so all five of these relations are indeed equivalence relations. Moreover, the relations $\mathcal{J}$ and $\mathcal{D}$ coincide for any *finite* S. Since we are only interested in the structure of finite semigroups, we will consequently always refer to the $\mathcal{J}$-relation but the reader should be aware that some of the results stated below do not hold for infinite monoids in which $\mathcal{D}\neq\mathcal{J}$.

For an element x of M, we denote by $\mathcal{J}_x$ (resp. $\mathcal{R}_x$, $\mathcal{L}_x$, $\mathcal{H}_x$) the $\mathcal{J}$-class (resp. $\mathcal{R}$-, $\mathcal{L}$-, $\mathcal{H}$-class) of x. We also define natural pre-orders $\leq_{\mathcal{J}}$, $\leq_{\mathcal{R}}$, $\leq_{\mathcal{L}}$ on M with e.g. $x\leq_{\mathcal{J}} y$ if and only if $MxM\subseteq MyM$. We will say that "x is (strictly) $\mathcal{J}$-above y" if $x\geq_{\mathcal{J}} y$ (resp. $x>_{\mathcal{J}} y$), and similarly for $\leq_{\mathcal{R}}$ and $\leq_{\mathcal{L}}$. Note that $x\leq_{\mathcal{J}} y$ if and only if there exists u,v such that $x=uyv$. Similarly, $x\leq_{\mathcal{R}} y$ if and only if there is u with $x=yu$ and $x\leq_{\mathcal{L}} y$ if and only if there is u with $x=uy$. One can easily prove:

Lemma 2.3 *For any a,b in M, if $a\leq_{\mathcal{J}} ab$ (resp. $a\leq_{\mathcal{J}} ba$) then $a\,\mathcal{R}\,ab$ (resp. $a\,\mathcal{L}\,ab$).*
For any $a\,\mathcal{J}\,b$, if $a\leq_{\mathcal{R}} b$ (resp. $a\leq_{\mathcal{L}} b$) then in fact $a\,\mathcal{R}\,b$ (resp. $a\,\mathcal{L}\,b$).

The following lemma is the fundamental result about Green's relations:

Lemma 2.4 (Green's Lemma) *Suppose a and b are two elements of the same $\mathcal{R}$-class, i.e. there exist u,v s.t. $au=b$ and $bv=a$. Denote by $\rho_u: M\to M$ the function defined by $\rho_u(s)=su$. Then ρ_u and ρ_v are bijections from $\mathcal{L}_a$ to $\mathcal{L}_b$ and from $\mathcal{L}_b$ to $\mathcal{L}_a$ respectively.*

Moreover $\rho_u=\rho_v^{-1}$ and they preserve $\mathcal{H}$-classes.

The basic properties of Green's relations lead to the so-called "egg-box" representation of (finite) semigroups. Each $\mathcal{J}$-class of the semigroup is represented as a table where rows correspond to $\mathcal{R}$-classes, columns to $\mathcal{L}$-classes and cells to $\mathcal{H}$-classes. From Green's Lemma, we also know that each cell contains the same number of elements. When writing out the egg-box representation, the $\mathcal{J}$-classes are often laid out with respect to the $\leq_{\mathcal{J}}$ preorder (see for example Figure 2.1).

We say that $e \in S$ is *idempotent* if $e^2 = e$. Idempotents play an important role in the structure of semigroups. In particular, the identity element 1_M is an idempotent of M. We say that S has a *zero* is there is an element $0 \in S$ such that $0s = s0 = 0$ for all $s \in S$. Note that 0 is idempotent. We state two easy lemmas which further stress the importance of idempotents:

Lemma 2.5 *Let $e = e^2$ be an idempotent. $a \leq_{\mathcal{R}} e$ if and only if $ea = a$. Similarly $a \leq_{\mathcal{L}} e$ if and only if $ae = a$.*

Lemma 2.6 *Let $a \mathcal{J} b$. Then $ab \in \mathcal{R}_a \cap \mathcal{L}_b$ if and only if $\mathcal{L}_a \cap \mathcal{R}_b$ contains an idempotent $e = e^2$.*

We include here a proof of this simple but very useful fact to give an example of arguments using basic properties of Green's Lemma.

Proof. If there is an idempotent e in $\mathcal{L}_a \cap \mathcal{R}_b$, we have $ae = a$, $eb = b$. By Green's lemma $aeb \in \mathcal{R}_a \cap \mathcal{L}_b$ and $aeb = (ae)(eb) = ab$.

Conversely, if $ab \in \mathcal{R}_a \cap \mathcal{L}_b$, then, by Green's lemma, there is an f in $\mathcal{L}_a \cap \mathcal{R}_b$ such that $fb = b$. By lemma 2.5, since $b \leq_R f$, f must be idempotent. $\square$

The subsemigroup generated by an element s of S is finite of course, so there must exist t, p such that $s^{t+p} = s^t$ and the subsemigroup can be shown to have a unique idempotent. We will denote by ω the smallest integer such that s^ω is idempotent for all $s \in S$ and call ω the *exponent* of S. For any idempotent

$e \in S$, the set eSe forms a submonoid of S with identity e which we call the *local submonoid* of S associated with e.

Groups are a well-known special case of monoids. Recall that a monoid G is a *group* if every element $g \in G$ has an inverse g^{-1} such that $gg^{-1} = g^{-1}g = 1_G$. Every idempotent in a monoid M forms a trivial subgroup of M. Note also that by Lemma 2.6 an $\mathcal{H}$-class containing an idempotent is closed under multiplication and, more generally, one can show:

Lemma 2.7 *Let H be any $\mathcal{H}$-class of M, then H contains an idempotent if and only if H is a maximal subgroup of M.*

Consequently every $\mathcal{H}$-class contains at most one idempotent. Using Green's Lemma, one can further establish:

Lemma 2.8 *Any two maximal subgroups of a common $\mathcal{J}$-class are isomorphic.*

We say that S is a *union of groups* if each of its elements lies in a maximal subgroup of S. This is equivalent to the requirement that $s^{\omega+1} = s$ for each $s \in S$.

If every maximal subgroup of S is trivial, i.e. contains a single element, then S is said to be *aperiodic* or *group-free*. An important consequence of Lemma 2.7 is

Lemma 2.9 *S is aperiodic if and only if all its $\mathcal{H}$-classes contain a single element.*

An element a of S is said to be *regular* if there exists some $x \in S$ such that $axa = a$. A $\mathcal{J}$-class is said to be regular if all its elements are regular. As the next lemma shows, regularity is not a property of individual elements but rather of $\mathcal{J}$-classes.

Lemma 2.10 *The following are equivalent for a $\mathcal{J}$-class J of a finite semigroup:*

1. *J is regular;*

2. *J contains a regular element;*

3. *Every $\mathcal{R}$-class and every $\mathcal{L}$-class of J contains an idempotent;*

4. *J contains an idempotent.*

We say that a semigroup is *regular* if all its elements are regular.

A semigroup is said to be *completely simple* if it consists of a single $\mathcal{J}$-class and *0-simple* if it consists of two $\mathcal{J}$-classes one of which contains only 0. As we will see next, the structure of these semigroups is very well understood.

Note that by Lemma 2.6, a $\mathcal{J}$-class of S forms a completely simple subsemigroup of S if and only if it all its $\mathcal{H}$-classes are subgroups. We seek a refinement of Lemma 2.6 in order to understand the structure of multiplication within a regular $\mathcal{J}$-class. Let J be a regular $\mathcal{J}$-class of S and let $\cdot$ denote the multiplication in S. We denote by J^0 the 0-simple semigroup consisting of the elements of J and a 0 with the multiplication $\circ$ given by $s \circ t = s \cdot t$ if $s \cdot t$ lies in J and $s \circ t = 0$ otherwise.

Let G denote some finite group with multiplication $\circ$ and n, m be positive integers. A *Rees matrix* is an m by n matrix R with entries in $G \cup \{0\}$ and the corresponding *Rees semigroup* is the semigroup with elements in $([m] \times G \times [n]) \cup \{0\}$ and where the multiplication of non-zero elements is given by:

$$(i_1, g_1, j_1) \cdot (i_2, g_2, j_2) = (i_1,\ g_1 \circ R_{j_1,i_2} \circ g_2\ , j_2)$$

if R_{j_1,i_2} is in G and

$$(i_1, g_1, j_1) \cdot (i_2, g_2, j_2) = 0$$

if $R_{j_1,i_2} = 0$.

Theorem 2.11 *Every 0-simple semigroup is isomorphic to a Rees semigroup.*

In particular, for every regular $\mathcal{J}$-class J, we can construct a Rees-matrix representation of J^0. Say that J has m $\mathcal{R}$-classes and n $\mathcal{L}$-classes, with $\mathcal{H}_{ij}$ denoting the intersection of the i^{th} $\mathcal{R}$-class and j^{th} $\mathcal{L}$-class: because J is regular, we can assume that it has at least one idempotent per $\mathcal{R}$-class and per $\mathcal{L}$-class and in particular we assume that e_{11} is an idempotent in $\mathcal{H}_{11}$. Let G be the maximal group $\mathcal{H}_{11}$ and construct $R \in (G \cup \{0\})^{m \times n}$ as follows: first, entries $R_{i,1}$ and $R_{1,j}$ are assigned 1_G or 0 depending on whether $\mathcal{H}_{i1}$ or $\mathcal{H}_{1j}$ contains an idempotent or not. By Green's Lemma, there are elements l_j (resp. r_i) such that multiplication on the right (resp. left) by l_j maps the j^{th} $\mathcal{L}$-class (resp. i^{th} $\mathcal{R}$-class) to the 1st one and these can be chosen so that $r_i l_j = e_{11}$. The other entries $R_{i,j}$ are also 0 if $\mathcal{H}_{ij}$ does not contain an idempotent and is $r_i l_j$ otherwise. Simple calculations show that this Rees semigroup is isomorphic to J^0.

When a Rees matrix contains no 0 entries, we usually think of the corresponding semigroup as completely simple and every completely simple semigroup can be represented in this way [Gra68]. A 0-simple semigroup S whose Rees matrix contains only entries 0 and 1_G is said to be *flat*. By extension, a regular $\mathcal{J}$-class J is said to be *flat* if J^0 is flat and a semigroup is said to be *flat* if all its regular $\mathcal{J}$-classes are flat. An easy exercise shows that the 0-simple semigroup S is non-flat if and only if there exist idempotents $e, d, f \in S$ such that $e \,\mathcal{L}\, d \,\mathcal{R}\, f$ such that $defd \neq d$. In other words, a 0-simple S is flat if its idempotents generate an aperiodic subsemigroup. In fact, more generally, any semigroup S is flat if and only if its idempotents generate an aperiodic subsemigroup.

We further say that S is *orthodox* if its idempotents form a subsemigroup in S. Every orthodox semigroup is flat and the two notions coincide for simple and 0-simple semigroups.

2.1.4 Operations on Semigroups

We next describe a number of ways to construct semigroups from other semigroups. It is natural to ask of course how the computational power of the new semigroup compares with that of its building blocks and to try and relate algebraic operations on semigroups with combinatorial operations on languages or, taking the machine point of view on semigroups, with operations that somehow combine the computing power of two machines. For instance, running two automata in parallel on the same input can obviously be related to the direct product of two semigroups.

The *wreath product* of semigroups S and T, denoted $S \circ T$, is the set $S^{T^1} \times T$ with an operation defined as

$$(f_1, t_1) \cdot (f_2, t_2) = (f_1 \cdot f_2^{t_1}, t_1 t_2)$$

where $f_2^{t_1}(x) = f_2(xt_1)$, and $\cdot$ is the operation in S. There is a nice machine interpretation of the wreath product in terms of series connection of finite automata (see e.g. [Str94]).

Wreath products are central to a number of results about decompositions of certain semigroups. For instance, it can be shown that every semigroup divides a wreath product of groups and aperiodic semigroups [KR65] and that every solvable group divides a wreath product of Abelian groups.

For varieties $\mathbf{V}$, $\mathbf{W}$, we will denote by $\mathbf{V} * \mathbf{W}$ the variety of semigroups generated by the wreath products $S \circ T$ for $S \in \mathbf{V}$ and $T \in \mathbf{W}$. At this level, the wreath product is associative, that is we have $\mathbf{U} * (\mathbf{V} * \mathbf{W}) = (\mathbf{U} * \mathbf{V}) * \mathbf{W}$ for any varieties $\mathbf{U}, \mathbf{V}, \mathbf{W}$.

The *block product* of S and T, denoted $S \square T$ is a two-sided version of the wreath product. Its underlying set is $S^{T^1 \times T^1} \times T$ with the multiplication given by

$$(f_1, t_1) \cdot (f_2, t_2) = (g, t_1 t_2)$$

where $g : T \times T \to S$ is given by $g(x, y) = f_1(x, t_2y)f_2(xt_1, y)$. We will denote $\mathbf{V} \,\Box\, \mathbf{W}$ the variety of monoids generated by the block products $M \,\Box\, N$ for $M \in \mathbf{V}$ and $N \in \mathbf{W}$. In contrast with the wreath product, the block product is *not* associative, even at the variety level. Iterated block products also appear in many decomposition theorems although such statements must crucially take into account the bracketing of such products. For instance, a finite monoid is aperiodic if and only if it divides some

$$(M_1 \,\Box \ldots (M_{n-2} \,\Box\, (M_{n-1} \,\Box\, M_n)) \ldots)$$

where each M_i is idempotent and commutative [RT89], whereas a monoid belongs to the strictly smaller variety $\mathbf{DA}$ (see Subsection 2.1.6) if and only if it divides some

$$(\ldots((M_1 \,\Box\, M_2) \,\Box\, M_3) \ldots \Box\, M_n)$$

where each M_i is idempotent and commutative [ST02].

A *relational morphism* from a semigroup S to a semigroup T is a mapping $\pi : S \to 2^T$ such that $\pi(x)\pi(y) \subseteq \pi(xy)$ for any $x, y \in S$ and $\pi(x) \neq \emptyset$ for all $x \in S$. Furthermore, if S, T are monoids, we require that $1_T \in \pi(1_S)$.

The *Mal'cev product* $\mathbf{V} \,Ⓜ\, \mathbf{W}$ of the semigroup variety $\mathbf{V}$ and the monoid variety $\mathbf{W}$ is the class of monoids M such that there exists a relational morphism π from M onto a monoid N of $\mathbf{W}$ such that for all idempotents $e \in N$ we have $\pi^{-1}(e) = \{m \in M | e \in \pi(m)\}$ forms a semigroup belonging to $\mathbf{V}$.

2.1.5 Congruences and Finite Counting

Many combinatorial descriptions of language-varieties can be obtained through congruences that do some sort of "finite counting". We introduce here some useful notation and terminology. Let $t \geq 0$ and $p \geq 1$ be integers. We say that x and y are equal threshold t (and write $x = y\,(\text{thresh } t)$) if $x = y$ or x and y are both greater or equal to t. We further say that x, y are equal threshold t and modulo p (and write $x = y\,(\text{thresh } t, \mod p))$ if $x = y$ or $x, y \geq t$ and $x = y \pmod p$. For a word $u \in A^*$, we denote by $\alpha(u)$ the *alphabet*[2] of u, that is the set of letters of A that occur in u. For any $a \in A$ we further denote by $|u|_a$ the number of occurrences of a in u and by $\alpha_t(u)$ the vector of dimension $|A|$ which gives for every $a \in A$ the value of $|u|_a$ up to threshold t. Similarly, $\alpha_{t,p}(u)$ denotes the $|A|$-dimensional vector holding the values $|u|_a$ up to threshold t and modulo p.

A *subword* $u = a_1a_2\dots a_s$, with $a_i \in A$ of a word $x \in A^*$ is a factorization of x as

$$x = x_0a_1x_1\dots x_{s-1}a_sx_s$$

with $x_i \in A^*$. We denote by $\binom{x}{u}$ the number of occurrences of u as a subword of x, i.e. the number of possible different factorizations of x as above.

For any, $k \geq 0$ $t \geq 0$, $p \geq 1$, we can define an equivalence relation $\gamma_{k,t,p}$ on A^* as $x\,\gamma_{k,t,p}\,y$ if and only if x and y have the same number (threshold t and modulo p) of occurrences of each subword u of length at most k. In fact, the γ's are congruences of finite index. When the syntactic congruence of L is refined by $\gamma_{k,t,p}$, we say that membership in L depends on the number threshold t modulo p of subwords of length k. We will later on give algebraic descriptions of such languages and we note for now:

Lemma 2.12 *Let A be some finite alphabet. Any $x \in A^*$ is $\gamma_{k,1,p}$-equivalent to a word of length at most $p \cdot |A|^{kp}$.*

More generally, if $L_0, \dots, L_k$ are languages in A^* and $a_1, \dots, a_k$ are letters in A, we denote by $\binom{x}{(L_0a_1L_1\dots a_kL_k)}$ the number of factorizations of x as $x = w_0a_1w_1\dots a_kw_k$ with $w_i \in L_i$. When the a_i and L_i are such that for any x we have either $\binom{x}{(L_0a_1L_1\dots a_kL_k)} = 0$ or $\binom{x}{(L_0a_1L_1\dots a_kL_k)} = 1$ then we say that the concatenation $L_0a_1L_1\dots a_kL_k$ is *unambiguous*.

[2]In some of the litterature, this is alternatively called the *content* of u and is denoted $c(u)$.

For a language-variety $\mathcal{L}$, we denote by $UPol(\mathcal{L})$ the language variety generated by unambiguous concatenations $L_0a_1 \dots a_kL_k$ with $L_i \in \mathcal{L}$. We further let $M_pPol(\mathcal{L})$ denote the variety generated by the languages $\{x | \binom{x}{(L_0a_1L_1 \dots a_kL_k)} = j \pmod p\}$ for some $0 \leq j < p-1$ and $L_i \in \mathcal{L}$.

2.1.6 A Catalog of Varieties

The variety Theorem clearly establishes varieties as the central object of study in the algebraic theory of regular languages. There are a number of ways in which we might define varieties: through restrictions on automata or regular expressions, through congruences, through generators for the variety, through identities and so on. By the latter, we mean that varieties of semigroups can often be conveniently characterized as the class of semigroups whose elements satisfy a certain set of equalities[3], thus yielding an obvious algorithm to decide if S lies in $\mathbf{V}$ when this set is finite. Consider for instance the variety **Com** of commutative monoids: these are exactly the monoids satisfying the identity $xy = yx$. (It is also a simple exercise to show that the corresponding languages are exactly those for which membership depends on the number of occurrences of each letter threshold t and modulo p.) In fact, we will sometimes loosely use what are known as pseudo-identities although a formal treatment of them requires the presentation of a topological framework which we prefer to leave out (see [Pin97]).

We list here a number of varieties of semigroups and monoids which will be of importance in later chapters and for each of them give a number of alternate descriptions. We will be particularly interested in the combinatorial descriptions of the corresponding varieties of languages (when such descriptions are known). These varieties are listed for quick reference in the index of notation.

[3] In fact, *every* variety of finite semigroups can be characterized as the class of semigroups that ultimately satisfy a certain set of identities.

Varieties of Groups

It should first be noted that no "nice" combinatorial description of the languages whose syntactic monoids lie in the variety $\mathbf{G}$ of finite groups is known. This is related to the apparent impossibility of understanding the combinatorics of non-solvable groups. There are, however, good descriptions of languages recognized by a number of subvarieties of $\mathbf{G_{sol}}$ the variety of solvable groups [Thé79].

First, for a prime p let us denote by $\mathbf{G_p}$ the class of p-groups i.e. the groups of order p^α for some integer α. This variety is characterized by the identity $x^{p^\omega} = 1$. Moreover, it can be shown that L's syntactic monoid belongs to $\mathbf{G_p}$ if and only if there exists a k such that membership of x in L depends on the values $\binom{x}{u} \pmod p$ where $|u| \leq k$.

A group is said to be *nilpotent* if and only if it is the direct product $G_1 \times \ldots \times G_k$ where each G_i is a p_i-group for some prime p_i. Alternatively, if x, y are elements of a group G, we call $[x, y] = x^{-1}y^{-1}xy$ the *commutator* of x and y. For two subgroups H_1, H_2 of G we denote by $[H_1, H_2]$ the subgroup generated by commutators $[h_1, h_2]$ with $h \in H$ and $k \in K$. We can form the sequence $G = G_0 \supseteq G_1 \supseteq \ldots$ by setting $G_{i+1} = [G_i, G]$ and say that G is *nilpotent of class* k if G_k is the trivial group. It can be shown that this coincides with our previous definition of a nilpotent group. We also recursively define a *commutator of weight* t: any element of G is a commutator of weight 1 and $g \in G$ is a commutator of weight $t > 1$ if and only if there exist $u, v \in G$ commutators of weight t_1, t_2 respectively with $t_1 + t_2 = t$ and such that $g = [u, v]$. It is fairly simple to show that a group is nilpotent of class k if and only if the sole commutator of weight $k + 1$ in G is the identity element 1_G.

We denote by $\mathbf{G_{nil,k}}$ the variety of nilpotent groups of class k and by $\mathbf{G_{nil}}$ the variety of all nilpotent groups. In particular, all $\mathbf{G_p}$ are subvarieties of $\mathbf{G_{nil}}$ and $\mathbf{G_{nil,1}}$ coincides with the variety $\mathbf{Ab}$ of Abelian groups.

Theorem 2.13 ([Eil76, Thé83]) *A language L is recognized by a nilpotent* *group G of class k if and only if there is an integer $m \geq 2$ such that membership of x in L depends on the number modulo m of occurrences in x of each subword u of length at most k.*

In fact, one can choose m in this theorem to be the exponent of G. We will also need the following fact about a very special subclass of of solvable groups:

Lemma 2.14 *The group G lies in $\mathbf{G_p} * \mathbf{Ab}$ if and only if $[G,G]$ lies in $\mathbf{G_p}$.*

For any variety of groups $\mathbf{H}$, we will denote as $\overline{\mathbf{H}}$ the variety of monoids whose subgroups all lie in $\mathbf{H}$. We will refer to $\overline{\mathbf{G_{sol}}}$ as the variety of *solvable monoids.*

Aperiodic Varieties

Recall that M is aperiodic if no subset of it forms a non-trivial group. We denote by $\mathbf{A}$ the variety of aperiodic monoids. One can show that M lies in $\mathbf{A}$ if and only if $m^\omega = m^{\omega+1}$ for all $m \in M$.

A regular language L is said to be *star-free* if it can be described by an extended regular expression (i.e. a regular expression built using letters of the alphabet, $\emptyset$, concatenation, Kleene star and the Boolean operations union *and* complement) without using the Kleene star. For example, if $A = \{a, b\}$, the language $L = A^*ab^*$ is star-free because $L = \bar{\emptyset}a\overline{\bar{\emptyset}a\bar{\emptyset}}$. The following (much celebrated) theorem is due to M. P. Schützenberger:

Theorem 2.15 ([Sch65]) *A language L is star-free if and only if $M(L)$ is aperiodic.*

We have already mentioned that M is aperiodic if and only all its $\mathcal{H}$-classes are trivial. One can similarly consider the variety $\mathbf{J}$ of $\mathcal{J}$-trivial monoids which is known to be defined by the identities $(xy)^\omega = (yx)^\omega$ and $x^\omega = x^{\omega+1}$.

We say that a language L is *piecewise testable* if there exists $k \in \mathbb{N}$ such that membership of any word w in L depends on the set of subwords of length at most k occurring in w. The following is due to I. Simon [Sim75]:

Theorem 2.16 *The language L is piecewise testable if and only if $M(L)$ lies in* $\mathbf{J}$.

There are similar well-known descriptions of the variety of languages corresponding to the varieties $\mathbf{L}$ and $\mathbf{R}$ of respectively $\mathcal{L}$-trivial and $\mathcal{R}$-trivial monoids.

We denote by $\mathbf{A_1}$ the variety of *idempotent semigroups*, that is semigroups in which every element is idempotent. Such semigroups are often called *bands* and the lattice of subvarieties of $\mathbf{A_1}$ is completely understood (see e.g. [GP89]) although only some of the smallest of these varieties will be of importance in our work. Most importantly, we will consider the varieties:

- $\mathbf{J_1}$ of commutative bands or semilattices;
- $\mathbf{NB}$ of normal bands, that is bands satisfying $xyzx = xzyx$;
- $\mathbf{R_1}$ of $\mathcal{R}$-trivial bands, i.e. bands satisfying $xyx = xy$;
- $\mathbf{L_1}$ of $\mathcal{L}$-trivial bands, i.e. bands satisfying $xyx = yx$;
- $\mathbf{RB}$ of regular bands, i.e. bands satisfying $xyxzx = xyzx$. It can be shown that $\mathbf{RB}$ is the smallest variety containing both $\mathbf{R_1}$ and $\mathbf{L_1}$.

Note that a language L has its syntactic monoid in $\mathbf{J_1}$ if and only if membership of x in L depends on $\alpha(x)$. Correspondingly, $\mathbf{J_1}$ is generated, as a variety of monoids, by a single two-element monoid U_1 consisting of the idempotents $\{1, 0\}$ and multiplication defined in the obvious way. It can easily be shown that U_1 divides any monoid which is not a group.

Similarly, languages corresponding to the variety $\mathbf{A_{com}}$ of commutative aperiodic semigroups are the ones for which membership depends on occurrences of each letter threshold t for some t.

DS and its Subvarieties

For any variety of semigroups $\mathbf{V}$, we denote by $\mathbf{DV}$ the variety of semigroups whose regular $\mathcal{J}$-classes all lie in $\mathbf{V}$. All such varieties are contained in $\mathbf{DS}$, where $\mathbf{S}$ denotes the variety of all semigroups, and S lies in $\mathbf{DS}$ if and only if each of its regular $\mathcal{J}$-classes is closed under multiplication, which, from Lemma 2.6 amounts to requiring that each $\mathcal{H}$-class of a regular $\mathcal{J}$-class contain an idempotent. Thus, a semigroup in $\mathbf{DS}$ which is regular is in the variety $\mathbf{UG}$ of unions of groups. The following can be used to characterize $\mathbf{DS}$:

Lemma 2.17 *For any semigroup S, the following are equivalent:*

1. *S lies in $\mathbf{DS}$;*
2. *S satisfies the identity $((xy)^\omega(yx)^\omega(xy)^\omega)^\omega = (xy)^\omega$;*
3. *for any $x, y \in S$ where $x \leq_{\mathcal{J}} y$ and x is regular we have $x \mathrel{\mathcal{J}} xy \mathrel{\mathcal{J}} yx$;*
4. *for any $x, y \in S$ such that $xy \mathrel{\mathcal{R}} x$ we have in fact $R_x y \subseteq R_x$.*

This has a number of interesting consequences. For instance, if M is in $\mathbf{DS}$ and $u, v \in M^*$ are such that $\alpha(u) = \alpha(v)$, then u^ω and v^ω are $\mathcal{J}$-related. Moreover, in the special case where M is a union of groups (or furthermore is idempotent) then $u \mathrel{\mathcal{J}} v$ whenever $\alpha(u) = \alpha(v)$.

If M is outside $\mathbf{DS}$, then there exist two $\mathcal{J}$-related idempotents e, f such that $ef <_{\mathcal{J}} e$. One can use this to show that if M is not in $\mathbf{DS}$ then M is either divided by B_2 the syntactic monoid of $(ab)^*$ or divided by U the syntactic monoid of A^*bbA^*. These two monoids are aperiodic and both contain the six elements $\{1, a, b, ab, ba, 0\}$ although U has one more idempotent element than B_2. Their egg-box representations are given in Figure 2.1 with idempotents marked by *'s. It is also easy to show that B_2 divides $U \times U$.

Let $\mathbf{O}$ denote the variety of orthodox semigroups. The variety $\mathbf{DO}$ will play an important role in later chapters as we will exploit the combinatorial

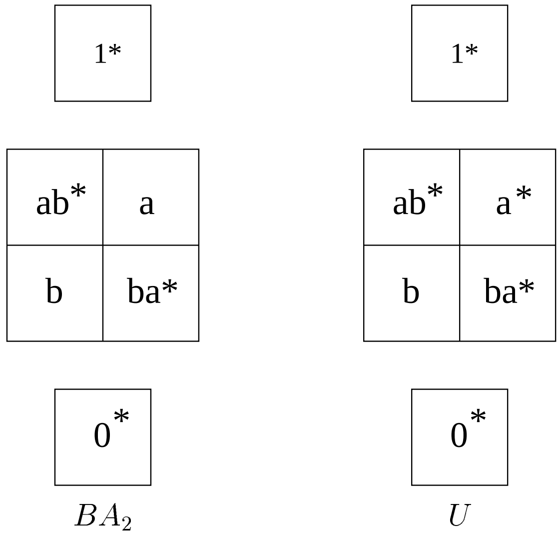


Figure 2.1: The egg-box representations of U and B_2.

description of the corresponding language-variety. By definition, M lies in **DO** if and only if the product of any two $\mathcal{J}$-related idempotents of M is also an idempotent in the same $\mathcal{J}$-class. This is equivalent to requiring that M be flat and lying in **DS**.

For a finite monoid M (and in particular any finite group G), we say that $x, y \in A^*$ are *M-equivalent* if for all morphisms $\psi : A^* \to M$ we have $\psi(x) = \psi(y)$. For example, ab and ba are M-equivalent for any commutative M.

Lemma 2.18 *Suppose $M \in$ **DO** and let $w_1, w_2 \in A^*$ be G-equivalent for any subgroup G of M. For any morphism $\phi : A^* \to M$ and any $x \in M$ such that $x\phi(w_1) \,\mathcal{H}\, x\phi(w_2) \,\mathcal{R}\, x$ we have in fact $x\phi(w_1) = x\phi(w_2)$.*

Proof. We first observe that for any idempotent $e \in M$ and any $u, v \in M$ lying $\mathcal{J}$-above e we have $(eu)^\omega e(ve)^\omega = (eu)^\omega(ve)^\omega$ since $e, (eu)^\omega$ and $(ve)^\omega$ are $\mathcal{J}$-related idempotents. Similarly, if f is another idempotent $\mathcal{J}$-related to e we have $euvf = eufevf$.

Let $\phi(w_1) = y_1$ and $\phi(w_2) = y_2$. Since $xy_1 \,\mathcal{H}\, xy_2 \,\mathcal{R}\, x$ there must exist $s, t \in M$ lying $\mathcal{J}$-below y_1 and y_2 with $xst = x = x(st)^\omega$ and $xy_its = xy_i =$

$xy_i(ts)^\omega$. Let T be the submonoid of elements of M lying $\mathcal{J}$-above $(st)^\omega$ and $\psi : T \to H_{(st)^\omega(ts)^\omega}$ given by $\psi(x) = (st)^\omega x(ts)^\omega$. Using our earlier observation we conclude that ψ is a morphism since

$$\psi(xy) = (st)^\omega xy(ts)^\omega = (st)^\omega x(ts)^\omega (st)^\omega y(ts)^\omega = \psi(x)\psi(y).$$

Since w_1 and w_2 are equivalent with respect to the group $H_{(st)^\omega(ts)^\omega}$, we must have $(st)^\omega y_1(ts)^\omega = \psi(y_1) = \psi(y_2) = (st)^\omega y_2(ts)^\omega$. Thus,

$$xy_1 = x(st)^\omega y_1(ts)^\omega = x(st)^\omega y_2(ts)^\omega = xy_2.$$

$\square$

Schützenberger [Sch76] proved the following characterization of languages whose syntactic monoids lie in **DO** and have subgroups in some **H**:

Theorem 2.19 *Let* $\mathbf{H}$ *be a variety of groups and* $\mathcal{L}$ *denote the language variety corresponding to* $\mathbf{J_1} \vee \mathbf{H}$*. Then the syntactic monoid of a language* L *lies in* $\mathbf{DO} \cap \overline{\mathbf{H}}$ *if and only if* L *is in* UPol($\mathcal{L}$).

As Lemma 2.22 will show, this also means $\mathbf{DO} \cap \overline{\mathbf{H}} = \mathbf{LI} \circledM (\mathbf{J_1} \vee \mathbf{H})$. Building on the work of D. Thérien and T. Wilke [TW98], we will now prove a slight refinement of Schützenberger's Theorem by characterizing these languages in terms of a convenient congruence. For $a \in \alpha(u)$, the *a-left* (resp. a-right) *decomposition* of u is the unique factorization $u = u_0 a u_1$ with $a \notin \alpha(u_0)$ (resp. $a \notin \alpha(u_1)$). For a finite group G, we define $\sim^G_{n,k}$ on A^* where $n = |A|$ by induction on $n + k$. First, we have $x \sim^G_{n,0} y$ if and only if x, y are G-equivalent. Next, we let $x \sim^G_{n,k} y$ when and only when:

1. $x \sim^G_{n,k-1} y$;
2. $\alpha(x) = \alpha(y)$;
3. For any $a \in \alpha(x) = \alpha(y)$, if $x = x_0 a x_1$ and $y = y_0 a y_1$ are the a-left decompositions of x and y then $x_0 \sim^G_{n-1,k} y_0$ and $x_1 \sim^G_{n,k-1} y_1$;

4. For any $a \in \alpha(x) = \alpha(y)$, if $x = x_0 a x_1$ and $y = y_0 a y_1$ are the a-right decompositions of x and y then $x_0 \sim^G_{n,k-1} y_0$ and $x_1 \sim^G_{n-1,k} y_1$.

This equivalence relation is well-defined since $|\alpha(x_0)| < |\alpha(x)|$ in (3) and $|\alpha(x_1)| < |\alpha(x)|$ in (4). It is easy to check that $\sim^G_{n,k}$ is in fact a congruence of finite index.

Theorem 2.20 *Let $M = A^*/\gamma$, with $|A| = n$. Then $M \in \mathbf{DO} \cap \bar{\mathbf{H}}$ if and only if $\sim^G_{n,k} \subseteq \gamma$ for some $k \in \mathbb{N}$ and $G \in \mathbf{H}$.*

Proof. For one direction, we need to show that $A^*/\sim^G_{n,k}$ is in $\mathbf{DO} \cap \overline{\mathbf{H}}$ for any integer k and any group $G \in \mathbf{H}$. We will appeal to the theorem of Schützenberger which we previously cited as Theorem 2.19: it is an easy exercise to verify that each $\sim^G_{n,k}$-class can be described by an unambiguous concatenation $K_0 a_1 \ldots a_s K_s$ where the K_i can be recognized by the direct product of an idempotent and commutative monoid (to verify whether $\alpha(x) = \alpha(y)$) and a group in $\mathbf{H}$ (to verify that x, y are G-equivalent). Schützenberger's Theorem thus insures that any disjoint union of these classes forms a language whose syntactic monoid lies in $\mathbf{DO} \cap \overline{\mathbf{H}}$.

For the second part of our proof, let us denote as $[u]$, for any $u \in A^*$, the γ equivalence class of u. We define the $\mathcal{R}$-decomposition (with respect to the congruence γ) of a string $u \in A^*$ as the unique factorization $u = u_0 a_1 u_1 \ldots a_t u_t$, with $a_i \in A$ and $u_i \in A^*$ such that:

1. $[u_0 a_1 \ldots a_s u_s] \not\mathcal{R} [u_0 a_1 \ldots a_s u_s a_{s+1}]$ for any $s < t$.

2. $[u_0 a_1 \ldots a_s] \mathcal{R} [u_0 a_1 \ldots a_s u_s]$ for any $s \leq t$.

3. $[u_0] \mathcal{H} 1_M$.

Because M lies in $\mathbf{DS}$ we know by Lemma 2.17 that $a_i \notin \alpha(u_{i-1})$. In particular, $u = u_0\, a_1\, u_1 a_2 \ldots a_t u_t$ is the a_1-left-decomposition of u. Symmetrically, we can define $\mathcal{L}$-decompositions of strings, which will relate to right-decompositions.

Let now k be the maximum of the number of $\mathcal{R}$-classes and the number of $\mathcal{L}$-classes of M. Let $B \subseteq A$ be a sub-alphabet with $|B| = m$ and suppose u, v are strings in B^*. We claim that if $G \in \mathbf{H}$ denotes the direct product of all maximal subgroups of M, then $u \sim^G_{m,mk} v$ implies $[u] = [v]$ and prove this by induction on m.

For $m = 0$, the claim is trivially true. For the inductive step, assume now $m \geq 1$ and suppose $u \sim^G_{m,mk} v$, with $\alpha(u) = \alpha(v) \neq \emptyset$. Let $u = u_0 a_1 u_1 \dots a_t u_t$ be the $\mathcal{R}$-decomposition of u with $t \leq k$. We write w_i for $u_i a_{i+1} \dots a_t u_t$ and, by our earlier remark, have $w_i = u_i a_{i+1} w_{i+1}$ is the a_{i+1}-left-decomposition of w_i, so there must be a decomposition $v = v_0 a_1 \dots a_t v_t$ such that $u_i \sim^G_{m-1,mk-i} v_i$ for $i < t$ (Note that this actually implies $u_i \sim^G_{m-1,(m-1)k} v_i$). By the induction hypothesis, we have $[u_i] = [v_i]$ for all $i < t$ so $[u] \,\mathcal{R}\, [u_0 a_1 \dots u_{t-1} a_t] = [v_0 a_1 \dots v_{t-1} a_t] \geq_{\mathcal{R}} [v]$. Symmetrically, we get $[v] \geq_{\mathcal{R}} [u]$ and thus $[u] \,\mathcal{R}\, [v]$. With the symmetric argument, we can also establish $[u] \,\mathcal{L}\, [v]$ so we have $[u] \,\mathcal{H}\, [v]$.

It remains to show that in fact $[u] = [v]$. Note that by definition of $\sim^G$, we have that u_t and v_t are G-equivalent. So by Lemma 2.18 we obtain,

$$[u] = [u_0 a_1 \dots u_{t-1} a_t][u_t] = [u_0 a_1 \dots u_{t-1} a_t][v_t] = [v_0 a_1 \dots v_{t-1} a_t][v_t] = [v]$$

$\square$

The variety $\mathbf{DA}$ of monoids whose regular $\mathcal{J}$-classes form aperiodic semigroups is contained in $\mathbf{DO}$ and is in fact equal to $\mathbf{DO} \cap \bar{\mathbf{I}}$ if $\mathbf{I}$ denotes the trivial variety and so $\mathbf{DA}$ is captured by the congruences $\sim^I_{n,k}$ for the trivial group I. Specifically, Theorem 2.19 yields

Corollary 2.21 *The syntactic monoid of a language $L \subseteq A^*$ lies in $\mathbf{DA}$ if and only if L is the disjoint union of unambiguous concatenations*

$$A_0^* a_1 A_1^* \dots a_k A_k^*$$

with $a_i \in A$ and $A_i \subseteq A$.

Monoids in **DA** and corresponding languages have many beautiful properties and interesting algebraic, logical and combinatorial characterizations which place them at the heart of many investigations in automata theory, complexity and logic (see [TT02b] for a survey). In particular, **DA** is characterized by the identity $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$. We note also that any aperiodic outside of **DA** is not in **DS** and therefore admits one of B_2 or U as a divisor.

Varieties Defined using Varieties

Given algebraic characterizations for varieties $\mathbf{V}$ and $\mathbf{W}$ and combinatorial descriptions for the corresponding languages, we can sometimes get good descriptions for varieties defined in terms of $\mathbf{V}$ and $\mathbf{W}$. The first example that comes to mind is of course their *join* $\mathbf{V} \vee \mathbf{W}$, that is the variety generated by elements of $\mathbf{V}$ and $\mathbf{W}$. It is easy to see that the languages corresponding to $\mathbf{V} \vee \mathbf{W}$ are Boolean combinations of languages recognized by monoids in $\mathbf{V}$ or $\mathbf{W}$ but more often than not, obtaining a convenient algebraic description for the join is very difficult. Similarly, a number of important varieties can be defined as $\mathbf{V} * \mathbf{W}$ or $\mathbf{V} \,\square\, \mathbf{W}$ and so on.

For any variety of monoids $\mathbf{V}$, we denote by $\mathbf{LV}$ the variety of semigroups S in which all local submonoids (i.e. submonoids of the form eSe) lie in $\mathbf{V}$. Relevant local varieties in this work include the local p-groups $\mathbf{LG_p}$ and the locally trivial semigroups $\mathbf{LI}$.

Theorem 2.22 ([PST88]) *Let* $\mathbf{V}$ *be a variety of finite monoids with an associated variety of languages* $\mathcal{L}$. *The variety of monoids associated with* $UPol(\mathcal{L})$ *is* $\mathbf{LI} \circledM \mathbf{V}$.

Theorem 2.23 ([Wei92]) *Let* p *be prime and* $\mathbf{V}$ *be a variety of finite monoids with an associated variety of languages* $\mathcal{L}$. *The variety of monoids associated with* $M_pPol(\mathcal{L})$ *is* $\mathbf{LG_p} \circledM \mathbf{V}$.

The application of this theorem to the case $\mathbf{V} = \mathbf{Com}$ will bear special importance in Chapter 4. We will also need the following characterization of the variety $\mathbf{LG_p} \circledM \mathbf{Com}$.

Theorem 2.24 *The monoid M lies in $\mathbf{LG_p} \circledM \mathbf{Com}$ if and only if every subgroup of M lies in $\mathbf{G_p} * \mathbf{Ab}$ and for all $\mathcal{J}$-related idempotents $e, f \in M$ holds $(efe)^{p^\omega} = e$.*

Proof. The simplest way to obtain this theorem is through topological results of [PW96]. Since we do not really need to introduce these sophisticated methods, we sketch here a proof from elementary principles.

Suppose first that there is a relational morphism $\pi : M \to T$ with T commutative and for any idempotent $d \in T$, $\pi^{-1}(d) \in \mathbf{LG_p}$. If e is an idempotent of M, then there is an idempotent d in $\pi(e)$ and $e\pi^{-1}(d)e$ is a p-group. It is easy to show that any idempotent f $\mathcal{J}$-related to e is also in $\pi^{-1}(d)$ so $(efe)^{p^\omega} = e$. Moreover, since T is commutative, for any x, y in the group $\mathcal{H}_e$, holds $d \in \pi(x^{-1})\pi(y^{-1})\pi(x)\pi(y)$. Thus $[x, y] \in \pi^{-1}(d)$ and so $[x, y]^{p^\omega} = e$ and therefore $\mathcal{H}_e$ is a group in $\mathbf{G_p} * \mathbf{Ab}$.

Conversely, let M be such that for all $\mathcal{J}$-related idempotents $e, f \in M$ holds $(efe)^{p^\omega} = e$. In particular, M lies in $\mathbf{DS}$ and so $x^\omega y^\omega$ is $\mathcal{J}$-related to $(xy)^\omega$. From each $\mathcal{J}$-class J_i of M, we pick a maximal subgroup G_i: the subgroup $[G_i, G_i]$ is normal in G_i and the group $K_i = G_i/[G_i, G_i]$ is Abelian. We also denote by ρ_i the canonical morphism from G_i into K_i and e_i the idempotent in G_i.

We define a monoid T in $\mathbf{J_1} \vee \mathbf{Ab}$ on the set $\bigcup K_i$. For $t_1 \in K_{i_1}$ and $t_2 \in K_{i_2}$ such that $(e_{i_1}e_{i_2})^\omega$ and e_k are $\mathcal{J}$-related idempotents in M, we choose m_1 and m_2 arbitrary pre-images in G_{i_1}, G_{i_2} in $\rho_{i_1}^{-1}(t_1)$ and $\rho_{i_2}^{-1}(t_2)$ respectively and define the multiplication of t_1 and t_2 in T as:

$$t_1 \cdot t_2 = \rho_k(e_k(m_1m_2)^{\omega+1}e_k).$$

Note that the particular choice of m_1, m_2 is unimportant. It is also easy to verify associativity.

We claim that there is a morphism ϕ from M to T such that the inverse image of any idempotent in T is a local p-group. We define ϕ by:

$$\phi(x) = \rho_i(exe)$$

where x^ω belongs to the $\mathcal{J}$-class J_i and e is the idempotent in group G_i. One must check that this is a well defined morphism. The crucial property we will use is that if e is the idempotent of G_i and f is another idempotent $\mathcal{J}$-related to e, then $\rho_i(efe) = e$. Indeed, since every group in M is in $\mathbf{G_p} * \mathbf{Ab}$ and $(efe)^{p^\omega} = e$ then efe lies in the subgroup $[G_i, G_i]$. Now we get:

$$\begin{aligned} \phi(x) \cdot \phi(y) &= \rho_x(e_x x e_x) \cdot \rho_y(e_y y e_y) \\ &= \rho_{(e_x y)}(e_{xy}(e_x x e_x)(e_y y e_y) e_{xy}) \\ &= \rho_{(e_{xy})}(e_{xy} x y e_{xy}) \\ &= \phi(xy) \end{aligned}$$

The same remark allows us to conclude that the inverse image of any idempotent of T is a local p-group. □

If an element x of S is regular, there exists by definition some a with $axa = a$ and $xax = x$ and we say that a is an *inverse* of x. In general, inverses in that sense may not be unique and from Lemma 2.6 one can easily show that every regular element of S has a unique inverse if and only if every regular $\mathcal{J}$-class of S has exactly one idempotent per $\mathcal{R}$ and $\mathcal{L}$-class. If each element of S has a unique inverse, we say that S is an *inverse semigroup* and denote by $\langle \mathbf{Inv} \rangle$ the variety generated by such semigroups. It is characterized by the identity $x^\omega y^\omega = y^\omega x^\omega$ and we have $\langle \mathbf{Inv} \rangle = \mathbf{J_1} * \mathbf{G} = \mathbf{J_1} Ⓜ \mathbf{G}$ (see [Pin95]).

A *Brandt semigroup* is a 0-simple inverse aperiodic semigroup. We denote by BS_k the (unique) Brandt semigroup such that the non-trivial $\mathcal{J}$-class has k^2 elements.

Lemma 2.25 *The monoid B_k can be generated by k elements $a_1, a_2, \ldots, a_k$ subject to the relations:*

1. $a_i a_j = 0$ *for any* $j \neq i+1$,

2. $a_i a_{i+1 \pmod k} \ldots a_{i+k \pmod k} = a_i$ *for all* i.

We will write B_k to denote the *Brandt monoid* BS_k^1. Note that we already have stressed the importance of B_2 when discussing subvarieties of $\mathbf{DS}$.

One last variety which we will consider is $\mathbf{DA} * \mathbf{G}$ which has recently emerged as one of particular importance in logical descriptions of regular languages. The following theorem is part of semigroup folklore and an explicit proof can be found for instance in [ST01].

Theorem 2.26 *The monoid M lies in $\mathbf{DA} * \mathbf{G}$ if and only if for any two $\mathcal{J}$-related idempotents e, f in M holds either $ef <_{\mathcal{J}} e$ or ef is idempotent.*

In particular, if M does not lie in $\mathbf{DA} * \mathbf{G}$, then it either admits U as a divisor or it is non-flat. In other words $\mathbf{DO} = (\mathbf{DA} * \mathbf{G}) \cap \mathbf{DS}$.

2.2 Computational Complexity

Computational complexity theory is concerned with classifying languages in terms of the resources needed to decide them in a certain model of computation. The classical and most natural measures are that of time and space required on a Turing machine (see [Sip97, Pap94, GJ79]) but the study of alternative measures and computation models have been a major part of the successful development of the theory.

2.2.1 Complexity Classes, Reductions and Completeness

Computational complexity theory has been hampered by the frustrating inability of the field to provide explicit lower bounds on resources needed for explicit functions. We trust that the reader is familiar with the P versus NP problem: despite years of intensive research, many leading theoreticians [Gas02b] feel that super-polynomial time lower bounds for a problem in NP are as far out of reach today as they were twenty-five years ago. Basic containments for the classes of languages L, P, PSPACE of languages recognizable by deterministic Turing machines in, respectively, logarithmic space, polynomial time, polynomial space and their non-deterministic counterparts NL, NP, NSPACE are easy to obtain:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NSPACE;$$

but although $L \neq PSPACE$ can be obtained through the space-hierarchy theorem, none of the other inclusions above is known to be strict.

Because getting explicit complexity lower bounds is so difficult, reductions have been a central tool of complexity theory since they allow us to at least *compare* the relative complexity of various problems and obtain strong indications that a given problem is hard. A *many-one reduction* from $L \subseteq A^*$ to $K \subseteq B^*$ is a function $f : A^* \to B^*$ such that for any $x \in A^*$, $x \in L$ if and only if $f(x) \in K$. If f is "easy enough" to compute (this might take on different meaning in different contexts), then A is "at least as hard" to compute as B.

For a complexity class $\mathcal{C}$, we say that a language K is $\mathcal{C}$-*complete* if K lies in $\mathcal{C}$ and for all $L \in \mathcal{C}$ there is an "easy enough" reduction from L to K. For instance, NL, P, NP and PSPACE are all known to have complete problems under many-one reductions computable in logarithmic space. Establishing that a problem K is, say, NP-complete under logspace reductions is significant because then K belongs to P if and only if all other problems in NP do. In Chapter 5 we will be concerned with a number of problems lying in NP and will write $L \leq_P K$ to denote the existence of a many-one reduction from L to K computable in polynomial time.

For $K \subseteq B^*$ we define a *Turing machine with a K oracle* to be an ordinary Turing machine with the ability to query a K-oracle, that is to decide in 1 time step whether some $w \in B^*$ belongs to K. We say that L is *polynomial-time Turing-reducible* to K, and denote this $L \leq_P^T K$ if L can be decided by a Turing machine with oracle K that runs in polynomial time. Note that a polynomial-time many-one reduction is a polynomial-time Turing reduction where only one query to the oracle is made. Similarly, we will say that L is *polynomial-time bounded-truth-table-reducible* to K, and denote this $L \leq_P^{btt} K$, if L can be decided by a Turing machine with oracle K that runs in polynomial time and makes only a constant number k of oracle queries. We will say that a language is NP-complete if it is NP-complete under polynomial time Turing-reductions.

Of course, for subclasses of L, completeness under, say, many-one logspace-computable reductions is meaningless and much weaker notions of reductions need thus be defined. A *projection* π of length s maps A^n to B^s in such a way that for each $j \in [s]$ there is a unique $i \in [n]$ such that the j^{th} bit of $\pi(x)$ depends only on the i^{th} bit of x.

2.2.2 Circuit Complexity

A *Boolean circuit* C with n (Boolean) inputs $X_1, \ldots, X_n$ is a directed acyclic graph with three types of nodes (or gates): $2n$ input nodes of in degree 0, a single output node of out-degree 0 and inner nodes with in- and out-degree at least 1. The input nodes are labeled with X_i or $\overline{X}_i$ while the inner nodes and output node are labeled with a *symmetric* Boolean function chosen from some predetermined base (unless otherwise specified, this base is $\{\textsc{And}, \textsc{Or}\}$).

Such a circuit naturally computes a function from $\{0,1\}^n \to \{0,1\}$ as follows: given an input $x = b_1 \ldots b_n$, nodes in C are recursively assigned a Boolean value. First, the input gates $X_i, \overline{X}_i$ get value b_i and $1-b_i$ respectively. Next, if the gates $g_1, \ldots, g_t$ have been assigned values $v_1, \ldots, v_t$ and are the inputs to gate g (i.e. are the set of nodes with arcs to g) then g is assigned the value $f(v_1, v_2, \ldots, v_t)$ where f is the label assigned to g. Because, every inner node has in-degree and out-degree at least 1, every gate in the circuit is assigned some value during this process. The output node is always assigned a value last and this value $C(x)$ is the *output of the circuit.*

The language $L(C)$ accepted by C is the set $\{x|C(x) = 1\}$. The *size* of the circuit is the number of gates in C and its *depth* is the length of the longest path from an input node to the output node. Of course, we can easily refine this definition to allow circuits to process non-Boolean inputs or to allow non-symmetric functions computed at each gate.

A circuit can only process inputs of some fixed length although in general we are interested in using circuits as machines to recognize subsets of $\{0, 1\}^*$. We say that the language $L \subseteq \{0, 1\}^*$ is recognized by the circuit family $\mathcal{C} = (C_0, C_1, \ldots)$ if the n^{th} circuit C_n processes inputs of length n and accepts $L \cap \{0, 1\}^n$. We can then define the size and depth of $\mathcal{C}$ as functions from $\mathbb{N}$ to $\mathbb{N}$ in the obvious way.

Such models of computation, where different lengths of input are processed by different machines are called *non-uniform* models. Their power exceeds that of Turing machines since they can, for example, recognize arbitrary languages over a 1-letter alphabet.

The following symmetric Boolean functions are traditionally used as parts of bases in Boolean circuits: $\textsc{Mod}_m$ is the function which returns 1 if the sum of its input bits is divisible by m; $\textsc{Threshold}_t$ returns 1 if at least t of its input bits are 1; $\textsc{Majority}$ returns 1 if its input contains more 1's than 0's (i.e. $\textsc{Majority}$ is $\textsc{Threshold}_{n/2}$). We now define the following circuit complexity classes:

- AC^0 is the class of languages which can be recognized by a family of $\{\textsc{And},\textsc{Or}\}$-circuits of unbounded fan-in, polynomial size and constant depth;

- $\mathrm{CC}^0[m]$ is the class of languages which can be recognized by a family of {Mod$_m$}-circuits of unbounded fan-in, polynomial size and constant depth;

- CC^0 is the union over all m of the $\mathrm{CC}^0[m]$ classes;
- $\mathrm{ACC}^0[m]$ is the class of languages which can be recognized by a family of {And,Or,Mod$_m$}-circuits of unbounded fan-in, polynomial size and constant depth;
- ACC^0 is the union over all m of the $\mathrm{ACC}^0[m]$ classes;
- TC^0 is the class of languages which can be recognized by a family of {Majority}-circuits of unbounded fan-in, polynomial size and constant depth;
- NC^1 is the class of languages which can be recognized by a family of {And,Or}-circuits with bounded fan-in, polynomial size and $O(\log n)$ depth;
- NC is the class of languages which can be recognized by a family of {And,Or}-circuits with bounded fan-in, polynomial size and $O(\log^k n)$ depth for some k.

Since all these classes are defined using a non-uniform model of computation, there is no way to relate them to the usual Time/Space classes. However, uniform versions of these classes can be devised by requiring that there exist a Turing machine which, given n can produce a description of the circuit C_n within strict resource bounds. Conversely, Turing machine models can be made non-uniform by introducing so-called advice tapes. We will for the most part completely disregard uniformity issues in this work but we note that the inclusions

$$AC^0, CC^0 \subseteq ACC^0 \subseteq TC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq NC \subseteq P \subseteq NP \subseteq PSPACE$$

hold in suitably defined uniform and suitably defined non-uniform variants of these classes.

Boolean circuits, and particularly shallow Boolean circuits have served as an interesting tool in the study of parallel computation. Just as P is usually viewed as capturing the notion of sequential tractability, uniform NC is usually thought of as the class of problems with efficient *parallel* algorithms. The book of H. Vollmer [Vol99] presents a nice overview of circuit complexity and its importance in theoretical computer science today.

Subclasses of NC^1 have been the subject of intense research since the 1980's (see survey [All97]). The hope once was that techniques developed to separate various subclasses of NC^1 would be eventually built upon and refined in order to separate more powerful classes. Unfortunately, the important work on so-called "natural proofs", introduced in the seminal paper of A. Razborov and S. Rudich [RR97], indicates that a separation of even TC^0 from NP will require radically different methods than current combinatorial lower bound methods for circuits.

Still, circuit complexity has delivered very interesting results. A series of papers (starting from [Ajt83, FSS84]) established that PARITY (i.e. MOD_2) does not lie in AC^0, thus separating AC^0 from ACC^0. Subsequent work culminated in exponential size lower bounds for depth k AC^0 circuits computing an explicit function computable by depth $(k+1)$ linear-size AC^0 circuits [Hås87]. Very different techniques further showed that in fact MOD_m does not lie in $\mathrm{CC}^0[p]$ for p prime, unless m is a power of p [Raz87, Smo86]. A lot of other results have shown lower bounds for restricted classes of $\mathrm{CC}^0[m]$, ACC^0 and TC^0 circuits but despite impressive work in this field, we know of no super-linear size lower bound for depth-3 $\mathrm{CC}^0[6]$-circuits computing an NP-complete problem.

Many surprising circuit-complexity upper bounds have also been established. Let us mention for instance that $\mathrm{THRESHOLD}_{\log^c n}$ [FKPS85, HWWY94] and addition of $\log n$ n-bit numbers (see [Str94]) can be done in AC^0 while division and multiplication of n n-bit numbers can be computed in uniform TC^0 [HAB02].

2.2.3 Branching Programs

A *branching program* on n Boolean variables is a directed acyclic graph with a distinguished source node of in-degree 0 and two sink nodes s_0 and s_1 of out-degree 0. The source nodes and the inner nodes are labeled with a variable x_i and have two outgoing edges labeled 0 and 1. A branching program represents in a natural way a Boolean function $f : \{0,1\}^n \to \{0,1\}$: a given $x \in \{0,1\}^n$, defines a unique path from the source to one of the sinks by following at any node labeled x_i the edge labeled with the value of the i^{th} bit of x. Naturally, $f(x)$ is the label of the sink reached in this way.

We will view branching programs (or BP's for short) as a natural non-uniform computation model somewhat akin to Boolean circuits. They are also very useful when seen as a data structure for Boolean functions (in that case they are alternatively referred to as binary decision diagrams). Natural measures for the complexity of a branching program include size (number of nodes) and depth (length of longest source to sink path).

Branching programs have received a lot of attention both from theoreticians and from researchers in more application oriented fields such as verification and model checking. The book of I. Wegener is an excellent introduction to both theory and applications of BP's [Weg00].

A restricted class of BP's plays an important role in the motivation for our work. A *bounded-width branching program* (or BWBP) of width k is a special case of BP in which each inner node belongs to some *level* and edges go only from level i to level $i+1$. Furthermore, all nodes of a given level query the same input bit. This model was introduced by [BDFP86], partly as means of identifying interesting subclasses of NC^1. Indeed, it is easy to show that every Boolean function that can be represented by a family of BWBP's of polynomial depth can be computed by a family of NC^1 circuits. It was conjectured at the time that the MAJORITY function could not be computed by BWBP's in sub-exponential length. This was disproved by the remarkable result of D. Barrington which will be discussed in our next chapter.

Chapter 3

Programs over Monoids

The class of languages which can be recognized via morphism by a finite monoid is quite limited. Over the past twenty years, different formalisms have been introduced to generalize the notion of recognition of a language by a monoid. The focus of this chapter is a model of computation introduced by D. Barrington and D. Thérien known as programs over monoids.

3.1 From Homomorphisms to Programs

3.1.1 The Program Model

An n-input *program* ϕ over a monoid[1] M (or M-program) of length t is a sequence of *instructions*

$$\phi = (i_1, f_1)(i_2, f_2) \dots (i_t, f_t)$$

where the i_j's are indices in $[n]$ and the f_j's are functions from the input alphabet A into M. We will sometimes refer to the f_j's as *query functions*. As the terminology suggest, such programs process only inputs of length n and on input $x = x_1 x_2 \dots x_n$, the output $\phi(x)$ of ϕ on x is

$$\phi(x) = f_1(x_{i_1}) f_2(x_{i_2}) \dots f_t(x_{i_t}) = m.$$

[1]More generally, one can consider programs over semigroups although most of the literature focuses on programs over monoids. This reflects the fact that the existence of an identity element is helpful when designing such programs.

Note that the right-hand side can be seen alternatively as t elements of M or as m, their product in M. We say that ϕ *accepts* x if $\phi(x)$ belongs to some accepting subset $F \subseteq M$ and say that $L \subseteq A^n$ can be recognized by ϕ if there exists some F such that $L = \{x : \phi(x) \in F\}$.

In general, we say that $L \subseteq A^*$ can be recognized by an M-program of length $s(n)$ if there is a sequence $(\phi_0, \phi_1, \ldots)$ where ϕ_n is an n-input program of length bounded by $s(n)$ that can recognize $L \cap A^n$. In effect, such a sequence can be seen as a non-uniform projection of length $s(n)$ from L to the word problem for M but we will use in this context the program terminology. Note that a morphism is a special case of a program-family: each ϕ_n has length n, the i^{th} bit queried is always the i^{th} input letter and the query function is always the same.

The program model is non-uniform and has super-Turing computational power although, just as Boolean circuit models, standard uniform versions of it can be defined.

The motivation for introducing this model of computation originally lies in the study of bounded-width branching programs. D. Barrington observed that the edges connecting two levels of a BWBP can be seen as two transformations on k points $f_0, f_1 : [k] \to [k]$ where f_0 and f_1 correspond to the edges with labels 0 and 1 respectively. Thus, a BWBP can be seen as a program over the finite monoid generated by all functions occurring in the BP. This algebraic point of view led to a surprising theorem.

Theorem 3.1 ([Bar89]) *The language $L \subseteq A^*$ is recognized by a program of polynomial length over some finite monoid M if and only if L lies in non-uniform NC^1.*

In fact, we can replace the finite monoid M by any non-solvable group and the theorem relies on a property of finite simple non-Abelian groups which had already been uncovered twenty years earlier by W. Maurer and J. Rhodes

[MR65] although its computational complexity implications were first noticed by Barrington. In particular, because MAJORITY lies in NC^1 and because S_5, the group of permutations on 5 points is non-solvable, MAJORITY was shown to be computable by a family of width-5 branching programs of polynomial depth.

There exists a deep connection between shallow Boolean circuits and programs over monoids which was gradually uncovered in the 80's using tools which had been developed in the context of algebraic automata theory. Barrington and D. Thérien showed that many natural subclasses of NC^1 also had nice algebraic characterizations [BT88], the most important of which are:

Theorem 3.2 *Let L be a language in A^*:*

- *L lies in non-uniform AC^0 if and only if it can be recognized by a program of polynomial length over some finite aperiodic monoid M;*
- *L lies in non-uniform CC^0 if and only if it can be recognized by a program of polynomial length over some finite solvable group G;*
- *L lies in non-uniform ACC^0 if and only if it can be recognized by a program of polynomial length over some finite solvable monoid M.*
- *L lies in non-uniform NC^1 if and only if it can be recognized by a program of polynomial length over some finite non-solvable group G.*

Moreover, uniform versions of these theorems can easily be obtained. Remarkably, PARITY, the word problem of the group C_2, the smallest non-aperiodic monoid, was historically the first language shown not to belong to AC^0 [FSS84]. In retrospect, Theorem 3.2 shows how natural a target PARITY truly was.

Programs over monoids therefore offer an algebraic approach to the study of shallow Boolean circuits and BWBP's. This has many advantages: foremost powerful results and insights from algebraic automata theory can be ported to circuit complexity theory. Secondly, very fine natural parametrizations of NC^1

can be obtained [MPT91], some of which do not have a natural description in terms of circuits. The same phenomenon also helps us choose reasonable and natural targets for progress towards the resolution of questions such as CC^0 vs. NC^1: indeed, this question boils down to bounding the power of polynomial length programs over a solvable group. Once the question is posed in these terms, a natural research program emerges: try to bound the computational power of polynomial length programs over what algebraic automata theory suggests are more and more intricate classes of groups.

This was pursued early on in [BST90] where it was shown that AND cannot be computed by a program over a nilpotent group and cannot be computed in sub-exponential length by a program over any group in the variety $\mathbf{G_p} * \mathbf{Ab}$. It can also be shown that computing MOD_m requires programs of exponential length over groups in this variety unless m divides the order of the group [BS94, BS99]. These results can be translated into size lower bounds for circuits consisting of MOD_q gates at the input level followed by a number of MOD_p gates for a prime p (see also [ST, Cau96]). The same techniques led to a proof that $CC^0[q]$ circuits cannot compute AND in sub-linear size [Thé94] and it is fair to say that the state of the art lower bound technology for modular circuits was developed using the algebraic approach.

The power of programs over aperiodics has also been explored: one can give an algebraic characterization of AC^0 restricted to circuits of depth k using the "dot-depth" parametrization of aperiodics (see [BT88]) and precisely characterize the power of polynomial-length programs over semigroups of dot-depth 1 [MPT00] and monoids in **DA** [GT03]. It has also been shown that some aperiodics are too weak to compute the MOD_m function with programs, regardless of length [Thé89].

The results mentioned thus far bound the power of polynomial length programs over fixed varieties of "weak" monoids. Other results bound the expressiveness of *short* programs over "powerful" varieties of monoids: it can be shown that families of programs of length $o(n \log \log n)$ have very limited capabilities (see comments in Section 3.4) and in particular that they cannot compute MAJORITY regardless of what the underlying monoid is [BS95]. A completely different approach has shown that for any group G, the output of any "rich enough" G-program which queries each input bit only once is very close to the uniform distribution on G [GRT00].

3.1.2 Summary of Results

In this chapter we present some new results about the limited computational power of programs over certain varieties of monoids as well as introduce further motivation for the next chapters. We first review in the next section basic properties of programs and establish some of the tools that we will use in the later sections. Section 3.3 constitutes the core of this chapter: we try to answer two fundamental questions about the program over monoid model. On one hand we seek to characterize the monoids which are powerful enough to recognize arbitrary languages when no length restriction is imposed on the programs. The existence of such monoids, which we will call *universal* can easily be proved and corresponds to the well-known fact that even exponential-size depth-2 AC^0 circuits have universal computing power. On the other hand some monoids are so weak that any computation they can do can be realized by a program of polynomial length. Surprisingly, this polynomial length property, which we will define more formally, appears to be dual to universality. We conjecture that every finite monoid M is either universal or able to perform all of its computations in polynomial length, depending on whether M belongs to the variety $\mathbf{DA} * \mathbf{G} \cap \overline{\mathbf{G_{nil}}}$. Our main result supporting the conjecture is that this dichotomy does hold if M belongs to the variety $\mathbf{DS}$. We also prove that M is universal if it does not belong to $\mathbf{DA} * \mathbf{G}$ but that it has the polynomial length property if it is the wreath product of an aperiodic monoid in **DA** and a p-group G. For the most part, these results have been published as [TT02a].

P. McKenzie, P. Péladeau and D. Thérien [MPT91] showed that for any variety $\mathbf{V}$ of monoids, the class $\mathcal{P}(\mathbf{V})$ of languages which can be recognized by polynomial length programs over a monoid in $\mathbf{V}$ is essentially characterized by the regular languages in $\mathcal{P}(\mathbf{V})$. For every $\mathbf{V}$ there exists a maximal variety of monoids $\mathbf{W}$ such that $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ and when $\mathbf{V} = \mathbf{W}$, we say that $\mathbf{V}$ is a *program-variety*. In the terminology of H. Straubing [Str00], $\mathbf{V}$ is a program-variety if the only monoids whose multiplication can be "simulated" by a polynomial length program over a monoid in $\mathbf{V}$ are the monoids of $\mathbf{V}$ itself. In Section 3.4, we argue that identifying such varieties is the very goal of an algebraic approach to NC^1. For certain program-varieties $\mathbf{V}$, the computational limits of $\mathbf{V}$-programs are even more dramatic because *any* language L that contains a so-called neutral letter and can be recognized by a program over some $M \in \mathbf{V}$ can be recognized by a morphism over some $N \in \mathbf{V}$. We prove that **Com** and **J** have this property and discuss the implications.

3.2 Basic Properties of Programs

Lemma 3.3 *Let G be an arbitrary group and let G_c be the subgroup generated by all commutators of weight c. Then any function $f : A^c \to G_c$ can be realized by a G-program ϕ_f of length $(d_G)^c$, where d_G depends on G.*

Proof. We use induction on c: if $c = 1$, the program is simply $\phi_f = (1, f)$. For $c > 1$, let g be a commutator of weight c_1 and h be a commutator of weight c_2, where $c_1 + c_2 = c$. For any fixed x in A^{c_1}, there exists, from the induction hypothesis, a program $\phi_{x,g}$ that outputs g on input x and 1_G on any input different from x. Similarly, for any fixed y in A^{c_2} there is a program $\phi_{y,h}$ that outputs h on input y and 1_G on any other input. By the induction hypothesis, such programs also exist for g^{-1} and h^{-1} since these are also in G_{c_1} and G_{c_2}

respectively. Let now z be in A^c, with $z = xy$, where x is in A^{c_1} and y is in A^{c_2}: we construct $\phi_{z,[g,h]} = \phi_{x,g^{-1}}\phi_{y,h^{-1}}\phi_{x,g}\phi_{y,h}$, where it is understood that the first and third segments query the prefix of length c_1 of the input while the second and fourth segments query the suffix of length c_2. This program is easily seen to have the property that it yields $[g, h]$ on input z and the identity element on any other input. We finally get the desired program ϕ_f as the concatenation of the various $\phi_{z,f(z)}$, for all z in A^c. Note that the program has length exponential in c. □

> *Remark 3.3.* In particular, if a group G is not nilpotent of class c then G_{c+1} contains some $g \neq 1_G$ and there exists a G-program ϕ of length exponential in c such that $\phi(x) = 1$ if and only if all bits of $x \in \{0,1\}^{c+1}$ are on and $\phi(x) = g$ otherwise.

It has proved fruitful, in the study of complexity classes lying within NC^1 to represent subsets of $\{0,1\}^n$ by polynomials over a finite ring. This was the starting point of R. Smolensky and A. Razborov's algebraic approach to proving that PARITY and MAJORITY do not lie in AC^0 [Smo86, Raz87] and other subsequent similar work (see the survey [Bei93]). We will represent functions from $A^n \to \mathbb{Z}_p$ as polynomials over the finite ring $\mathbb{Z}_p$ in the $k \cdot n$ boolean variables $x_1^{a_1}, x_1^{a_2}, \dots, x_1^{a_s}, \dots, x_n^{a_1}, \dots, x_n^{a_s}$ where $A = \{a_1, \dots, a_s\}$. The intended meaning of these variables, of course, is that $x_i^{a_j}$ is equal to 1 if the i^{th} letter of the input $x \in A^n$ is a_j and is 0 otherwise. For this reason we will in fact be working over the semi-ring $\mathbb{Z}_p[x_1^{a_1}, \dots, x_n^{a_s}]$ modulo the identities $(x_i^{a_j})^2 = x_i^{a_j}$ for all i, j and $x_i^{a_j} \cdot x_i^{a_l} = 0$ for all i and all $j \neq l$.

Such polynomials naturally represent a function from $A^n \to \mathbb{Z}_p$ and, conversely, any function $f : A^n \to \mathbb{Z}_p$ can be represented as a polynomial of this form because the polynomial $x_1^{c_1} x_2^{c_2} \dots x_n^{c_n}$ is equal to 1 if the input x is $c_1c_2 \dots c_n$ and is 0 otherwise. We say that the language $L \subseteq A^n$ is *recognized* by the polynomial r if for all $x \in A^n$, $r(x) = \chi_L(x)$. Typical measures of "complexity" for these polynomials include degree and size (number of terms).

We will find it is useful to take this point of view when bounding the computational power of certain monoids. The next lemma, for instance, shows that programs over p-groups can be represented by particularly "simple" polynomials over $\mathbb{Z}_p$.

Lemma 3.4 *Let G be a p-group. For every n-input G-program ϕ over the alphabet A and any accepting subset $F \subseteq G$, the language $L(\phi)$ is recognized over $\mathbb{Z}_p$ by a polynomial of degree at most d_G.*

Proof. We first note that this Lemma is proved in [PT88] in the special case $A = \{0, 1\}$. We will use the characterization of languages recognized by p-groups presented earlier in Chapter 2. Recall that the word problem for a p-group boils down to counting modulo p the number of occurrences of all subwords of length at most k_G.

Let ϕ be an n-input G-program over the alphabet $A = \{a_1, \ldots, a_s\}$. Note that if the polynomials r_1, r_2 recognize $L_1, L_2 \in A^n$ respectively, then $(1 - r_1)$ recognizes $A^n - L_1$ and $r_1 r_2$ recognizes $L_1 \cap L_2$. Thus, in light of the previous remarks, it is sufficient to show that for all $u \in G^k$ and all $0 < i < p - 1$, the set $\{x \in A^n : \binom{\phi(x)}{u} = i \pmod p\}$ is recognized over $\mathbb{Z}_p$ by a polynomial of bounded degree. To see this, note that any occurrence of u as a subword of $\phi(x)$, is the result of k instructions giving a specific output. In other words, there are input variables $x_{i_1}, \ldots, x_{i_k}$ and alphabet letters $b_1, \ldots, b_k$ such that u occurs in $\phi(x)$ precisely because $x_{i_j} = b_j$ in x. These x_{i_j} have the correct value if and only if the monomial $x_{i_1}^{b_1} \ldots x_{i_k}^{b_k}$ evaluates to 1 so we can count the number of occurrences of u modulo p using a polynomial of degree bounded by k. $\square$

We have already mentioned that when reasoning about programs it is often useful to think of the output of the program as the word in M^* corresponding to the concatenation of the elements output by each instruction. Similarly, notice that an n-input program is a finite sequence of instructions, i.e. a finite word over the alphabet Σ of size $n \times A^M$ consist of pairs (i, f) with $i \in [n]$ and $f : A \to M$. This point of view allows us to reason, for example, about the similarity in behavior of two programs which are equivalent with respect to certain finite index congruences over Σ^*.

Consider for instance two n-input programs ϕ_1, ϕ_2 over an idempotent commutative monoid M. It is easy to see that if over this large alphabet we have $\alpha(\phi_1) = \alpha(\phi_2)$ then for any $x \in A^n$ we will have $\alpha(\phi_1(x)) = \alpha(\phi_2(x))$ and thus $\phi_1(x) = \phi_2(x)$.

If N is a submonoid of M, then every N-program is also an M-program. If $N = \theta(M)$ for some surjective morphism θ, then this morphism can be used to transform an M-program ϕ to an N-program $\theta(\phi)$ in the obvious way: every instruction of ϕ, say (i, f) becomes $(i, \phi(f))$. Then, obviously, $\theta(\phi)(x) = \theta(\phi(x))$ for any input x. The opposite process is in fact more interesting: if ψ is an N-program, then there is an M-program ϕ such that $\theta(\phi) = \psi$ and so any language recognized by an N-program with accepting subset $F \subseteq N$ can be recognized by an M-program using accepting subset $\theta^{-1}(F)$.

Note also that when $\phi = (i_1, f_1) \dots (i_s, f_s)$ is a program over a group G, we will write ϕ^{-1} to denote the program $(i_s, f_s^{-1}) \dots (i_1, f_1^{-1})$. Of course for any input x we get $(\phi(x))^{-1} = \phi^{-1}(x)$.

It is sometimes convenient to consider so called *k-programs* over M in which instructions are allowed to query k-tuples of input positions instead of single positions. The computing power of polynomial-length k-programs does not exceed NC^1 although for a specific monoid polynomial length $(k+1)$-programs might be strictly more powerful than polynomial length k-programs. A simple trick shows that every polynomial length k-program over M can be rewritten as a t-program in which the t-tuples are only queried once and in some fixed order

[Str00]. This formalism is helpful when relating programs to logic [Str94, Str01].

3.3 Universality *vs.* Polynomial Length Property

In Theorems 3.1 and 3.2 the polynomial restriction on the length of the programs is crucial. As the next example illustrates however some monoids cannot take advantage of a relaxation of the length requirement.

Example 3.4. Let ϕ be an n-input program of length s over a commutative M. Any two instructions in ϕ can be commuted at will without affecting the output of the program since the underlying monoid is commutative. Moreover, two adjacent instructions that query the same input letter can be coalesced into a single instruction outputting the product in M of the outputs of the two original instructions. Therefore there is an n-input program ϕ' of the form

$$\phi' = (1, f_1)(2, f_2) \dots (n, f_n)$$

such that $\phi(x) = \phi'(x)$ for all $x \in A^n$.

When analyzing the computational power of programs over commutative monoids, the restriction to polynomial length is thus completely irrelevant because any such program can be assumed of length n.

We say that a monoid M has the *polynomial length property* (often abbreviated PLP) if there exists a polynomial $p(n)$ such that for each n and for every n-input M-program ϕ with target set $F_\phi \subseteq M$, there exists an equivalent n-input M-program ψ, with possibly a different target set $F_\psi \subseteq M$, of length $p(n)$. By *equivalent* we mean that for any input x, we have $\phi(x)$ belonging to F_ϕ if and on $\psi(x)$ belongs to F_ψ.

It is not clear whether the PLP is preserved by taking submonoids or morphic images and by taking direct products. Let M have the PLP and let N be a submonoid of M: an N-program is also an M-program and this M-program can be reduced to an equivalent one of polynomial length, but this new M-program may involve in its instructions elements which are outside of N, and hence may not be an N-program. Let next $N = \theta(M)$: for any N-program ϕ, we have seen that we can construct an M-program ρ such that $\theta(\rho) = \phi$, and ρ can be reduced to an equivalent program of polynomial length, say ψ. We are unfortunately not guaranteed that the accepting subset of ψ is the pre-image of some subset of N, i.e. it may be that $\theta(n_1) = \theta(n_2)$ where n_1 is accepting and n_2 is rejecting: hence, there is no clear way of transforming ψ into an N-program. A similar problem occurs when we look at programs ϕ over $M \times N$ where M and N have the PLP unless the accepting subset is the direct product of a subset of M and a subset of N. This is probably not an easy problem to get around as the following example illustrates.

Example 3.4. Since the AND function lies in NC^1, there is a polynomial length program to compute it over the (non-solvable) group $S_3 \times A_5$. However, any program computing AND over the subgroup S_3 is known to require exponential length [BST90]. This does not ruin the possibility that the polynomial length property is preserved under division, as PLP provably does not hold in $S_3 \times A_5$, but the example shows that an argument to prove the closure property will crucially depend on PLP holding for the larger monoid.

This inconvenience, however, motivates the following definition: we say that an M-program ψ is a *contraction* of an M-program ϕ if

1. $\psi(x) = \phi(x)$ for any $x \in A^n$;
2. Every instruction in $\psi(x)$ is an instruction ϕ.

In other words, ψ can be obtained from ϕ by permuting, deleting or duplicating instructions of ϕ and the two programs always have the same output. We further say that M has the *polynomial length contraction property* (abbreviated PLCP) if there exists a polynomial $p(n)$ such that for each n and for every n-input M-program ϕ, there exists a contraction ψ of ϕ whose length is bounded by $p(n)$. For example, the arguments given in Example 3.3 show that commutative monoids have the PLCP. Of course, any M having the PLCP also has the PLP but in addition:

Lemma 3.5 *If M, N are monoids with $N \prec M$ and M has the polynomial length contraction property than so does N. Moreover, if M_1 and M_2 have the PLCP, then $M_1 \times M_2$ has the PLP.*

Proof. Suppose that N is a submonoid of M and let ϕ be some n-input N-program. One can alternatively consider ϕ as an M-program and, since M has the PLCP, there exists a contraction ψ of ϕ whose length is bounded by $p(n)$. Now ψ is itself an N-program since all its instructions are instructions in ϕ.

If $N = \theta(M)$, then for any n-input N-program ϕ let ψ be a contraction (of length at most $p(n)$ of an M-program $\tau \in \theta^{-1}(\phi)$. For any $x \in A^n$, we have $(\theta(\psi))(x) = \theta((\psi)(x)) = \phi(x)$ and, since every instruction of ψ is an instruction of τ, every instruction of $\theta(\psi)$ is an instruction of $\theta(\tau) = \phi$. Hence, $\theta(\psi)$ is a contraction of ϕ of length at most $p(n)$.

Let $\phi = (i_1, f_1) \dots (i_s, f_s)$ be a program over $M_1 \times M_2$ with and let ϕ_{M_1} (resp. ϕ_{M_2}) be the program obtained from ϕ by replacing each f_i by $g_i : A \to M_1 \times M_2$ (resp. $h_i : A \to M_1 \times M_2$) defined as follows: for each $a \in A$, if $f_i(a) = (m_1, m_2)$, with $m_1 \in M_1$ and $m_2 \in M_2$ then $g_i(a) = (m_1, 1_{M_2})$ and $h_i(a) = (1_{M_1}, m_2)$. Of course, $\phi(x) = \phi_{M_1}(x) \cdot \phi_{M_2}(x)$ for all x. Now ϕ_{M_1} and ϕ_{M_2} can be viewed as programs over M_1 and M_2 respectively and can therefore be contracted to polynomial length ψ_{M_1} and ψ_{M_2} respectively. Clearly ϕ and $\psi_{M_1} \cdot \psi_{M_2}$ are equivalent. $\square$

As we will see, it is often convenient, in order to establish that N has the PLP, to prove that N divides some M having the PLCP.

On the other hand, say that M is *universal* if every language $L \subseteq A^*$ can be recognized by a family of M-programs, possibly of super-polynomial size. Such monoids exist of course and we will give many such examples. For instance, it is a simple exercise to build a branching program of width 3 for an arbitrary Boolean function on $\{0,1\}^n$. We also note:

Lemma 3.6 *The class of non-universal monoids is closed under division.*

Proof. Let N be a non-universal monoid. Let first M be a submonoid of N. Since any M-program is also an N-program, M cannot be universal either. Let next $M = \theta(N)$ for some surjective morphism θ. As we argued in the last section, every language recognized by an M-program can be recognized by an N-program so if N is non-universal, M cannot be. □

We do not know if non-universal monoids form a variety however because we are unable to prove yet that the class of non-universal monoids is closed under direct product.

Are universality and PLP related properties? It is easy to see that if M has the PLP then it certainly is not universal for there are doubly-exponentially many subsets of A^n but only exponentially many M-programs of length $p(n)$. In fact, we believe that the two notions are dual to one another and in the remainder of this section, we will argue in favor of the following conjecture.

Conjecture 3.7 *Let M be a finite monoid. The following are equivalent:*

1. *M has the polynomial length property;*
2. *M is non-universal;*
3. *M belongs to the variety* $\mathbf{DA} * \mathbf{G} \cap \overline{\mathbf{G_{nil}}}$.

We have already argued for $(1 \Rightarrow 2)$ and will show $(2 \Rightarrow 3)$. Our strongest indication that this conjecture is true is that this duality of universality and PLP holds for any monoid in $\mathbf{DS}$.

3.3.1 A Dichotomy Theorem for DS

We start with a generalization of Lemma 3.3.

Lemma 3.8 *If G is nilpotent, it has the polynomial length contraction property.*

Proof. Suppose G is nilpotent of class k with exponent m. We know that if $u, v \in G^*$ have the same number, mod m, of occurrences of any subword of length at most k then $eval_G(u) = eval_G(v)$. Let us consider an n-input G-program ϕ as a word over the alphabet $\Sigma = [n] \times G^A$ of possible instructions. By Lemma 2.12 there exists a word ψ over Σ of length at most $m \cdot |\Sigma|^{k \cdot m}$ and such that ψ and ϕ have the same number of occurrences of any subword of length at most k (mod m). Of course, ψ is just another G-program and we claim that for any input x we will have $\phi(x) = \psi(x)$. Indeed, any occurrence of a length t subword $g_1 \dots g_t$ of $\phi(x)$ (seen as a word in G^*) results from a subword $w \in \Sigma^*$ of length t in ϕ comprising the instruction which output these $g_1, \dots, g_t$ on input x. Because ϕ and ψ have the same number of occurrences of any subword in Σ^* of length at most k, then for each x, $\phi(x)$ and $\psi(x)$ will also have the same number of occurrences of any subword in G^* of length at most k and so $\phi(x) = \psi(x)$ on any x. Any ϕ thus has a contraction ψ of length $m \cdot |\Sigma|^{k \cdot m} = O(n^d)$. □

We can extend this proof to direct products of a commutative idempotent with a nilpotent group.

Lemma 3.9 *If M is in $\mathbf{J_1} \vee \mathbf{G_{nil}}$ then M has the PLCP.*

Proof. The proof is just a slight complication of the previous argument. Suppose M has exponent m and assume all its subgroups are nilpotent of class k. Consider an n-input M-program ϕ as a word over the alphabet $\Sigma = [n] \times M^A$

of possible instructions. By Lemma 2.12 there exists a word ψ over Σ of length at most $m \cdot |\Sigma|^k$ and such that ψ and ϕ have the same alphabet and the same number of occurrences of any subword of length at most $k \pmod m$. Now ψ is just another M-program and for any input x we will have $\phi(x) = \psi(x)$ since the words $\phi(x)$ and $\psi(x)$ (seen as words in M^*) will have the same alphabet and same number of subwords of length at most $k \pmod m$. $\square$

Considering congruences on the alphabet of possible instructions made sense in these examples because we could find a small representative of each congruence class. Similarly, we will exploit:

Lemma 3.10 *For any finite alphabet A of size n, any integer t, and any nilpotent group G of class k and exponent m, each $\sim^G_{n,t}$ class has a representative of size $O(n^k)$.*

Proof. Recall the definition of $\sim^G$: two words x, y are $\sim^G_{n,t}$-equivalent if they are G-equivalent, have the same alphabet and their a-left-decompositions (and same for a-right) x_0ax_1 and y_0ay_1 are such that x_0 and y_0 are $\sim^G_{n-1,t}$-equivalent and x_1, y_1 are $\sim^G_{n,t-1}$-equivalent. We will say that a position in x is a $\sim^G_{n,t}$-*bookmark* if it holds the occurrence of a such that x_0ax_1 is the a-right or a-left decomposition of x or if for some b-left-decomposition of $x = x'_0bx'_1$ it is a $\sim^G_{n-1,t}$-bookmark of x'_0 or a $\sim^G_{n,t-1}$ of x'_1 (or symmetrically for a b-right-decomposition). Note that for any x, the $\sim^G_{n,t}$ bookmarks are the same no matter what the group G is.

Let x be a word in A^*. We begin by marking certain special positions in x and define our marking scheme $S_{n,t}$ by induction on $n + t$ as follows. For $n + t = 1$, we do not mark any letter. For $n + t > 1$, we begin by marking the first and last occurrence of any letter in $\alpha(x)$. If we have marked an occurrence of a corresponding to the a-left decomposition $x = x_0ax_1$ we recursively mark x_0 using marking scheme $S_{|\alpha(x_0)|,t}$ and x_1 using $S_{n,t-1}$. We symmetrically mark recursively the segments defined by the a-right decomposition of x. A simple induction shows that the number of letters marked in $S_{n,t}$ is bounded by $O(n^t)$.

Of course, this scheme was tailor-made to mark all the occurrences of letters which are $\sim^G_{n,t}$-bookmarks in x. Suppose $x = uavbw$ where a, b are consecutive marked letters and let v' be such that $\alpha(v) = \alpha(v')$ and v and v' have the number modulo m of occurrences of any subword of length k or less. If we let $x' = uav'bw$ then $x \sim^G_{n,t} x'$ because v and v' are G-equivalent and because the $\sim^G_{n,t}$-bookmarks in x and x' coincide. We know that v' can be chosen of length $O(n^k)$ and since there are at most $O(n^t)$ in x that are delimited by two consecutive marked positions we can construct some word in the $\sim^G_{n,t}$-class of x that has length $O(n^{k+t})$. □

Lemma 3.11 *If M is in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ then M has the PLCP.*

Proof. Since M lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$, there exists a nilpotent group of class k and exponent m as well as a constant t such that any $u, v \in M^*$ with $u \sim^G_{|M|,t} v$ we have $eval_M(u) = eval_M(v)$. Consider an n-input M-program ϕ as a word over the alphabet $\Sigma = [n] \times M^M$ of possible instructions. By the previous Lemma, there exists a word ψ over Σ of length at most $m \cdot |\Sigma|^{k^s}$ and such that $\psi \sim^G_{|\Sigma|,t} \phi$.

For any input x, we will show that if ϕ and ψ are $\sim^G_{s,t}$-related then the words $\phi(x)$ and $\psi(x)$ in M^* can be shown $\sim^G_{s,t}$-related, where $s = |\alpha(\phi(x))|$, using induction on $s + t$. The case $t = 0$ has already been argued in the proof of Lemma 3.8. Fix x and consider the a-left decomposition of $\phi(x)$ for some $a \in \alpha(\phi(x))$. This a had to be output by some instruction querying bit i and applying some function f_j where $f_j(x_i) = a$. Thus, the (i, f_j)-left decomposition of ϕ is $\phi = \phi_0(i, f_j)\phi_1$ where $\phi_0(x)$ does not contain any a. There must exist a corresponding (i, f_j)-left decomposition of ψ as $\psi_0(i, f_j)\psi_1$ and such that $\psi_0(x)$ does not contain any a either. By induction we get $\phi_0 \sim^G_{s-1,t} \psi_0(x)$ and $\phi_1(x) \sim^G_{s,t-1} \psi_1(x)$ and by left-right symmetry $\phi(x) \sim^G_{s,t} \psi(x)$ as claimed.

We can therefore construct a polynomial length contraction of ϕ so M has the PLCP. $\Box$

At least within **DS**, we will show that $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ is the variety that exactly captures monoids having the PLP. We first state a result of [BST90]:

Lemma 3.12 *If G is a group which is not nilpotent, then it is universal.*

Proof. If G is not nilpotent, then for every n there is a commutator $h \in G_n$ that is not the identity. Fix such an h arbitrarily. Let L be an arbitrary subset of A^n. Let $f : A^n \to G_n$ be defined by $f(x) = h$ if $x \in L$ and $f(x) = 1$ otherwise. By Lemma 3.3, f can be realized by a G-program and L is recognized by this program and accepting subset $\{h\}$. $\Box$

In particular this lemma ensures universality for any monoid containing a non-nilpotent subgroup.

Lemma 3.13 *If M contains a non-flat regular $\mathcal{J}$-class then M is universal.*

Proof. If M contains a non-flat regular $\mathcal{J}$-class, there exist $\mathcal{J}$-related idempotents d, e, f in M with $d\,\mathcal{H}\,(ef)$, $ed = d$, $de = e$ and $df = d$ but $efe \neq e$.

Let L be an arbitrary subset of A^n. Fix a word $w \in L$ and consider the program $\phi = e \cdot (1, g_1) \ldots (n, g_n) \cdot fe$ where, for any $c \in A$, $g_i(c) = 1$ if $c = w_i$ and $g_i(c) = d$ otherwise. For any $x \neq w$, at least one instruction outputs a d and, since $d^2 = d$, $\phi(x) = edfe = e$. On the other hand $\phi(w) = efe$. Concatenating such programs for all elements of L, we get a program ψ with the property that $\psi(x) = e$ for $x \notin L$. On the other hand, if x does belong to L then exactly one of the segments will output efe and so we will get $\psi(x) = efe \neq e$. $\Box$

Combining the four previous lemmas, we obtain:

Theorem 3.14 *If M is in* **DS** *then it has the PLCP if it is in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ and it is universal otherwise.*

Proof. If M contains a non-nilpotent subgroup then it is universal by Lemma 3.12 and if it is in $\mathbf{DS} - \mathbf{DO}$ then it contains a simple non-orthodox subsemigroup and is thus universal by Lemma 3.13. If M lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ however, then it has the PLCP by Lemma 3.11. $\square$

It follows that programs over monoids in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ cannot compute, for instance, the word problem of a non-nilpotent group. Furthermore, the function MOD_q cannot be computed by any program over $\mathbf{DO} \cap \overline{\mathbf{G_p}}$ where p and q are distinct primes. This follows from the observation that such programs can be contracted and then simulated with bounded depth $\{AND, OR, MOD_p\}$-circuits of polynomial size. The latter cannot recognize MOD_q [Smo86]. We conjecture that, similarly, MAJORITY cannot be computed by any program over $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$.

3.3.2 Some Results in $\mathbf{DA} * \mathbf{G}$

Can we find a similar dichotomy for monoids outside of $\mathbf{DS}$? The following result, originally proved in [Thé89], restricts quite dramatically the space of candidates for the PLP. Recall that the monoid U, which we introduced in Section 2.1.6, is the syntactic monoid of the language $\{a, b\}^* bb \{a, b\}^*$.

Lemma 3.15 *U is universal.*

We sketch this proof for completeness.

Proof. For any $w \in A^n$, consider the program

$$\phi_w = ab(1, f_1)(2, f_2) \dots (n, f_n)b$$

where $f_i(c) = a$ if $c \neq w_i$ and $f_i(c) = 1$ otherwise. Note that since a is idempotent in U, we get $\phi_w(x) = abab = ab$ if $w \neq x$ and $\phi_w(x) = abb = 0$ if $w = x$. So for any $L \subseteq A^n$, the program ϕ of length $n \cdot |L|$ consisting of the concatenation of all programs ϕ_{w_i} where $w_i \in L_i$ is such that $\phi(x) = 0$ if and only if $x \in L$. $\square$

As a corollary, we thus obtain implication $(2 \Rightarrow 3)$ of conjecture 3.7.

Theorem 3.16 *If M does not belong to the variety $\mathbf{DA} * \mathbf{G} \cap \overline{\mathbf{G_{nil}}}$ then M is universal.*

Proof. Recall from Chapter 2 that if $M \notin \mathbf{DA} * \mathbf{G}$ then there exist two idempotents $e, f \in M$ such that $e \mathcal{J} f \mathcal{J} (ef)$ but ef is not idempotent. If $(ef)^\omega$ also is $\mathcal{J}$-related to e and f then this $\mathcal{J}$-class is non-flat and the universality follows from Lemma 3.13. Otherwise, since $(ef)\mathcal{J}e$, there must be an idempotent s in the $\mathcal{H}$-class $\mathcal{L}_e \cap \mathcal{R}_f$. Thus, U is a divisor of M and we appeal to Lemma 3.15. $\square$

Therefore, universality and PLP questions only remain open for the variety $\mathbf{DA} * \mathbf{G}$.

Theorem 3.17 *If M is of the form $N \circ G$ with $N \in \mathbf{DA}$ and $G \in \mathbf{G_p}$ then M has the PLP.*

Our proof of this theorem is unfortunately quite technical and its main idea is best illustrated in the following example which we prove as a warm-up.

Example 3.17. *Claim: the monoid $M = U_1 \circ C_p$ has the PLP.*

We assume for simplicity that the input alphabet is $\{0, 1\}$. By Lemma 3.4, we know that for any n-input program ψ over a p-group G such as C_p, there exists a polynomial r in $Z_p[X_1, \dots X_n]$ of degree at most d_{C_p} such that $r = 1$ whenever ψ outputs $h \in F \subseteq C_p$ and $r = 0$ otherwise.

Let us denote by the pair $(\bar{f}_i, \bar{g}_i) \in (U_1^{C_p}, C_p)$ the product of the first i instructions of ϕ on some input. If (f_i, g_i) is the result of the i^{th} instruction, we have

$$(\bar{f}_i, \bar{g}_i) = (\bar{f}_{i-1} f_i^{\bar{g}_{i-1}}, \bar{g}_{i-1} g_i)$$

We will say that instruction i is an *h-crash site* for $x \in \{0,1\}^n$ if on x we have $\bar{f}_{i-1}(h) = 1$ but $\bar{f}_i(h) = 0$. In particular, this implies that $f_i^{\bar{g}_{i-1}}(h) = 0$.

Our main claim is that for all h in C_p there can only be polynomially h-crash sites querying bit X_1. Consider in particular all such instructions such that a crash occurs on an input where $X_1 = 1$. Since the value $\bar{g}_{i-1}$ is computed by a C_p program, we know that there exists a fixed degree $\mathbb{Z}_p$-polynomial r_i associated with this instruction such that $f_i^{\bar{g}_{i-1}}(h) = 0$ if and only if $r_i = 1$. There are only $\binom{n}{d_{C_p}}$ many linearly independent such r's, hence if we have more than $\binom{n}{d_{C_p}}$ crash sites it must be the case that some r_i can be expressed as a linear combination of r_j's with $j < i$. Hence if $r_i = 1$, there must be $j < i$ with $r_j = 1$. This shows that i is actually not a crash site since whenever $X_1 = 1$ and $f_i^{\bar{g}_{i-1}}(h) = 0$ we already had $\bar{f}_j(h) = 0$.

Therefore, we have at most $p \cdot 2 \cdot n \cdot \binom{n}{d_{C_p}}$ instructions which are crash sites, i.e. where the $U_1^{C_p}$ part of the computation is truly active. For all but polynomially many instructions in ϕ, we can thus replace the $U_1^{C_p}$ component of the instruction by the identity without affecting the result of our computation. In between any two potential crash sites, we are thus left with subprograms over the subgroup C_p but these can be made to have polynomially bounded length using Lemma 3.8.

We now extend the same idea, at the expense of technical complications, to prove the full version of Theorem 3.17.

Proof. Let ϕ be an n-input M-program of length s. As in the example above, we will begin by identifying a polynomial number of key instructions in ϕ and argue that for all inputs x the N^G-component of $\phi(x)$ can only be affected at one of these locations. Suppose that N divides $N^*/\sim_{|N|,k}$ and let us again denote by $(\bar{f}_i(x), \bar{g}_i(x)) \in (N^G, G)$ the product of the first i instructions of ϕ on input x and by $f_{i,x} \in N^G$ the N^G component of the output of the i^{th} instruction on x. For an element $g \in G$, we will say that the i^{th} instruction of ϕ is *h-critical* if there exists an input x such that in the following word in N^*:

$$f_{1,x}(h)f_{2,x}^{\bar{g}_1}(h)\dots f_{i,x}^{\bar{g}_{i-1}}(h)\dots f_{s,x}^{\bar{g}_{s-1}}(h)$$

the i^{th} letter is a $\sim_{|N|,k}$ bookmark.

We claim that for each $h \in G$ there are only polynomially many h-critical instructions in ϕ. First, the number of bookmarks in a given word over N^* is bounded by some constant c depending on N and k and so, for a given x, only c input letters are ever queried by the corresponding h-critical instructions. Keeping the analogy with our example, let us consider the set R of all x's for which the h-critical instructions were the result of querying $x_1, \dots, x_c$ and finding them holding, say, 1. Now, the $\bar{g}_i$ are computed by a program over some group in $\mathbf{G_p}$ and so, by Lemma 3.4, for any h-critical instruction producing the bookmark, say b_1 there exists a polynomial r_i in $\mathbb{Z}_p$ with fixed degree such that $f_i^{\bar{g}_{i-1}}(h) = b_1$ if and only if $r_i = 1$.

For any $w \in R$, let $l_1, \dots, l_c$ be the locations at which the h-critical instructions appear and say they hold letters $b_1, \dots, b_c$. There exists a constant degree polynomial q_x over $\mathbb{Z}_p$ such that positions $l_1, \dots, l_c$ hold $b_1, \dots, b_c$ if and only if $q_w(x) = 1$ and $q_w(x) = 0$ otherwise. There are only polynomially many linearly independent such q_w so if R is too large, we can find $w_1, w_2 \in R$ such that $q_{w_1} = q_{w_2}$.

In this case, the two words of N^*

$$u_1 = f_{1,w_1}(h)f_{2,w_1}^{\bar{g}_1}(h)\dots f_{i,w_1}^{\bar{g}_{i-1}}(h)\dots f_{s,w_1}^{\bar{g}_{s-1}}(h)$$

and

$$u_2 = f_{1,w_2}(h)f_{2,w_2}^{\bar{g}_1}(h)\dots f_{i,w_2}^{\bar{g}_{i-1}}(h)\dots f_{s,w_2}^{\bar{g}_{s-1}}(h)$$

are such that they agree on every letter which is a $\sim_{|N|,k}$ bookmark of u_1 or a $\sim_{|N|,k}$ bookmark of u_2. These locations must coincide. Thus, only polynomially many different instructions in the program can be found h-critical because of some $x \in R$. Since there are only polynomially many possibilities for R we have proved our claim.

The N^G component of $\phi(x)$ can truly be affected only at one of the polynomially many critical instructions. In all other instructions, we can replace the N^G component of the computation by the identity without affecting the output of the program. The resulting segments in between two critical instructions can be viewed as G-programs and thus be contracted to polynomial length using Lemma 3.11 since G is nilpotent. $\Box$

These constructions do not imply the existence of polynomial length contractions of arbitrary programs over $\mathbf{DA} * \mathbf{G_p}$ and it is quite possible that some monoids in this variety do not have the PLP. They are, however, divisors of a monoid which has the PLP and so no monoid in $\mathbf{DA} * \mathbf{G_p}$ is universal. In particular, a theorem of Thérien [Thé89] building on the work of Smolensky [Smo86] shows that, regardless of length, no program over a monoid in $\mathbf{J_1} * \mathbf{G_p}$ can compute the function MOD_q for any primes $p \neq q$. In fact, Thérien's argument can easily be extended to any variety of the form $(\mathbf{G_p} \,\Box\, (\mathbf{G_p} \,\Box\, (\ldots \Box\, \mathbf{G_p} \,\Box\, (\mathbf{J_1} * \mathbf{G_p}) \ldots)))$. By the results of [PST88], any monoid in $\mathbf{DA} * \mathbf{G_p}$ is in one of these varieties and thus no program over a monoid in $\mathbf{DA} * \mathbf{G_p}$ can compute the function MOD_q for any primes $p \neq q$.

Using a different argument, we will next show that no program over a Brandt monoid can compute the function MOD_q for any q. Although this is a strictly weaker result than the ones just mentioned, we believe that the novel proof technique could help in proving non-universality of other aperiodic monoids with similar properties.

Theorem 3.18 *If M is a Brandt monoid, then no M-program can compute MOD_m for any integer $m \geq 2$.*

Proof. Suppose that ϕ is a B_k-program computing MOD_m for inputs of length $n \geq 2m+1$. We can assume without loss of generality that the output of each instruction is either a generator or the identity and that there exists[2] $x \in \{0,1\}^n$ such that $\phi(x) = 0$. Note that a string w in $\{1, a_1, a_2 \dots a_k\}^*$ evaluates to 0 in B_k if and only if for some $j \neq i+1$, it contains occurrences of a_i and a_j separated only by 1's and, in particular, w evaluates to 0 if there exists i, j such that $|w|_{a_i} - |w|_{a_j} > 1$. Thus if $\phi(x) = 0$, we can find instructions $s < t$ querying (not necessarily distinct) bits b_s, b_t and producing a_i, a_j respectively while all instructions in between them output the identity. Let ψ be the subprogram of ϕ consisting of instructions between s and t and suppose that $x \in MOD_m$. For any x' at Hamming distance 1 from x, we have $x' \notin MOD_m$ and thus $\phi(x') \neq 0$. Assuming $i < j$, this means that if any one bit of x other than b_s or b_t is flipped then ψ must now output a word $w \in \{1, a_1, a_2 \dots a_k\}^*$ such that for all[3] $i < l < j$ we have $|w|_{a_l} - |w|_{a_i} = 1$. However, if we now flip c bits of x other than b_s or b_t, the output of ψ contains c more occurrences of a_l than a_i and so the program's output is 0 again. This is a contradiction for unless c is a multiple of m we should have $\phi(x') \neq \phi(x)$.

This argument can clearly be adapted to handle the case where $x \notin MOD_m$.

□

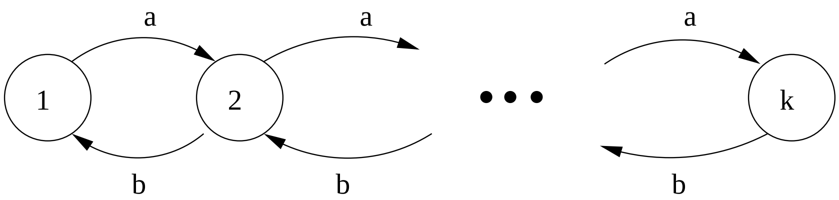
The above proof actually shows that programs over Brandt monoids have very limited ability to compute symmetric functions. In particular, they cannot compute $\textsc{Threshold}_t$ unless t or $n - t$ is a constant.

The proof can also be adapted to obtain similar limits on the power of programs over the transition monoids M_k associated to the following finite automata:

---

[2] If $\phi(x) \neq 0$ for all $x \in \{0,1\}^n$ then $\phi(x)$ is completely determined by the output of the first and last instructions whose output is not 1.

[3] If $j < i$, we want to consider all l *except* those between j and i.



The M_k's were studied in [Thé89] as examples of non-universal aperiodic monoids of dot-depth k. Brandt monoids, as well as the M_k's are inverse aperiodics but have the very special property that they lie in $\mathbf{J_1} * \mathbf{G_p}$ for some prime p. Still, the idea behind our last proof is that as soon as a program ϕ over B_k or M_k is doing non-trivial computation, then for a vast majority of inputs we have $\phi(x) = 0$. Intuitively, all inverse aperiodic monoids have this property.

3.3.3 Open Problems

Do Brandt monoids have the PLP? Intuitively, the fact that each of them is a divisor of a monoid which does have the PLP leads us to believe so, but even the case of B_2 has so far eluded proof. This is the subject of ongoing work with K. Reinhardt and D. Thérien.

Another outstanding problem concerns the power of program over monoids which lie in $\mathbf{DA} * \mathbf{G}$ but not in $\mathbf{DA} * \mathbf{G_p}$ for any prime p. We conjecture that Lemma 3.17 can be extended to show that monoids of the form $N \circ G$ where N is in $\mathbf{DA}$ and G is a nilpotent group have the PLP. A first step would be to show that no monoid in $\mathbf{DA} * \mathbf{G_{nil}}$ is universal. As of yet, we have no proof that even $U_1 \circ C_6$ is non-universal. We believe that resolving these two problems are key steps towards a possible proof of our conjecture.

3.4 Crane Beach Properties and Program Varieties

How can we classify monoids in terms of their computational power? If we are using morphisms to recognize languages, then we have seen that varieties are the natural unit of classification. In the case of programs over monoids however, any two non-solvable monoids recognize exactly languages in non-uniform NC^1. For a variety of monoids $\mathbf{V}$, let us denote by $\mathcal{P}(\mathbf{V})$ the class of languages which can be recognized by a polynomial length program over some monoid in $\mathbf{V}$.

Theorem 3.19 ([MPT91]) *If $\mathbf{V}, \mathbf{W}$ are varieties of monoids $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ if and only if $\mathcal{P}(\mathbf{V})$ and $\mathcal{P}(\mathbf{W})$ contain the same regular languages.*

One might expect that for some relatively weak and robust varieties the regular languages in $\mathcal{P}(\mathbf{V})$ coincide with the regular languages with syntactic monoids in $\mathbf{V}$. This intuition is unfortunately incorrect in most cases because a lot of computation can be hardwired into the program itself. Consider for instance the regular language $L \subseteq \{a, b\}^*$ consisting of words that hold an a in some even-indexed position. In order to recognize membership in L, an automaton must have a mechanism that keeps track of the parity of the number of input letters read so far and one can easily see that $M(L)$ correspondingly contains the group C_2. On the other hand, we can write a program over U_1 that recognizes L by making sure that only the even-indexed positions of the input are queried by the program. Similarly, the language of words that have an even number of a's beyond the first ten positions can be recognized by a program over C_2 even though its syntactic monoid is not a group.

We say that a language $L \subseteq A^*$ has a *neutral letter* if there is a letter $e \in A$ such that for any $u, v \in A^*$ we have $uv \in L$ if and only if $uev \in L$. In other words, the letter e is neutral if and only if e is equivalent to the empty word in the syntactic congruence of L. For any language $L \subseteq A^*$, we denote by $L^\epsilon \subseteq (A \cup \{e\})^*$, where $e \notin A$, the language with neutral letter e such that $u \in A^*$ lies in L^ϵ if and only if u lies in L. Note that L and L^ϵ have the same syntactic monoids.

Intuitively, a program computing L^ϵ cannot exploit its ability to look for specific input letters appearing in specific input positions.

Theorem 3.20 *If* $\mathbf{V}, \mathbf{W}$ *are varieties of monoids* $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ *if and only if* $\mathcal{P}(\mathbf{V})$ *and* $\mathcal{P}(\mathbf{W})$ *contain the same regular languages with neutral letter.*

We say that a variety $\mathbf{V}$ of monoids is a *program-variety* (or P-variety) if the regular languages with neutral letter lying in $\mathcal{P}(\mathbf{V})$ have their syntactic monoid in $\mathbf{V}$. This is equivalent to the requirement that for any monoid M, if all sets $T_m \subseteq M^*$ with $T_m = \{w | eval_M(w) = m\}$ can be recognized by polynomial length programs over monoids in $\mathbf{V}$ then in fact $M \in \mathbf{V}$. In light of the above theorem, program-varieties are the natural unit of classification of monoids in terms of their power as language recognizers via programs. Of course, most varieties are not program varieties.

Theorem 3.21 (see [Str00]) *The varieties of all finite monoids* $\mathbf{M}$*, p-groups* $\mathbf{G_p}$*, aperiodics* $\mathbf{A}$*,* $\mathcal{J}$*-trivial monoids* $\mathbf{J}$*, commutative monoids* $\mathbf{Com}$ *all are program-varieties.*

Of course, Theorem 3.2 shows that "$\mathbf{A}$ is a program variety" and "MOD_p does not belong to AC^0 for any p" are equivalent statements. Many fundamental circuit complexity questions can similarly be rephrased in this way. Showing that AND does not belong to CC^0 for instance, is equivalent to showing that $\mathbf{G_{sol}}$ is a program-variety.

Showing that $\mathbf{Com}$ is a program variety is a simple exercise. In fact, one can establish an even stronger statement about the languages with neutral letter that programs over commutative monoids can compute.

Example 3.21. Suppose that $L \subseteq A^*$ is a language with a neutral letter, say e, that can be recognized by a program $\phi = (\phi_0, \phi_1, \ldots)$ over a commutative monoid M of threshold t and exponent p. As we have seen in the previous section, we can assume that each ϕ_n consists of n instructions each querying a different input letter. Let u, v be two words in Σ^* such that $\alpha_{t,p}(u) = \alpha_{t,p}(v)$.

Note that the lengths of u and v must be equal modulo p. We can thus pad u and v with e's to obtain words u' and v' of equal length k and with $\alpha_{t,p}(u') = \alpha_{t,p}(v')$.

Consider the s-input program ϕ_s with $s = k \cdot |M|^{|A|}$. Because there are only $|M|^{|A|}$ possible query-functions, there must be a set of positions $I \subseteq [s]$ of size k such that the corresponding instructions in ϕ_s all have the same query-function f. We denote by u'' (resp. v'') the word of length s obtained by placing the letters of u' (resp. v') in the k positions of I and placing neutral letters e in the other $s - k$ positions. We claim that $\phi_s(u'') = \phi_s(v'')$.

Indeed, since u'' and v'' agree on all positions outside I, it is sufficient to show that we have

$$\alpha_{t,p}(\phi_s^I(u'')) = \alpha_{t,p}(\phi_s^I(v''))$$

where $\phi_s^I(w)$ denotes the output in M^* of the instructions of ϕ_s querying positions in I. Since the positions I in u'' hold the word u' and since the corresponding instructions use the same query-function f, the number of occurrences of m in $\phi_s^I(u'')$ is just

$$\sum_{a \in A} |u'|_a \cdot |f(a)|_m.$$

Since we have $\alpha_{t,p}(u') = \alpha_{t,p}(v')$ we have for all $m \in M$

$$\sum_{a \in A} |u'|_a \cdot |f(a)|_m \equiv \sum_{a \in A} |v'|_a \cdot |f(a)|_m (\text{thresh } t, \text{mod } p)$$

which proves our claim.

Thus $\phi(u'') = \phi(v'')$ and so u'' and v'' are either both in L or both in $\bar{L}$. Because u and v can be obtained 'from u'' and v'' respectively by deleting neutral letters, they are also either both in L or both in $\bar{L}$. Hence the syntactic congruence of L is coarser than the congruence induced by $\alpha_{t,p}$ so $M(L)$ is a finite commutative monoid.

The *Crane-Beach Conjecture*, first postulated by D. Thérien and C. Lautemann, stated that all languages with a neutral letter recognized by AC^0 circuits were in fact star-free languages or, equivalently, that all languages with a neutral letter recognized by a polynomial length program over an aperiodic monoid had a finite and aperiodic syntactic monoid. The Crane-Beach Conjecture was disproved by N. Immerman [BIL$^+$01].

We will say that programs over a monoid variety $\mathbf{V}$ have the *Crane-Beach property* if any language with neutral letter recognized by a polynomial length program over a monoid in $\mathbf{V}$ has its syntactic monoid in $\mathbf{V}$. By definition, every such variety is a program variety although the example of the variety $\mathbf{A}$ shows that the converse does not hold in general.

Theorem 3.22 *Programs over* $\mathbf{J}$ *have the Crane-Beach property.*

Proof. This can be obtained as a corollary of Theorem 3.11 of [BIL$^+$01] where it is shown that every language with a neutral letter which can be defined by a Boolean combination of Σ_1-sentences using arbitrary numerical predicates is in fact regular and has a syntactic monoid in $\mathbf{J}$. Every language recognized by a family of polynomial-length programs over $\mathbf{J}$ is in fact definable in this way and the result is not difficult to obtain once the logical framework has been precisely defined.

However, we want here to prove this directly from the programs, in the spirit of Example 3.4. Unfortunately, we will only show this for the following special case. Let M be $\mathcal{J}$-trivial: we say that a family of M-programs $(\phi_n)_{n\geq 0}$ recognizing a language L that contains a neutral letter e is *silent* if for every n we have $\phi_n(e^n) = 1_M$. We claim that in this case L is regular and $M(L)$ lies in $\mathbf{J}$.

Let k be minimal such that any two words in M^* in which the same subwords of length k or less appear evaluate to the same element (note that it is sufficient to consider the subwords over the alphabet $M - \{1_M\}$. It suffices to show that any two words u, v in A^* that have the same occurrences of subwords of length at most k are either both in or both not in L. Because the neutral letter allows various forms of padding, we can assume without loss of generality that u and v have the same length $l \geq k$.

Note that if a subword $t_1 \dots t_s$ with $s \leq k$ occurs in $\phi_n(x)$, then each t_i is the output of an instruction querying only one position in x. In particular, there is an s-tuple of positions in x such that for any y agreeing with x on these positions $\phi_n(y)$ also contains the subword $t_1 \dots t_s$.

For the program ϕ_n, given k-tuple of input positions $(x_1, \dots, x_k)$, with the $x_i \in [n]$ written in increasing order, and given assignment $(a_1, a_2, \dots, a_k)$ to these positions, we denote by $W^{(x_1,\dots,x_k)}_{(a_1,\dots,a_k)}$ the set of subwords in $(M - \{1_M\})^*$ of length at most k in $\phi_n(q)$ where q holds a_i in position x_i and neutral letters everywhere else. Note that since we assumed that the program is silent every instruction querying a position holding a neutral letter outputs 1_M and so every such subword results from instructions querying one of the x_i. In particular, any subword of $\phi_n(q)$ will also occur in $\phi_n(y)$ for any word y holding a_i in position x_i.

We color k-tuples of positions $(x_1, \dots, x_k)$ with the sets

$$\{W^{(x_1,\dots,x_k)}_{(a_1,\dots,a_k)} | (a_1, \dots, a_k) \in A^k\}.$$

There are only finitely many colors of course since $|A|$, k and $|M|$ are all fixed. Thus, by Ramsey's Theorem, there exists n such that we can find a set $I \subseteq [n]$ of size l and such that any k-tuple from I is labeled with the same color. We will call these l positions *special positions.*

Let u', v' be the words of length n obtained by placing respectively u and v in the l special positions and neutral letters at all other positions. We will now compare the set of subwords in $(M - \{1_M\})^*$ of length at most k occurring in $\phi_n(u')$ and $\phi_n(v')$.

For any occurrence of the subword $t_1 \dots t_s$, with $t_i \in M - \{1_M\}$ and $s \leq k$, occurring in $\phi_n(u')$, we can find a k-tuple of special positions $x_1 < x_2 < \dots < x_k$ such that each t_i is the output of an instruction querying one of these positions. Let us suppose these positions hold the letters $a_1, a_2, \dots, a_k$. Thus u contains the subword $a_1 \dots a_k$ and so v also does. Therefore, we can find k special positions $y_1 < y_2 < \dots < y_k$ in v' which hold $a_1, a_2, \dots, a_k$.

The subword $t_1 \dots t_s$ belongs to $W^{(x_1,\dots,x_k)}_{(a_1,\dots,a_k)}$ and since the tuples $(x_1, \dots, x_k)$ and $(y_1, \dots, y_k)$ were assigned the same color, we must have $t_1 \dots t_s$ belongs to $W^{(y_1,\dots,y_k)}_{(a_1,\dots,a_k)}$. Since v' holds a_i in position y_i, the subword $t_1 \dots t_s$ occurs in $\phi_n(v')$.

Therefore, $\phi_n(u')$ and $\phi_n(v')$ contain exactly the same subwords of length at most k which implies $\phi_n(u') = \phi_n(v')$ and, in turn, that u and v are either both in L or both not in L. $\square$

The above argument will fail when the programs are not silent although it is reasonnable to believe that this technical difficulty can be addressed by either refining the coloring or showing that every program over a $\mathcal{J}$-trivial monoid is equivalent to one which is silent. The following was also established using similar Ramsey-theoretical tools by C. Lautemann and D. Thérien:

Theorem 3.23 ([LT01]) *Programs over* $\mathbf{G_{nil,k}}$ *have the Crane-Beach property.*

Note that in the context of groups the programs *can be* assumed "silent" without loss of generality because of the presence of inverses.

We will show in Section 4.4 that varieties $\mathbf{DA}$, $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ and $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ are program-varieties using communication complexity results. It is unclear at this time whether the convenient combinatorial descriptions we have of regular languages recognized by such monoids can be combined with extremal combinatorics to further show that these have the Crane-Beach property.

Do programs over solvable groups have the Crane-Beach property? If they do, then CC^0 is a strict subset of NC^1 and we cannot reasonably expect a simple proof of this fact. However, although it is widely believed that $\mathbf{G_{sol}}$ forms a program-variety, it is possible that the Crane-Beach property fails for $\mathbf{G_{sol}}$ much like it does for aperiodic monoids. Providing an explicit example of a language L with neutral letter that can be recognized by polynomial size CC^0 circuits but such that $M(L)$ is not in $\mathbf{G_{sol}}$ would be of great interest.

We should note that a related result of D. Barrington and H. Straubing [BS95] shows that any language with neutral letter recognized by an M-program of length $o(n \log \log n)$ is regular and has a syntactic monoid dividing the direct product of a number of copies of M and M^r, where M^r is the *reverse monoid* of M in which multiplication by $s \cdot_{M^r} t = t \cdot_M s$.

Chapter 4

Communication Complexity

4.1 Introduction

The need for efficient communication is omnipresent in modern computer science. It is a natural concern in distributed computing, networking, computer architecture, cryptography but although the formal study of communication complexity, beginning in the late 70's, was originally motivated by such practical concerns (note the title of Yao's seminal paper [Yao79]) its later development has mostly served as a surprisingly versatile tool in just about every area of theoretical computer science. The game at the heart of communication complexity is the following: Alice and Bob are given inputs x and y respectively and want to collaborate to compute a function $f(x, y)$ while minimizing the communication that they need to exchange. Many variants of it can be defined: non-deterministic, probabilistic, round-bounded, approximate and so-on. A. Chandra, M. Furst and R. Lipton also introduced in [CFL83] an interesting multiparty extension of the usual Alice and Bob model. In their game, k players collaborate to compute $f(x_1, \ldots, x_k)$ but each player is given access to all but one of the x_i's. This model gives rise to subtle combinatorics and has also found many applications to other areas of complexity theory.

In many cases, it is possible to *upper bound* the communication complexity (in an appropriate model) of functions which can be computed using a limited amount of resources so that obtaining *lower bounds* on the communication complexity (in the same model) of an explicit function f translates into lower bounds for the resources needed to compute f. This approach has yielded results in VLSI (see [Lov89]), lower bounds for monotone circuits [KW88] including a complete separation of the monotone NC hierarchy [RM97], Time-Space trade-offs for Turing machines [BNS92, KN97] to cite only a few. Communication complexity is also the main tool used in the study of branching programs: the book of Wegener [Weg00] provides a complete overview of the theory of branching programs and OBDD's with an emphasis on communication complexity and more recent results include [BSSV00, BV02]. Perhaps even more significant in the context of this thesis are the applications to lower bounds for classes of circuits lying within NC^1, mainly threshold circuits [HG90, Nis94, ROKY94], but also CC^0 circuits [Gro92, Gro94b] and ACC^0 circuits [HG90, Lok01]. As we will see, some of these results can be rephrased in algebraic terms using the circuits/monoids correspondence offered by programs.

Another rather unexpected link between communication complexity and monoids was uncovered by Szegedy [Sze93] who showed that a language has bounded two-party deterministic communication complexity if and only if it can be recognized by a program over a commutative monoid. This amazing result is strong indication that an algebraic point of view on communication complexity can be fruitful.

In [BFS86], Babai, Frankl and Simon built a complexity theoretic view of communication and formally introduced notions of complexity classes, reductions and completeness in a (two-party) communication complexity context. Their goal was twofold: this provides, on one hand, a natural yet powerful framework to compare the power of different extensions or restrictions of the usual deterministic model and understand the complexity of concrete functions while building, on the other hand, a rich structure of classes in which we can hopefully gain intuition on the nature of non-determinism, alternation, counting and so on. This "world picture" of communication complexity classes was further described in various papers [HR90, DKMW92] and it should be noted that there exist regular languages which are complete for many of these complexity classes, including the communication complexity analogues of NP, $\oplus$P, PSPACE, etc. Whereas regular languages are the simplest languages from a classical Time/Space complexity point of view, they can have large communication complexity even in quite powerful models. In fact, some of the most studied languages in communication complexity (Disjointness, Inner Product mod p), although not regular languages themselves, are *equivalent* from a communication complexity perspective to regular languages.

4.1.1 Summary of Results

This chapter develops an algebraic approach to communication complexity. On one hand, this point of view allows us to use properties of finite monoids to understand the limits of various communication complexity models and compare their relative power and, on the other hand, it provides a systematic way of using communication complexity to understand the computational limits of programs over monoids.

We first consider the well known deterministic two-party model as well as its simultaneous, probabilistic, simultaneous probabilistic, and MOD$_p$-counting variants. We set out to answer the following question: what is the communication complexity, in a worst-case partition sense, of any regular language in each of these models? Specifically, we look at the complexity of determining if the word $a_1b_1a_2b_2\dots a_nb_n$ is a member of a given regular language $L \subseteq \Sigma^*$ where the $a_i \in \Sigma \cup \{\epsilon\}$ are known to Alice and the $b_i \in \Sigma \cup \{\epsilon\}$ are known to Bob. It was established in [RTT98] that, in these models, regular languages having communication complexity $O(f)$ for some $f : \mathbb{N} \to \mathbb{N}$ form a variety of languages so our question *has* an algebraic answer. In Section 4.2, we use algebraic tools to completely characterize the communication complexity of any regular language in the deterministic, probabilistic, simultaneous, probabilistic simultaneous and MOD_p-counting (for prime p) models. Remarkably, our classifications feature in all five cases complexity gaps. For instance, we find that a regular language L has deterministic communication complexity either $O(1)$ (when its syntactic monoid is commutative), $\Theta(\log n)$ (when $M(L)$ lies in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ but is not commutative, i.e. when $M(L)$ is not commutative but L is the disjoint union of unambiguous concatenations of the form $L_0 a_1 L_1 \dots a_k L_k$ with $M(L_i)$ commutative) and $\Theta(n)$ otherwise. This is in sharp contrast with the general case where for an arbitrary $f : \mathbb{N} \to \mathbb{N}$ with $1 \leq f(n) \leq n$ it is easy to artificially construct a non-regular language of complexity $\Theta(f)$.

In all four variants of the deterministic model, we find that communication complexity induces classifications with only a small number of classes: a regular language L has simultaneous complexity either $O(1)$, $\Theta(\log n)$ or $\Theta(n)$, probabilistic complexity $O(1)$, $\Theta(\log\log n)$, $\Theta(\log n)$ or $\Theta(n)$, probabilistic simultaneous complexity $O(1)$, $\Theta(\log n)$ or $\Theta(n)$, and, for any prime p, MOD_p-counting complexity $O(1)$, $\Theta(\log n)$ or $\Theta(n)$. Moreover, some of these classes are related in unexpected ways: a regular language has $O(\log n)$ probabilistic complexity only if it has $O(\log n)$ deterministic complexity and further has $\Theta(\log\log n)$ probabilistic complexity if and only if it has $\Theta(\log n)$ simultaneous complexity if and only if it has $\Theta(\log n)$ probabilistic simultaneous complexity. In fact, we prove that the simultaneous and probabilistic simultaneous complexities of any regular language are equal, up to a constant. We also find that a regular language has MOD_p-counting and MOD_q-counting complexity $O(\log n)$ for distinct primes p, q if and only if it has deterministic complexity $O(\log n)$. All varieties involved in these classifications, some of which have already been shown to be of importance in previous chapters, have convenient descriptions both algebraically and combinatorially and are decidable. In obtaining these results we amazingly use communication complexity reductions to and from only four (well-known) problems: Disjointness, Inner Product modulo p, Greater Than and Index and this retrospectively both highlights and explains their importance as fundamental examples in communication complexity theory.

In Section 4.3 we consider the tricky multiparty communication complexity model. As in the two-party case, we set out to describe the multiparty complexity of each regular language and show that this question, as in the two-party case, has an algebraic answer. We are able to prove that any regular language L recognized by a group has k-party complexity $O(1)$ if $M(L)$ is nilpotent of class $k-1$ and k-party complexity $\Theta(n)$ otherwise. The general case, however, seems very challenging and we can only prove partial results. Most notably, we show that there exists a k such that the regular language L has k-party complexity $O(1)$ if and only if $M(L)$ lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ and give a characterization of regular languages with constant three-party communication complexity. The techniques used in these proofs are a combination of Ramsey theoretical arguments akin to the ones of [CFL83] and probabilistic techniques using the discrepancy method in the line of [BNS92, Gro93].

Our results shed an interesting light on a poorly understood, yet important, communication model and identify problems, such as the multiparty communication complexity of piecewise testable languages, as natural targets for further research in the field. We also argue that our results might be a first step towards an analog of Szegedy's Theorem which would provide an algebraic characterization of functions with bounded k-party communication complexity.

In Section 4.4, we discuss the impact of our communication complexity bounds on issues surveyed in Chapter 3. In particular we use a very general communication complexity argument to show that $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ and $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ are program-varieties. We further give a new proof of an exponential lower bound for the length of $(\mathbf{G_p} * \mathbf{Ab})$-programs computing Disjointness and propose a communication complexity conjecture in the same spirit which would separate the computing power of polynomial length $(\mathbf{G_p} * \mathbf{G_{nil,k+1}})$-programs and polynomial length $(\mathbf{G_p} * \mathbf{G_{nil,k}})$-programs.

4.2 Two-party Communication Complexity

Is is hard to overstate the quality of [KN97] as an introduction to communication complexity and we refer the reader to it for further details on the concepts introduced throughout this section. We will use that book's notation.

4.2.1 Two-party Models

In the deterministic model, two players, Alice and Bob, wish to compute a function $f : S^{n_A} \times S^{n_B} \to T$ where S and T are finite sets. Alice is given $x \in S^{n_A}$ and Bob $y \in S^{n_B}$ and they collaborate in order to obtain $f(x, y)$ by exchanging bits (using, say, a common blackboard) following the format imposed by a previously agreed upon *communication protocol* $\mathcal{P}$.

It is convenient to think of a protocol in an informal way as a scheme ensuring that Alice and Bob will never speak simultaneously and will be able to make sense of the information they send each other. Intuitively, $\mathcal{P}$ determines, at every stage, whether the current run of the protocol is over and if not, whose turn it is to write the next bit. This is a function of the communication written thus far but is independent of the players' inputs. If it is Alice's turn to speak (resp. Bob's turn), the protocol specifies what the next bit sent will be as a function of x and the communication exchanged so far (resp. y and the communication exchanged so far). When a run of $\mathcal{P}$ terminates, its output, denoted $\mathcal{P}(x, y)$, is a function of the blackboard's content. We define the *cost* of $\mathcal{P}$ as the maximum number of bits exchanged for any input. Note that we assume that Alice and Bob each have arbitrary computational power.

Formally, a protocol $\mathcal{P}$ with domain $S^{n_A} \times S^{n_B}$ and range T is a finite binary tree where each internal node v is labeled either by a function $A_v : S^{n_A} \to \{0,1\}$ or a function $B_v : S^{n_B} \to \{0,1\}$, and where each leaf is labeled by a value in T. To determine the output $\mathcal{P}(x,y) \in T$ of the protocol on input (x,y) we start walking along the tree from the root. When we visit an internal node v labeled with a function A_v (resp. B_v), we go to v's left child if $A_v(x) = 0$ (resp. $B_v(y) = 0$) and right if $A_v(x) = 1$ (resp. $B_v(y) = 1$). The value $\mathcal{P}(x,y)$ is the label of the leaf thus reached. The *cost* of $\mathcal{P}$ is the height of the tree and we say that $\mathcal{P}$ computes f if $\mathcal{P}(x,y) = f(x,y)$ for all $(x,y) \in S^{n_A} \times S^{n_B}$.

The *deterministic communication complexity* of f, denoted $D(f)$ is the cost of the cheapest protocol computing f. In general, we will be interested in the complexity of functions $f : S^* \times S^* \to T$ and will thus consider $D(f)$ as a function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ (or from $\mathbb{N}$ to $\mathbb{N}$ when the length of inputs given to Alice and Bob are related) and study its asymptotic behavior.

In a *simultaneous protocol* $\mathcal{P}$, we disallow any interaction between Alice and Bob: Each of them *simultaneously* sends a message to a trusted referee which has access to none of the input and the referee produces the output $\mathcal{P}(x,y) \in T$. We denote $D^{\|}(f)$ the simultaneous communication complexity of f, i.e. the cost of the cheapest simultaneous protocol computing f.

In a *probabilistic communication protocol* $\mathcal{P}$, Alice and Bob have access to private random bits which determine their behavior. The protocol is said to compute f if for all x, y, the probability over the choices of these random bits that $\mathcal{P}(x,y) = f(x,y)$ is at least $3/4$. We denote $R(f)$ the probabilistic (or randomized) communication complexity of f.

Combining properties of the two previous models, a *simultaneous probabilistic communication protocol* $\mathcal{P}$, is one in which Alice and Bob *simultaneously* send a message depending on their inputs and their random bits to a trusted referee which then outputs $\mathcal{P}(x,y)$ which should equal $f(x,y)$ with probability at least 3/4. We denote $R^{\|}(f)$ the simultaneous probabilistic (or randomized) communication complexity of f.

In contrast to these first four models, the non-deterministic and MOD_p communication models that we present next will only be used to recognize languages, i.e. to compute functions from $S^* \times S^*$ into $\{0,1\}$.

In a *non-deterministic communication protocol*[1] $\mathcal{P}$ another player, say God, having access to *both* x and y first sends to Alice and Bob a proof π whose length is a function of the length of x and y. Alice and Bob then follow an ordinary deterministic protocol $\mathcal{P}'$ with output in $\{0,1\}$. The protocol $\mathcal{P}$ accepts the input (x, y) if and only if there is some proof π such that the output of the ensuing deterministic protocol $\mathcal{P}'$ outputs 1. The cost of a non-deterministic protocol is the maximum number of bits exchanged in the protocol (*including* the bits of π) for any input (x, y). We denote the non-deterministic communication complexity of a language L as $N^1(L)$. The co-non-deterministic communication complexity of L, denoted $N^0(L)$ is the non-deterministic communication complexity of L's complement.

A *MOD_p-counting communication protocol* $\mathcal{P}$ is similar to a non-deterministic protocol but it accepts those (x, y) such that the number of proofs that lead Alice and Bob to acceptance is *not* divisible by p. We denote by $N^{Mod_p}(L)$ the MOD_p-counting communication complexity of L.

Notice that for any function f, we have $R(f) \leq D(f) \leq max\{n_A, n_B\} + 1$ because every deterministic protocol is a probabilistic protocol and because f can always be computed by a protocol in which one player sends over all its data, subsequently letting the other player compute and then communicate the result. Moreover one can establish $R(f) \geq \log(D(f))$ using brute force derandomization of probabilistic protocols. Similarly the following elementary

[1] We use here a "guess and verify" presentation of non-deterministic protocols which is most convenient in the context of our discussion. Alternatively, we could introduce them as protocols in which Alice and Bob are allowed to act non-deterministically (see e.g. [BFS86, KN97, DKMW92] for alternative presentations).

facts can be easily established:

- $\log(D^{\|}(f)) \leq D(f) \leq D^{\|}(f) \leq n_A + n_B$;
- $\log(R^{\|}(f)) \leq R(f) \leq R^{\|}(f) \leq D^{\|}(f)$;
- $\log(D(L)) \leq N^1(L) \leq D(L)$;
- $\log(D(L)) \leq N^{Mod_p}(L) \leq D(L)$;

Moreover, if p is prime, then $N^{Mod_p}(L) = \Theta(N^{Mod_{p^\alpha}}(L))$ for all integers α and for any languages L_1, L_2:

$$N^{Mod_p}(L_1 \cup L_2) = O(N^{Mod_p}(L_1) + N^{Mod_p}(L_2))$$

and

$$N^{Mod_p}(L_1) = \Theta(N^{Mod_p}(\overline{L_1})).$$

All these models have been extensively studied. At the heart of what we understand about their combinatorics is the following simple observation: if the communication induced by a deterministic protocol $\mathcal{P}$ is the same on input pairs (x_1, y_1) and (x_2, y_2) then it will also be the same for (x_1, y_2) and (x_2, y_1). As an example, consider the function Equality: $EQ(x, y) = 1$ if and only $x = y$. A protocol computing EQ must induce different communication patterns for the 2^n pairs of the form (x, x) with $x \in \{0,1\}^n$ for otherwise the protocol will also accept some pair (x, y) with $x \neq y$. This suffices to show that $D(EQ) \geq n$.

The following four functions are, like Equality, classical examples studied in communication complexity:

- For $x, y \in \{0,1\}^n$, we define Disjointness as: $DISJ(x, y) = 1$ if and only if $\bigvee_{1 \leq i \leq n} x_i y_i = 0$;
- For $x, y \in \{0,1\}^n$, and any $m \in \mathbb{N}$ we define Inner Product $\pmod q$ as: $IP_q(x, y) = 1$ if and only if $\sum_{1 \leq i \leq n} x_i y_i \equiv 0 \pmod q$;

	D	R	$D^{\|\|}$	$R^{\|\|}$	N^{Mod_p}	N^1	N^0
$DISJ$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$
IP_q	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
IP_{p^α}	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n)$
GT	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
$INDEX$	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$

Table 4.1: Some well-known communication complexity bounds. (Note that p is prime, $\alpha \geq 1$ and q is not a power of p.)

- For two n-bit numbers $x, y \in [2^n]$ we define Greater Than as: $GT(x, y) = 1$ if and only if $x \geq y$.
- For $x \in \{0,1\}^n$ and a $\log n$-bit number $p \in [n]$ we define $INDEX(x, p) = x_p$;

The known communication complexity bounds for these problems can be summed up in Table 4.1. It should be noted that non-trivial work is needed to establish some of these bounds. The probabilistic lower bound for $DISJ$ received a lot of attention in the late 80's ([BFS86, KS92, Raz92]) while the probabilistic lower bound for IP_2 follows from quite technical results of [CG85] (see also [DKMW92, Gro94b] for the case $p \neq 2$). The GT probabilistic upper bound, due to N. Nisan and S. Safra, is also tricky (see exercise 3.18 in [KN97]) while the randomized simultaneous lower bounds for $INDEX$ and GT are due respectively to [KNR99] and [MNSW98]. Most MOD_p-counting bounds are theorems (or easy corollaries) of [DKMW92].

4.2.2 Communication Complexity of Regular Languages and Monoids

In general, we want to study the communication complexity of functions which do not explicitly have two inputs. In the case of regular languages and monoids we will use a form of *worst-case partition* definition. Formally, we define the deterministic (resp. randomized, simultaneous, probabilistic simultaneous, MOD$_p$-counting) *communication complexity of a regular language* $L \subseteq A^*$ as the deterministic (resp. randomized, simultaneous, probabilistic simultaneous, MOD$_p$-counting) communication complexity of the following problem: Alice and Bob respectively receive $a_1, a_3, \ldots a_{2n-1}$ and $a_2, a_4, \ldots, a_{2n}$ where each a_i is either an element of A or the empty word[2] ϵ and they want to determine whether $a_1 a_2 \ldots a_{2n}$ belongs to L.

Similarly, the deterministic (resp. randomized, simultaneous, probabilistic simultaneous) *communication complexity of a finite monoid* M is the deterministic (resp . randomized, simultaneous, probabilistic simultaneous) communication complexity of evaluating in M the product $m_1 \cdot m_2 \cdot \ldots \cdot m_{2n}$ where the odd-indexed $m_i \in M$ are known to Alice and the even-indexed m_i are known to Bob. We further define the MOD$_p$-counting communication complexity of M as the maximum over all $F \subseteq M$ of the MOD$_p$-counting complexity of determining if this product $m_1 \cdot m_2 \cdot \ldots \cdot m_{2n}$ belongs to F.

The following basic facts from [RTT98], whose proofs we sketch here for completeness, support our choices of definition:

Lemma 4.1 *Let* $L \subseteq A^*$ *be regular with* $M(L) = M$. *We have* $D(M) = \Theta(D(L))$ *and similarly for* $D^{||}$, R, $R^{||}$ *and* N^{Mod_p} *for* p *prime.*

Proof. [sketch] Let $\phi : A^* \to M$ be the recognizing morphism with $L = \phi^{-1}(F)$. Then a word $a_1 a_2 \ldots a_{2n}$ belongs to L if and only if the product $\phi(a_1)\phi(a_2)\ldots\phi(a_{2n})$ belongs to F and so the communication complexity of L in all four models is bounded by the complexity of $M(L)$.

[2]This definition of communication complexity of a regular language L is, up to a constant factor, equivalent to the worst-case partition complexity discussed in Section 4.5 as long as there exists an integer t such that each $m \in M(L)$ is the image, under the recognizing morphism, of a word of length t. In particular, the communication complexity of L is, up to a constant, the worst-case partition complexity of L^ϵ. These issues are discussed in greater detail in [Tes99].

Conversely if $M = M(L)$ we can pick for each $m \in M$ some word $u_m \in A^*$ where $\phi(u_m) = m$. By padding with empty words ϵ, we can assume that such u_m's can be chosen in $(A \cup \{\epsilon\})^*$ with $|u_m| = t$ for all m. Moreover, by definition of the syntactic congruence, we can find a finite set of pairs of words (x_i, y_i) such that for any $w \in A^*$ and any u_{m_j} holds $w \sim_L u_{m_j}$ (i.e. $\phi(w) = m_j$) if and only if we have $x_i w y_i \in L$ when and only when $x_i u_{m_j} y_i \in L$.

Suppose Alice and Bob are given monoid elements $m_1, m_3, \ldots, m_{2n-1}$ and $m_2, m_4, \ldots, m_{2n}$ respectively. If they have a protocol for L and want to evaluate the product $m_1 m_2 \ldots m_{2n}$ they can do so by repeatedly using the L-protocol to check if each of the words $s_i = x_i(u_{m_1} u_{m_2} \ldots u_{m_{2n}})y_i$ belong to L. In order to use the L-protocol, it must be the case that Alice knows all odd-indexed letters in s_i and Bob knows every even-indexed one so padding with ϵ has to be used once more to achieve this. Still, the length of the resulting s_i's will be no more than $4qn$. □

In particular the deterministic (resp. simultaneous, randomized, probabilistic simultaneous, MOD$_p$-counting) complexity of a monoid M is, up to a constant, the maximal communication complexity of any regular language that it can recognize.

Lemma 4.2 *For any increasing $f : \mathbb{N} \to \mathbb{N}$ the class of monoids such that $D(M)$ (resp. $D^{||}(M)$, $R(M)$, $R^{||}(M)$, $N^{Mod_p}(M)$ for p prime) is $O(f)$ forms a variety.*

Proof. [sketch] It is straightforward to verify that in all four models, the communication complexity of the direct product of monoids $M \times N$ is bounded by the sum of the complexities of M and N. Moreover, if $N \prec M$ then every language recognized by N is also recognized by M and by our previous remark the complexity of N is at most that of M. □

4.2.3 Rectangular Reductions

As we mentioned earlier, [BFS86] introduced a convenient notion of reductions in the communication complexity setting.

Definition 4.3 *A* rectangular reduction *of length t from a language $L \subseteq A^* \times A^*$ to a language $L' \subseteq A'^* \times A'^*$ is a pair of functions (r_A, r_B) such that for any $(x, y) \in A^* \times A^*$:*

1. *$|r_A(x)|$ and $|r_B(y)|$ depend respectively on $|x|$ and $|y|$ and are respectively bounded by $t(|x|)$ and $t(|y|)$;*

2. *$(x, y) \in L$ if and only if $(r_A(x), r_B(y)) \in L'$.*

Clearly, a rectangular reduction from L to L' can be used to infer a communication complexity lower bound for L' from a lower bound for L since $r_A(x)$ and $r_B(y)$ can be computed privately by Alice and Bob respectively.

We give here a variant of this definition which specifically suits our needs:

Definition 4.4 *Let $L \subseteq A^n \times A^{f(n)}$ and M be some finite monoid. We define a* rectangular reduction *of length t from L to M as a sequence of $2t$ functions $a_1, b_2, a_3, \ldots, a_{2t-1}, b_{2t}$, with $a_i : A^n \to M$ and $b_i : A^{f(n)} \to M$, such that for every $x \in A^n$ and $y \in A^{f(n)}$ we have $(x, y) \in L$ if and only if $eval_M(a_1(x)b_2(y)\ldots b_{2t}(y)) \in T$ for some target subset T of M.*

Such a reduction transforms a pair (x, y) into a sequence of $2t$ monoid elements $m_1, \ldots, m_{2t}$ where the odd-indexed m_i are obtained as a function of x only and the even-indexed m_i are a function of y.

In general, we are interested in reductions from $K \subseteq A^* \times A^*$ into M. In our definition we used the notation $L \subseteq A^n \times A^{f(n)}$ to stress that we focus on languages K in which pairs (x, y) have lengths related by a common parameter n. It should be clear that if K has communication complexity $\Omega(g(n))$ and has, for each n, a reduction of length $t(n)$ to M then M has complexity $\Omega(g(t^{-1}(n)))$. We will write $K \leq_r^t M$ to indicate that K has a rectangular reduction of length t to M and will drop the t superscript whenever $t = O(n)$.

Note that if the language K is recognized by a program of length $t(n)$ over M then we have $K \leq_r^t M$ since a program is a special form of rectangular reduction in which every a_i (resp. b_i) depends in fact on a single letter of x (resp. y).

4.2.4 Bounds and Classifications

We establish bounds on the two-party communication complexity of monoids and regular languages and provide complete classifications in the deterministic, probabilistic, simultaneous and MOD$_p$-counting (p prime) models. The analysis of the first three cases was published as [TT03]. We begin with an easy observation.

Lemma 4.5 *If M is commutative then $D^{||}(M) = O(1)$.*

Proof. Since M is commutative, we have

$$m_1 \cdot m_2 \cdot \ldots \cdot m_{2n} = (m_1 \cdot m_3 \cdot \ldots \cdot m_{2n-1}) \cdot (m_2 \cdot m_4 \cdot \ldots \cdot m_{2n}).$$

So if Alice and Bob send to the referee the $\log |M|$ bits representing $(m_1 \cdot m_3 \cdot \ldots \cdot m_{2n-1})$ and $(m_2 \cdot m_4 \cdot \ldots \cdot m_{2n})$ respectively, he can compute the product $m_1 \cdot m_2 \cdot \ldots \cdot m_{2n}$. □

Next, we use the combinatorial description of languages with syntactic monoids in **DO** to obtain another upper bound.

Lemma 4.6 *Let $L \subseteq A^*$ be such that $M(L) \in$ **DO** $\cap$ $\overline{\mathbf{Ab}}$. Then $D(L) = O(\log n)$.*

Proof. By Lemma 2.20, L is a union of $\sim^G_{|A|,k}$-classes for some Abelian group G. We claim that any such class has logarithmic communication complexity and argue by induction on $t = |A| + k$. For $t = 1$ there is nothing to prove. For $t > 1$, let $u \in (A \cup \{\epsilon\})^*$ be some predetermined representative of the class. Given the input $x = x_1 x_2 \ldots x_{2n} \in A^*$, Alice and Bob can check whether $\alpha(x) = \alpha(u)$ by exchanging $|A| + 1$ bits. Next, they need to verify that u and x are G-equivalent. If G has exponent p, we get u and x G-equivalent if and only if $|u|_a \equiv |x|_a \pmod p$ for all $a \in A$. The latter condition can easily be verified with communication cost about $|A|\lceil \log p \rceil$, a constant. Let $u = vaw$ be the a-left decomposition of u and i, j denote the locations of the leftmost occurrence of a in x that is seen respectively by Alice and Bob. These indices can be exchanged at logarithmic communication cost so that if, for example, i is smaller than j then Alice and Bob can conclude that $x = x_1 \ldots x_{i-1} a x_{i+1} \ldots x_n$ is the a-left decomposition of x and further verify, by induction, that $x_1 \ldots x_{i-1} \sim^G_{|A|-1,k} v$ and $x_{i+1} \ldots x_n \sim^G_{|A|,k-1} w$ using only $O(\log n)$ communication. Left-right symmetry completes the proof. □

The example of GT shows that probabilistic protocols can be much more efficient than deterministic ones and it is natural to ask whether such gains can be made for certain monoids in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$. This motivates the following definition:

Definition 4.7 *We call* $\mathbf{W}$ *the variety of monoids* M *satisfying:*

1. $M \in \mathbf{DO}$*;*

2. $exwyf = ewxyf = exywf$ *for all* $w, e, f, x \in M$ *such that* e, f *are idempotents lying* $\mathcal{J}$*-below* w*; i.e.* M *satisfies*

$$\begin{aligned} (swt)^\omega wxy (uwv)^\omega &= (swt)^\omega xwy (uwv)^\omega \\ &= (swt)^\omega xyw (uwv)^\omega. \end{aligned}$$

Remark 4.7. Suppose $M \in \mathbf{W}$ has exponent p and consider some $w \in M$ lying $\mathcal{J}$-above idempotents e, f, and some $x, y \in M$. Since $M \in \mathbf{DO}$, we have $ew^pe = e$ and so

$$\begin{aligned} exw^pyf &= ew^pexw^pyf \\ &= ew^pw^pexyf \text{ (By condition 2)} \\ &= exyf \end{aligned}$$

Note also that condition 2) shows that $\mathbf{W} \subseteq \overline{\mathbf{Ab}}$. Indeed, if u, v are elements of a subgroup with identity element e, we have

$$uv = euve = evue = vu.$$

For a word $u \in A^*$ and $a \in A$ we denote by $RED_t(u)$ be the unique word of A^* obtained by keeping in u only the first and last t occurrences of each letter a with $|u|_a \geq 2t$ and all occurrences of letters a with $|u|_a < 2t$. For example, $RED_2(abcbabbababba) = abcbaabba$. We will show that languages recognized by monoids in $\mathbf{W}$ have a useful combinatorial characterization: we set $u \approx_{t,p} v$ if and only if:

1. $RED_t(u) = RED_t(v)$;
2. For all $a \in A$ we have $|u|_a \equiv |v|_a \pmod p$.

Alternatively, we could define $RED_{t,p}(u)$ as the word obtained from u by the following process: For every $a \in \alpha(u)$ with $|q|_a \geq 2t$ mark the first and last t occurrences of a then move all *other* occurrences of a, if any, next to the t^{th} one and then reduce that block of a's modulo p. If $|q|_a < 2t$, all occurrences of a are left untouched. Note that we clearly have $RED_{t,1}(u) = RED_t(u)$ and $u \approx_{t,p} v$ if and only if $RED_{t,p}(u) = RED_{t,p}(v)$.

Theorem 4.8 *Let $M = A^*/\gamma$, then $M \in \mathbf{W}$ if and only if $\approx_{t,p} \subseteq \gamma$ for some t, p.*

Proof. For one direction, we need to show that $M = A^*/\approx_{t,p}$ lies in $\mathbf{W}$. By definition, we can see that the $\approx_{t,p}$ equivalence classes are unambiguous concatenations of languages with syntactic monoids in $\mathbf{J_1} \vee \mathbf{Ab}$ and so $M \in \mathbf{DO} \cap \overline{\mathbf{Ab}}$ by Theorem 2.20. Furthermore, let us consider the words $q = (uwv)^{tp}wx(ywz)^{tp}$ and $r = (uwv)^{tp}xw(ywz)^{tp}$. For any $a \in A$, $|q|_a \equiv |x|_a + |w|_a \equiv |r|_a \pmod p$ and

$$\begin{aligned} RED_t((uwv)^{tp}wx(ywz)^{tp}) &= RED_t((uwv)^{tp}x(ywz)^{tp}) \\ &= RED_t((uwv)^{tp}xw(ywz)^{tp}) \end{aligned}$$

since for any letter a occurring in w, the first t occurrences of a lie in $(uwv)^{tp}$ and its last t occurrences lie in $(ywz)^{tp}$. Thus, $q \approx_{t,p} r$ so M satisfies condition 2 of Definition 4.7.

Conversely, suppose M is in $\mathbf{W}$. We need to show that there exist t, p such that for any morphism $\phi : A^* \to M$ we have $\phi(q) = \phi(r)$ for any $q \approx_{t,p} r$ and it is in fact sufficient to establish $\phi(q) = \phi(RED_{t,p}(q))$ since $RED_{t,p}(q) = RED_{t,p}(r)$. In particular, we choose p as the exponent of M and t as $|M| + 1$.

Recall that to obtain $RED_{t,p}(q)$, one successively considers all $a \in A$ with $|q|_a \geq 2t$, "groups" together the "middle" a's and reduces their number modulo p. We will show that the image under ϕ is preserved by this process. Consider a word $u = u_0au_1a \dots au_t$ with at least t occurrences of a. Since $t = |M| + 1$, there must exist $1 \leq i < j \leq t$ such that

$$s_i = \phi(u_0au_1a \dots u_i) = \phi(u_0au_1a \dots u_j) = s_j.$$

This means that

$$s_i = s_i\phi(au_{i+1}a \dots au_j) = s_i\phi(au_{i+1}a \dots au_j)^\omega.$$

Therefore there exist $g, h \in M$ such that $\phi(u)$ can be written as geh where $e = \phi(au_{i+1}a \dots au_j)^\omega$ is an idempotent lying $\mathcal{J}$-below a.

Suppose now that q contains at least $2t+1$ occurrences of a. We can thus factor q as $q = u_0 a \dots u_{t-1} a x a y a v_{t-1} a \dots a v_0$ where the u_i's and v_i's do not contain a. From the remarks of the preceding paragraph, we can now use condition 2 in Definition 4.7 to obtain $\phi(q) = \phi(u_0 a \dots u_{t-1} a\, a\, xy\, a v_{t-1} a \dots a v_0)$. Repeating this same process for all occurrences of a in x or y we can get $\phi(q) = \phi(u_0 a \dots u_{t-1} a\, a^{kp+d} z\, a v_{t-1} a \dots a v_0)$ where $a \notin \alpha(z)$ and from condition 1: $\phi(q) = \phi(u_0 a \dots u_{t-1} a\, a^d z\, a v_{t-1} a \dots a v_0)$ where $0 \leq d < p$ is such that $|q|_a - 2t \equiv d \pmod{p}$. If the same manipulation is made for every $a \in A$, we obtain $\phi(q) = \phi(RED_{t,p}(q))$ as we needed. $\square$

At least intuitively, we have $M(L)$ lying in $\mathbf{W}$ if and only if membership of a word w in L can be determined by counting threshold t, mod p the occurrences of letters in w and determining the relative positions of any of the first and last t occurrences of letters in w. In terms of two-party communication complexity, $\mathbf{W}$ thus forms an "easy" subclass of $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ because these comparisons of $\log n$-bit numbers can be done relatively efficiently.

Lemma 4.9 *Let $L \subseteq A^*$ be such that $M(L) \in \mathbf{W}$. Then $D^{||}(L) = O(\log n)$ and $R(L) = O(\log\log n)$.*

Proof. As in the previous proof, we obtain these upper bounds for the $\approx_{t,p}$ classes. Let u be some representative of the target class and x the common input of Alice and Bob. Checking whether $|u|_a \equiv |x|_a \pmod{p}$ is easily done at constant cost so we only need to show that verifying $RED_t(x) = RED_t(u)$ can be done efficiently. For the simultaneous case, the players send to the referee the locations of the first and last t occurrences that they see of each letter $a \in A$. Given this information, the referee can reconstruct $RED_t(x)$ and compare it to $RED_t(u)$.

For the probabilistic case we use a subprotocol of cost $O(\log\log n)$ to determine for any $k \leq t$ which of Alice or Bob holds the k^{th} (or symmetrically the

k^{th} to last) occurrence of some letter a in x, provided of course that $|x|_a \geq k$. We argue by induction on k: For $k = 1$, let i, j be the positions of the first occurrence of a seen by Alice and Bob respectively. Of course Alice holds the first occurrence of a if and only if $i < j$ and, using the complexity bound mentioned in Table 4.1, this can be tested by a randomized protocol at cost $O(\log \log n)$ since i, j are only $\log n$-bits long. For $k > 1$ we can assume from induction that Alice and Bob have marked, in their respective inputs, the occurrences of a which are among the first $k - 1$ of a in x. The k^{th} occurrence must be either the first unmarked a that Alice sees or the first unmarked a that Bob sees, whichever comes first in x. Once again, Alice and Bob are left with comparing two $\log n$-bit numbers and apply the $O(\log \log n)$ cost protocol.

For $i, j \leq t$, the i^{th} occurrence of a in x comes before the j^{th} occurrence of b in x if and only if the i^{th} occurrence of a in $RED_t(x)$ comes before the j^{th} occurrence of b in $RED_t(x)$. This means that Alice and Bob can check $RED_t(x) = RED_t(u)$ by verifying that for all $i, j \leq t$ and all $a, b \in A$ the i^{th} occurrence of a precedes the j^{th} occurrence of b in $RED_t(u)$ if and only if the i^{th} occurrence of a precedes the j^{th} occurrence of b in x. Since they can determine which of them holds these occurrences, they can check precedence either privately (when one player holds both occurrences) or by using once more the $O(\log \log n)$ randomized protocol to compare two $\log n$ bit numbers.

It should be noted that in any event, the GT protocol is used only a constant number of times (depending on t and $|A|$) so we need not worry about the dwindling of the overall probability of correctness in the protocol. $\square$

We have seen that unambiguous products of languages with commutative syntactic monoids have $O(\log n)$ deterministic communication complexity. It should not come as much of surprise that in the MOD$_p$-counting model we can correspondingly obtain:

Lemma 4.10 *Let $L \subseteq A^*$ be regular with $M(L) \in$* $\mathbf{LG_p} ⓜ \mathbf{Com}$. *We have $N^{Mod_p}(L) = O(\log n)$.*

Proof. We know from the result of [Wei92] cited as Lemma 2.23 that L is in $M_p Pol(\mathcal{L}_{com})$, i.e. is a Boolean combination of languages of the form

$$\{x | \binom{x}{(L_0 a_1 L_1 \dots a_k L_k)} \equiv j \pmod p\}$$

where $M(L_i)$ is commutative for all i. So we only need to exhibit an $O(\log n)$-cost protocol to check if a given word $w \in (A \cup \{\epsilon\})^*$ has a number of factorizations as $u_0 a_1 u_1 \dots a_k u_k$, with $u_i \in L_i$, that is congruent to j modulo p.

Suppose first that $j = 0$. The protocol we present in the next paragraph will in fact output positively if and only if the number of valid factorizations is *not* congruent to 0 modulo p. This is sufficient as we have mentioned that for p prime $N^{Mod_p}(L) = N^{Mod_p}(\overline{L})$.

The proof sent by God in the first step of the protocol consists of k $\log n$-bit integers $t_1 < t_2 < \dots < t_k$. In the next stage of the protocol, Alice and Bob interpret the t_i's as possible locations for the bookmarks $a_1, a_2, \dots, a_k$ in w and accept if they correspond to a valid factorization $u_0 a_1 u_1 \dots a_k u_k$ with $u_i \in L_i$. This can be done at constant cost since Alice and Bob need only check that position t_i indeed contains letter a_i and that segment u_i belongs to L_i, which requires only $O(1)$ bits since $M(L_i)$ is commutative. The cost of the protocol is dominated by the length of the proof which is $O(\log n)$. The number of proofs accepted by Alice and Bob is thus exactly the number of legal factorizations so the protocol accepts if and only if it is non-zero modulo p. Note that for $j \neq 0$, we need to slightly modify our protocol by adding the possibility for God to send one of $(p - j)$ different "special" proofs that always lead Alice and Bob to accept. □

To obtain lower bounds matching the upper bounds presented above, we give a number of conditions under which a finite M admits a reduction from

Greater Than, Disjointness, Inner Product mod q and Index.

Lemma 4.11 *1. If M is non-commutative then $GT \leq_r^{2^n} M$ (Notice that the reduction has exponential length);*

2. If M is not in **DS** *then $DISJ \leq_r M$;*

3. If M lies in **DS** *but is not in* **DO** *then $IP_q \leq_r M$ for some integer q;*

4. For any prime p, if M lies in **DS** *but contains $\mathcal{J}$-related idempotents e, f such that $(ef)^{p^\alpha}$ is not idempotent for any $\alpha \geq 1$ then $IP_q \leq_r M$ for q not a power of p;*

5. If G is a non-commutative group then $IP_q \leq_r G$ for some integer q;

6. For any prime p, if G is a group outside of $\mathbf{G_p} * \mathbf{Ab}$ then $IP_q \leq_r G$ for some q which is not a power of p;

7. If M is in **DO** *but not in* **W** *then $INDEX \leq_r M$.*

Proof. 1- Let $a, b \in M$ be such that $ab \neq ba$. We obtain an exponential length reduction from $GT(x, y)$ by building $m_1 m_2 \dots m_{2^{n+1}}$ where $m_i = a$ for $i = m_{2x}$; $m_i = b$ for $i = m_{2y-1}$ and $m_i = 1_M$ otherwise. The product of the m_i is then ba if and only if $x \geq y$ and is ab otherwise.
2- If M is not in **DS** then it admits one of B_2 or U as a divisor. In both cases, the reduction from $DISJ$ builds for every pair x_i, y_i a four-tuple $m_{4i-3} \dots m_{4i}$ where $m_{4i-3} = a$ and $m_{4i-1} = b$ when $x_i = 1$ and $m_{4i-3} = m_{4i-1} = 1_M$ when $x_i = 0$, $m_{4i-2} = ab$ when $y_i = 1$ but $m_{4i-2} = 1_M$ when $y_i = 0$ and $m_{4i} = ab$ for any input. One can check that in both B_2 and U, any such four-tuple evaluates to 0 when $x_i = y_i = 1$ and to ab otherwise so the product of all of them is 0 if $x_i = y_i = 1$ for some i and is ab otherwise.
3- Since M is in **DS** but not **DO**, there must exist two $\mathcal{J}$-related idempotents e, f such that ef is not idempotent. Since M is in **DS**, however, we have $efe \neq$

$e = (ef)^\omega e$. Let q be minimal such that $(ef)^q e = e$: The reduction produces elements $m_1 \dots m_{2n}$ where $m_{2i-1} = e$ if $x_i = 1$ and $m_{2i-1} = e(ef)^\omega = (ef)^\omega$ otherwise and $m_{2i} = fe$ if $y_i = 1$ and $m_{2i} = (ef)^\omega e = e$ otherwise. In particular the product $m_{2i-1}m_{2i}$ is efe if and only if $x_i = y_i = 1$ and is e otherwise and so the product $m_1 \dots m_{2n}$ equals $(ef)^{\sum_{1\leq i\leq n} x_i y_i \equiv 0 \pmod q} e$ which equals e if and only if $IP_q(x, y) = 1$.

4- The argument is almost the same as 3-. Again, let q be minimal such that $(ef)^q e = e$. We are guaranteed that q is not a power of p and we can reuse the reduction described for 3-.

5- If G is not Abelian, there must exist $g, h \in G$ such that the commutator $[g, h] = g^{-1}h^{-1}gh$ is not the identity and thus has order $q \neq 1$. We obtain a reduction from IP_q by creating for each pair x_i, y_i a four-tuple of monoid elements $m_{4i-3}m_{4i-2}m_{4i-1}m_{4i}$ where $m_{4i-3} = g^{-1}$ and $m_{4i-1} = g$ when $x_i = 1$ and $m_{4i-3} = m_{4i-1} = 1_G$ when $x_i = 0$ and where $m_{4i-2} = h^{-1}$ and $m_{4i} = h$ when $y_i = 1$ and $m_{4i-2} = m_{4i} = 1_G$ when $y_i = 0$. This four-tuple thus evaluates to $[g, h]$ if and only if $x_i = y_i = 1$ and to 1_G otherwise and the product of all such tuples is $[g, h]^{\sum_{1\leq i\leq n} x_i y_i \pmod q}$.

6- If G is not in $\mathbf{G_p} * \mathbf{Ab}$, there must exist $g, h \in G$ such that $[g, h]$ has order q which is not a power of p so we can reuse the previous reduction.

7- If M lies in $\mathbf{DO} - \mathbf{W}$, we must consider two cases. Assume first that there exist e, f, u, v, w with e, f idempotent and $\mathcal{J}$-below w such that $euw^\omega vf \neq euvf$. Since M is in $\mathbf{DO}$ we have $ew^\omega e = e$ and $fw^\omega f = f$.

We obtain a rectangular reduction from $INDEX(x, s)$ by creating $m = m_1 m_2 \dots m_{2n+1}$ as follows:

$$m_i = \begin{cases} e & \text{for } i = 1, 3, \dots, 2s-3 \text{ (the first } s-1 \text{ odd-indexed } m_i\text{'s);} \\ (eu) & \text{for } i = 2s-1; \\ (vf) & \text{for } i = 2s+1; \\ f & \text{for } i = 2s+3, \dots, 2n+1 \text{ (all other odd indexed } m_i\text{'s);} \\ 1_M & \text{for } i = 2j \text{ and } x_j = 0; \\ w^\omega & \text{for } i = 2j \text{ and } x_j = 1. \end{cases}$$

The values of the odd indexed and even indexed m_i depend respectively on s and x as required and one can see that since $ew^\omega e$ is e and $fw^\omega f$ is f, the product of the m_i's is equal to $euvf$ when x_s is 0 and $euw^\omega vf$ when x_s is 1 which shows the correctness of the reduction.

If on the other hand we do have $euw^\omega vf = euvf$ for all suitable e, u, v, w, f, then there must exist $e, f, u, w \in M$ with e, f idempotent $\mathcal{J}$-below w but $ewuf \neq euwf$ for otherwise we have $euwvf = euw^\omega wvf = euw^\omega vwf = euvwf$ and M lies in $\mathbf{W}$. On the other hand, since we assume $euw^\omega vf = euvf$, we have $euwf = ew^\omega uwf$ and $ewuf = ewuw^\omega f$. We now obtain a rectangular reduction from $INDEX(x, s)$ (assuming w.l.o.g. that $x_n = 1$) as follows:

$$m_i = \begin{cases} e & \text{for } i = 1 \\ f & \text{for } i = 2n+1 \\ u & \text{for } i = 2s+1; \\ 1_M & \text{for all other odd-indexed } i; \\ w^\omega & \text{for even } i = 2j \text{ such that } x_j = x_{j-1}; \\ w & \text{for even } i = 2j \text{ such that } x_j = 1 \text{ and } x_{j-1} = 0; \\ w^{\omega-1} & \text{for even } i = 2j \text{ such that } x_j = 0 \text{ and } x_{j-1} = 1. \end{cases}$$

Again, the values of the odd indexed and even indexed m_i's depend respectively on s and x. The value of the even indexed m_i's are such that the product $m_2 m_4 \dots m_{2i}$ is $w^{k\omega+1}$, for some k, if and only if x_i is 1 and $w^{k\omega}$ if x_i is 0. Similarly, using the fact that $x_n = 1$, the product $m_{2i+2} \dots m_{2n}$ is $w^{k'\omega}$ if and only if $x_i = 1$ and $w^{k'\omega+1}$ otherwise. Using the values assigned by the reduction to the odd-indexed m_i's we have

$$m_1 m_2 \dots m_{2n+1} = e m_2 m_4 \dots m_{2s}\, u\, m_{2s+2} m_{2s+4} \dots m_{2n} f$$

and by our previous remarks this is $ew^{k\omega+1} u w^{k'\omega} f = ewuf$ if $x_s = 1$ and $ew^{k\omega} u w^{k'\omega+1} f = euwf$ if $x_s = 0$ so our reduction is correct. □

In particular, if M does not lie in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ then it either admits a linear length reduction from $DISJ$ (if it is outside $\mathbf{DS}$) or from IP_q for some q (if it is either in $\mathbf{DS}$ but not in $\mathbf{DO}$ or if it is outside $\overline{\mathbf{Ab}}$). Combining our last lemma with the upper bounds above we can obtain the three following theorems.

Theorem 4.12 *Let $L \subseteq A^*$ be a regular language with $M = M(L)$. Then*

$$D(L) = \begin{cases} O(1) & \text{if and only if } M \text{ is commutative;} \\ \Theta(\log n) & \text{if and only if } M \text{ is in } \mathbf{DO} \cap \overline{\mathbf{Ab}} \text{ but not commutative;} \\ \Theta(n) & \text{otherwise.} \end{cases}$$

Proof. We know $D(L) = O(1)$ if M is commutative and $D(L) = \Omega(\log n)$ otherwise since in that case $GT \leq_r^{2^n} M$. Lemma 4.6 gives the upper bound when $M \in \mathbf{DO} \cap \overline{\mathbf{Ab}}$. Finally, when M is not in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$, then it admits a linear length reduction from $DISJ$ or IP_q which yields the last $\Omega(n)$ lower bound. □

Theorem 4.13 *Let $L \subseteq A^*$ be a regular language with $M = M(L)$. Then*

$$R(L) = \begin{cases} O(1) & \text{if and only if } M \text{ is commutative;} \\ \Theta(\log\log n) & \text{if and only if } M \text{ is in } \mathbf{W} \text{ but not commutative;} \\ \Theta(\log n) & \text{if and only if } M \text{ is in } \mathbf{DO} \cap \overline{\mathbf{Ab}} \text{ but not in } \mathbf{W}; \\ \Theta(n) & \text{otherwise.} \end{cases}$$

Proof. When M is in $\mathbf{W}$ but not commutative we put together Lemma 4.9 and part 1 of Lemma 4.11 to get the tight $\log\log n$ bound. Similarly, if M lies in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ but not in $\mathbf{W}$ then it admits a reduction from $INDEX$ which proves the $\Omega(\log n)$ lower bound matching Lemma 4.6 and when M is not in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ we again use the linear lower bounds on the probabilistic complexity of $DISJ$ and IP_q. □

Theorem 4.14 *Let $L \subseteq A^*$ be a regular language with $M = M(L)$. Then*

$$D^{||}(L)) = \Theta(R^{||}(L) = \begin{cases} O(1) & \text{if and only if } M \text{ is commutative;} \\ \Theta(\log n) & \text{if and only if } M \text{ is in } \mathbf{W} \\ & \text{but } M \text{ is not commutative;} \\ \Theta(n) & \text{otherwise.} \end{cases}$$

Proof. When M is in $\mathbf{W}$ but not commutative we combine the upper bound of Lemma 4.9 with the lower bound obtained from part 1 of Lemma 4.11. When M is not in $\mathbf{W}$ then it admits a linear length reduction from $INDEX$, $DISJ$ or IP_q which all have $\Omega(n)$ probabilistic simultaneous complexity. □

There are non-regular languages for which probabilistic simultaneous protocols significantly outperform deterministic simultaneous ones. For instance, the probabilistic simultaneous complexity of Equality is $O(\sqrt{n})$ (folklore, see [NS96, BK97] for explicit protocols) and it was established by [BK97] that this quadratic gain is optimal, i.e. that for any L holds $D^{||}(L) = O(R^{||}(L)^2)$. Such gaps do not exist for regular languages because they do not exist for GT, IP_q, $INDEX$ or $DISJ$.

Theorem 4.15 *Let $L \subseteq A^*$ be a regular language with $M = M(L)$ and p be prime. Then*

$$N^{MOD_p}(L) = \begin{cases} O(1) & \text{if and only if } M \text{ is commutative;} \\ \Theta(\log n) & \text{if and only if } M \text{ is in } \mathbf{LG_p} \circledM \mathbf{Com} \\ \Theta(n) & \text{otherwise.} \end{cases}$$

Proof. If M is in $\mathbf{LG_p} \circledM \mathbf{Com}$ but not commutative, we use $GT \leq_r^{2^n} M$ for the lower bound and Lemma 4.10 for the upper bound. If M is not in $\mathbf{LG_p} \circledM \mathbf{Com}$ then it must be either outside of $\mathbf{DS}$, outside of $\overline{\mathbf{G_p} * \mathbf{Ab}}$ or have $\mathcal{J}$-related idempotents e, f with $(ef)^{p^\omega}$ not idempotent and by Lemma 4.11 we then have either $DISJ \leq_r M$ or $IP_q \leq_r M$ for some q not a prime power of p. In either cases this suffices to get $N^{MOD_p}(M) = \Omega(n)$. □

Corollary 4.16 *If L is a regular language such that for two distinct primes p and q we have both $N^{MOD_p}(L) = O(\log n)$ and $N^{MOD_q}(L) = O(\log n)$. Then $D(L) = O(\log n)$.*

Proof. By our previous theorem, M must lie in the intersection of $\mathbf{LG_p} \circledM \mathbf{Com}$ and $\mathbf{LG_q} \circledM \mathbf{Com}$. In particular, M lies in $\overline{\mathbf{G_p} * \mathbf{Ab}}$ and $\overline{\mathbf{G_q} * \mathbf{Ab}}$ so it must in fact lie in $\overline{\mathbf{Ab}}$. Similarly, if for all $\mathcal{J}$-related idempotents e, f holds both $(ef)^{p^\omega}$ and $(ef)^{q^\omega}$ then it must be that ef is itself idempotent and so M lies in $\mathbf{DO}$. By Theorem 4.12, all monoids in $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ have $O(\log n)$ deterministic communication complexity. □

One might suspect that such a phenomenon does not occur for non-regular languages although, to the best of our knowledge, this question has never been studied. On the other hand it *is* known that for any language L we have

$$D(L) = O(N^1(L) \cdot N^0(L)).$$

4.3 Multiparty Communication Complexity

4.3.1 The Input on the Forehead Model

With applications to distributed computing in mind, it seems natural to generalize the two-party model to a k-party model in which each player gets access to a $1/k$ fraction of the input. Although this model has been studied [DF89] it is of limited interest because its power goes down as the number of players increases.

Chandra, Furst and Lipton, on the other hand, introduced an alternative multiparty model [CFL83] which has since found numerous theoretical applications. In this variant, k players $P_1, \ldots, P_k$ collaborate to compute a function $f(x_1, \ldots, x_k)$ where each participant P_i knows the values of all the inputs *except* x_i. This game is often referred to as the "number on the forehead" model since it is convenient to picture that player i has x_i written on his forehead, available to everyone but himself. The players exchange bits, according to a previously agreed upon protocol, by writing them on a blackboard seen by everyone. The protocol specifies whose turn it is to speak and what a player broadcasts is a function of the communication history and the input he has access to. The protocol's output is a function of what is on the blackboard after the protocol's termination. We will denote[3] by $D_k(f)$ the deterministic k-party communication complexity of f. It should be clear that $D_2(f) = D(f)$.

We can of course define simultaneous, probabilistic, non-deterministic and MOD_p-counting variants of this model. We will use $D_k^{||}$, R_k, N_k^1 and $N_k^{Mod_p}$ to respectively denote simultaneous, probabilistic, non-deterministic and MOD_p-counting k-party communication complexity.

Much less is known about the multiparty models: they sometimes have very surprising power (see e.g. non-trivial upper bounds in [Amb96, Gro94a, Pud03] and bounds presented later in this section) and there seems to be no way to avoid tricky combinatorics when establishing lower bounds. The information known to each player overlaps a lot since any input letter is known to $k-1$ of the k players. This also means that the power of the multiparty model increases with the number of players involved as the fraction of inputs that a player cannot see decreases. Let us consider the 3-way generalization of the equality function: $EQ_3(x, y, z) = 1$ if and only if $x = y = z$. While EQUALITY is the canonical example of a function with maximal two-party deterministic communication complexity, EQ_3 can be computed by a 2-bit 3-party protocol. Indeed, it suffices for the player holding x on his forehead to verify that $y = z$ and for the player holding y to verify $x = z$.

In the combinatorial analysis of two-party models, the central notion was that of a rectangle. The corresponding notion in the multiparty model is that of cylinder intersections. A subset S of $X_1 \times X_2 \times \ldots \times X_k$ is said to be a *cylinder in the ith dimension* if membership in S is independent of the ith coordinate, i.e. if for all $x_1, x_2, \ldots, x_k$ and any x_i' we have $(x_1, \ldots, x_i, \ldots, x_k) \in S$ if and only if

[3] The k-party communication complexity is sometimes denoted by $D^k(f)$, a notation primarily used to denote k-round two-party complexity. We choose to put k as a subscript to avoid this confusion.

$(x_1, \ldots, x'_i, \ldots, x_k) \in S$. We say that S is a *cylinder intersection* if $S = \bigcap_{1 \leq i \leq k} S_i$ where S_i is a cylinder in the ith dimension.

Lemma 4.17 (see [KN97]) *Let $f : X_1 \times X_2 \times \ldots \times X_k \to \{0,1\}$ be a function of k-inputs. Any deterministic k-party communication protocol of cost c computing f partitions the input space into at most 2^c f-monochromatic cylinder intersections corresponding to the communication exchanged on a particular input.*

Two-dimensional cylinder intersections are just rectangles of course, but k-dimensional cylinder intersections have much less structure than k-dimensional hyper-rectangles. We will say that a set of k elements of $X_1 \times X_2 \times \ldots \times X_k$ forms a *star* if it is of the form:

$$(x'_1, x_2, \ldots, x_k), (x_1, x'_2, \ldots, x_k), \ldots, (x_1, x_2, \ldots, x'_k)$$

where for each i, $x_i \neq x'_i$ and $x_i \in X_i$. In that case, we call $(x_1, x_2, \ldots, x_k)$ the *center* of this star. These notions allow us to give a useful characterization of cylinder intersections.

Lemma 4.18 *A set $S \subseteq X_1 \times X_2 \times \ldots \times X_k$ is a cylinder intersection if and only if the center of any star contained in S is itself an element of S.*

Historically, the first multiparty lower bound is that of A. Chandra, M. Furst and R. Lipton who used Ramsey theory to obtain bounds on the complexity of adding k integers. Let EXACTLY_{2^n} (abbreviated E_{2^n}) be the function of k n-bit integers defined as

$$E_{2^n}(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } \Sigma x_i = 2^n; \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 4.19 ([CFL83]) *For all constant k, we have $D_k(E_{2^n}) = \omega(1)$.*

In fact, the bounds for the communication complexity of E_{2^n} are both tight and unclear since they are expressed in terms of Ramsey-theoretical sequences whose limit are known to be infinite but for which no reasonable bounds exist except for the cases $k = 2, 3$.

4.3.2 Discrepancy Bounds

A completely different approach to obtain lower bounds on the multiparty communication complexity of some function f is to bound the maximal size of any f-monochromatic cylinder intersection. In many cases, it is even possible to find strict bounds on the size of cylinder intersections which are "almost" f-monochromatic and this yields lower bounds on the cost of protocols that can approximate f.

This approach is formalized using the notion of *discrepancy*. Let p be prime and ω be some complex p^{th} root of unity. For a function $f : X_1 \times \ldots \times X_k \to \{1, \omega, \ldots, \omega^{p-1}\}$, we define the discrepancy of f as

$$Disc_k(f) = \max_S |\Sigma_{i=0}^{i=p-1} \omega^i Pr[f(x_1, \ldots, x_k) = \omega^i \text{ and } (x_1, \ldots, x_k) \in S]|$$

where the maximum is taken over all cylinder intersections S and where the tuple $(x_1, \ldots, x_k)$ is chosen uniformly at random from $X_1 \times \ldots \times X_k$. Intuitively, a function with low discrepancy can not have large cylinder intersections in which a large fraction of elements have the same image under f and thus any communication protocol with low cost is bound to disagree with f at many points. Discrepancy is thus used primarily to prove lower bounds on the communication complexity of approximations to a function but, for our purposes, we will only use the following basic lemma, whose proof can be found in e.g. [KN97, BNS92, Gro92].

Lemma 4.20 *For any* $f : X_1 \times \ldots \times X_k \to \{1, \omega, \ldots, \omega^{p-1}\}$,

$$R_k(f) = \Omega(\log(1/Disc_k(f))).$$

It should be noted that the discrepancy does not necessarily lead to optimal lower bounds (see examples in [KN97]). Furthermore, computing good upper bounds for $Disc_k(f)$ can prove to be quite difficult.

Let $x_1, \ldots, x_k$ be n-bit vectors, $x_i = b_i^1 b_i^2 \ldots b_i^n$. We define the k-WISE GENERALIZED INNER PRODUCT modulo p (or $GIP_{k,p}$) as the function of k n-bit vectors such that

$$GIP_{k,p}(x_1, \ldots, x_k) = \begin{cases} 1 & \sum\limits_{1 \le j \le n} x_1^j x_2^j \ldots x_k^j \equiv 0 \pmod{p}; \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $GIP_{k,p}$ is the language of k by n matrices that have a number of all-1 columns divisible by p. Of course, for p prime, we can also consider the non-Boolean version of $GIP_{k,p}$ which maps inputs $(x_1, \ldots, x_k)$ to $\omega^{\sum\limits_{1 \le j \le n} x_1^j x_2^j \ldots x_k^j}$.

Theorem 4.21 ([BNS92, Gro92]) *For all k and p holds $R_k(GIP_k, p) = \Omega(n)$.*

The original proof explicitly computed a $O(c^{-n})$ upper bound on the discrepancy of the non-Boolean version of $GIP_{k,p}$ for prime p. Historically, this was first established in the case $p = 2$ by L. Babai, N. Nisan and M. Szegedy [BNS92] and later generalized in [Gro92] to arbitrary p. Both proofs are somewhat intricate and are based on induction on k.

It was later shown by F. Chung and P. Tetali [Chu90, CT93] and R. Raz [Raz00] that this bound could be calculated with far less effort, at least in the case $p = 2$. It is unclear whether these techniques have an analog for the case $p > 2$.

We want to apply the discrepancy technique in order to lower bound the multiparty communication complexity of the following variant of the generalized inner product modulo p (in its non-Boolean version). We define the k-wise truncated inner product modulo p or $TGIP_{k,p}$ for short, as the function which maps k n-bit vectors $x^1, \ldots, x^k$ and an index $s \in [n]$ as $TGIP_{k,p}(x^1, \ldots, x^k, s) = \omega^{\sum_{i=1}^{i=s}(x_i^1 x_i^2 \ldots x_i^k)}$. In other words, $TGIP_{k,p}$ is $GIP_{k,p}$ computed on the inputs

$x^1, \ldots, x^k$ truncated after s bits. Note also that for any $x_1, \ldots x_k \in \{0,1\}^n$ and any $s \in [n]$ we have $TGIP_{k,p}(x_1, \ldots, x_k, s) = GIP_{k+1,p}(x_1, \ldots, x_k, 1^s 0^{n-s})$.

We will use a result of V. Grolmusz [Gro92] concerning the discrepancy of $GIP_{k,p}$. Let us define

$$\Delta_k(n) = \max_{\phi_1,\phi_2,\ldots,\phi_k} |\mathbf{E}_{\vec{x}}[GIP_{k,p}(\vec{x})\phi_1\phi_2\ldots\phi_k]|$$

where $\vec{x} = (x_1, \ldots, x_k) \in (\{0,1\}^n)^k$ and ϕ_i is shorthand for $\phi(\vec{x})$ and where the maximum is taken over all $\phi_i : (\{0,1\}^n)^k \to \{0,1\}$ such that $\phi_i(x_1, \ldots, x_k)$ does not depend on x_i. The expected value $\mathbf{E}_{\vec{x}}$ is taken on the uniform distribution over $(\{0,1\}^n)^k$.

Note that the function $\phi_1\phi_2\ldots\phi_k$ is the indicator function for some cylinder intersection and thus $\Delta_k(n)$ is exactly the discrepancy of $GIP_{k,p}$ on inputs of length n.

Lemma 4.22 ([Gro92]) *For all k, there exists $d > 1$ such that $\Delta_k(n) \leq d^{-n}$.*

We adapt Grolmusz's proof of this lemma and use its result to show:

Lemma 4.23 *For any k and any prime p,*

$$Disc(TGIP_{k,p}) = O(1/\sqrt{n}).$$

Proof. Similarly, to $\Delta_k(n)$, we define

$$\Xi_k(n) = \max_{\phi_1,\phi_2,\ldots,\phi_k,\psi} |\mathbf{E}_{\vec{x},s}[TGIP_{k,p}(\vec{x}, s)\phi_1\phi_2\ldots\phi_k\psi]|$$

where $\vec{x} = (x_1, \ldots, x_k) \in (\{0,1\}^n)^k$ and ϕ_i and ψ are shorthand for $\phi_i(\vec{x}, s)$ and $\psi(\vec{x}, s)$. The maximum is taken over all $\phi_i, \psi : (\{0,1\}^n)^k \times [n] \to \{0,1\}$ such that $\phi_i(x_1, \ldots, x_k, s)$ does not depend on x_i and $\psi(x_1, \ldots, x_k, s)$ does not depend on s. The expected value $\mathbf{E}_{\vec{x}}$ is taken on the uniform distribution on pairs in $(\{0,1\}^n)^k \times [n]$. Clearly, for inputs of length n we have $Disc(TGIP_{k,p}) = \Xi_k(n)$.

By convexity of expectations, we have

$$\Xi_k(n) \leq \mathbf{E}_{\vec{x}}|\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k \psi]|$$

and since ψ does not depend on s:

$$\Xi_k(n) \leq \mathbf{E}_{\vec{x}}|\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k]|$$

For any complex-valued random variable λ, we have $(\mathbf{E}[|\lambda|])^2 \leq \mathbf{E}[|\lambda|^2]$ by the Cauchy-Schwartz inequality. If in particular we choose

$$\lambda = |\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k]|$$

we obtain

$$\Xi_k(n) \leq (\mathbf{E}_{\vec{x}}|\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k]|^2)^{1/2}.$$

Furthermore $|\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k]|^2$ can be rewritten as

$$(\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k])\mathbf{E}_s[\overline{TGIP_{k,p}}(\vec{x},s)\phi_1 \dots \phi_k]$$

where $\overline{TGIP_{k,p}}$ denotes the complex conjugate of $TGIP$ and we can write

$$\Xi_k(n) \leq (\mathbf{E}_{\vec{x}}(\mathbf{E}_s[TGIP_{k,p}(\vec{x},s)\phi_1 \dots \phi_k])(\mathbf{E}_s[\overline{TGIP_{k,p}}(\vec{x},s)\phi_1 \dots \phi_k]))^{1/2}.$$

For any s, t in $[n]$ we write ϕ_i^s (resp. ϕ_i^t) to denote the restriction of ϕ_i where the last input is s (resp. t). Furthermore for fixed $s \leq t$ we will "split" the k-tuple of vectors $\vec{x}$ as $\vec{y}$ and $\vec{z}$ as follows: If the i^{th} coordinate of $\vec{x}$ is $x_i = b_1 b_2 \dots b_n$ then the i^{th} coordinate of $\vec{y}$ is $y_i = b_{s+1} b_{s+2} \dots b_t$ and the i^{th} coordinate of $\vec{z}$ is $z_i = b_1 b_2 \dots b_s b_{t+1} \dots b_n$. The crucial observation is that for fixed $s \leq t$ we have

$$\overline{TGIP_{k,p}}(\vec{x},s) \cdot TGIP_{k,p}(\vec{x},t) = GIP_{k,p}(\vec{y}).$$

Indeed, $TGIP_{k,p}(\vec{x},t) = TGIP_{k,p}(\vec{x},s) \cdot GIP_{k,p}(\vec{y})$ and

$$TGIP_{k,p}(\vec{x},s)\overline{TGIP_{k,p}}(\vec{x},s) = |TGIP_{k,p}(\vec{x},s)|^2 = 1.$$

So we can now bound $\Xi_k(n)$ as:

$$\Xi_k(n) \leq (\mathbf{E}_{s\leq t}\mathbf{E}_{\vec{z}}\mathbf{E}_{\vec{y}}[GIP_{k,p}(\vec{y})\phi_1^s\phi_1^t \ldots \phi_k^s\phi_k^t])^{1/2}.$$

Note that it is sufficient to consider the expectation over the pairs $s \leq t$ since the case $s \geq t$ is completely symmetric. For fixed $s \leq t$ and fixed $\vec{z}$ of length $s+n-t$, define $\zeta_i : (\{0,1\}^{t-s})^k \to \{0,1\}$ as $\zeta_i(\vec{y}) = \phi_i^s(\vec{x})\phi_i^t(\vec{x})$ where $\vec{x}$ is split as $\vec{y}$ and $\vec{z}$ for that choice of s and t. We can now write

$$\Xi_k(n) \leq (\mathbf{E}_{s\leq t}\mathbf{E}_{\vec{z}}[\mathbf{E}_{\vec{y}}[GIP_{k,p}(\vec{y})\zeta_1 \ldots \zeta_k]]^{1/2}.$$

Now ζ_i does not depend on the i^{th} component of $\vec{y}$ and we can thus use Lemma 4.22 to claim that there exists d such that $|[\mathbf{E}_{\vec{y}}[GIP_{k,p}(\vec{y})\zeta_1 \ldots \zeta_k]| \leq d^{-j}$ when we consider $\vec{y}$'s of length j.

Thus, the value of $\mathbf{E}_{\vec{z}}[\mathbf{E}_{\vec{y}}[GIP_{k,p}(\vec{y})\zeta_1 \ldots \zeta_k]$ depends most crucially on how far apart s and t are. We now use this bound in our estimate on $\Xi_k(n)$:

$$\begin{aligned}
\Xi_k(n) &\leq (\mathbf{E}_{s\leq t}d^{-(t-s)})^{1/2} \\
&\leq (\sum_{j=0}^{j=n-1} Pr[t-s=j]\cdot d^{-j})^{1/2} \\
&\leq (\sum_{j=0}^{j=n-1} (n-j)/n^2 \cdot d^{-j})^{1/2} \\
&\leq (1/n \sum_{j=0}^{j=n-1} d^{-j})^{1/2} \\
&= O(1/\sqrt{n}).
\end{aligned}$$

And so our claim about $Disc(TGIP_{k,p})$ is proved. □

As an immediate corollary to the latter two lemmas we obtain from Lemma 4.20:

Theorem 4.24 *For all k and any prime p*

- $D_{k+1}(TGIP_{k,p}) = \Omega(\log n)$;
- $D_k(GIP_{k,p}) = \Omega(n)$;

Notice that the lower bound for $TGIP$ is of course tight. Indeed, suppose player $k+1$ has s written on his forehead: it suffices for any other player to send him the $\log n$ bits of s for player $k+1$ to know the entire input and thus be able to output the correct value of $TGIP$.

4.3.3 Multiparty Complexity Bounds for Regular Languages and Finite Monoids

Similarly to the two-party case, we define the k-party communication complexity of a finite monoid M as the k-party complexity of evaluating in M the product $m_1 \cdot m_2 \cdot \ldots \cdot m_{kn}$ where the $m_i \in M$ is written on the forehead of player j where $j \equiv i \pmod{k}$.

Similarly, the k-party communication complexity of a regular language $L \subseteq A^*$ is the k-party complexity of determining whether the word $a_1 a_2 a_3 \ldots a_{kn}$ lies in L, with $a_i \in A \cup \{\epsilon\}$ written on the forehead player j's where $j \equiv i \pmod{k}$.

It is easy to show that for any $k \geq 2$ the $(k+1)$-party communication complexity of a regular language L (resp. of a monoid M) is at most its k-party complexity. We can also rework the proof for the two-party case (see also [RTT98]) to obtain the elementary facts:

Lemma 4.25 *Let $L \subseteq A^*$ be regular with $M(L) = M$. For any k, we have $D_k(M) = \Theta(D_k(L))$ and similarly for $D_k^{||}$, R_k and $N_k^{Mod_p}$ for p prime.*
For any increasing $f : \mathbb{N} \to \mathbb{N}$ and any k the class of monoids such that $D_k(M)$ (resp. $D_k^{||}(M)$, $R_k(M)$, $N_k^{Mod_p}$ for p prime) is $O(f)$ forms a variety.

We define the following generalizations of the two-party rectangular reductions from a language to a monoid:

Definition 4.26 *A k-dimensional hyper-rectangular reduction of length t from a language*[4] *$L \subseteq (A^n)^k$ to a monoid M is a sequence of kt functions $(s_1, \dots, s_{tk})$ where the s_i's are functions from A^n to M such that $(x_1, \dots, x_k) \in (A^n)^k$ belongs to L if and only if*

$$s_1(x_1)s_2(x_2)\dots s_j(x_{(j \bmod k)})\dots s_{kt-1}(x_{k-1})s_{kt}(x_k).$$

In other words, such a reduction maps a k-tuple $(x_1, \dots, x_k) \in (A^n)^k$ to a sequence of kt monoid elements where the j^{th} monoid element is obtained solely as a function of x_i where $i \equiv j \pmod k$.

We will write $L \leq^t_{r^k} M$ to denote the existence of a k-dimensional hyper-rectangular reduction from L to M. A program is a special form of hyper-rectangular reduction.

We have already mentioned that the multiparty model often has surprising power and is much harder to analyze than the two-party models. Ideally, we would like to obtain complete classifications similar to the ones of Section 4.2 but we only have partial results in this direction.

We begin by sketching the proof of a complete characterization for the k-party deterministic and probabilistic communication complexity of groups which first appeared in [RTT98, Tes99].

Theorem 4.27 *Let G be a group. If G is in $\mathbf{G_{nil,k}}$ then $D_{k+1}(G) = O(1)$. Otherwise $R_{k+1}(G) = \Omega(n)$.*

Proof. The upper bound is a result of the combinatorial description of languages whose syntactic monoids are nilpotent groups of class k (Theorem 2.13): if $M(L)$ is a nilpotent group of class k then membership in L can be determined by counting the number of occurrences of each subword of length at most k

[4]In general of course, we might need to consider languages consisting of k-tuples of inputs of *different* lengths. The definition can clearly be adapted for such cases at the cost of extra sub/superscripts.

modulo some integer m. In the $(k+1)$-party game, any set of k input letters is seen entirely by at least one player so a protocol using $(k+1)\cdot\lceil\log m\rceil$ can easily be devised to count the number of occurrences of a particular subword of length at most k.

The lower bound generalizes the idea of part 5 of Lemma 4.11. If G is not nilpotent of class k, there exists a commutator g of weight $k+1$ which is not the identity. By Lemma 3.3 there exists a G-program ϕ taking $k+1$ bits as input and such that

$$\phi(x_1x_2\dots x_{k+1}) = \begin{cases} g & \text{if all } x_i \text{ are on;} \\ 1_G & \text{otherwise.} \end{cases}$$

By concatenating n such programs, we can obtain a G-program ψ of linear length to recognize $GIP_{k+1,m}$ and this hyper-rectangular reduction shows that $R_{k+1}(G) = \Omega(n)$. $\square$

To analyze the general case, we introduce an alternative parametrization of languages recognized by monoids in **DO** by defining for any group G a family of congruences $\approx^G_{s,t}$ on A^* for any $s,t\in\mathbb{N}$. First, for any t, we let $x\approx^G_{0,t} y$ for all x,y. Then recursively, we define $x\approx^G_{s,t} y$ if and only if

1. x and y are G-equivalent;
2. $x\approx^G_{s-1,t} y$;
3. $\alpha_t(x)=\alpha_t(y)$;
4. For all $x=x_0ax_1$ and $y=y_0ay_1$ with $|x_0|_a=|y_0|_a\le t$, we have $x_0\approx^G_{s-1,t} y_0$ and $x_1\approx^G_{s-1,t} y_1$;
5. For all $x=x_0ax_1$ and $y=y_0ay_1$ with $|x_1|_a=|y_1|_a\le t$, we have $x_0\approx^G_{s-1,t} y_0$ and $x_1\approx^G_{s-1,t} y_1$.

One can check that for all s,t, $\approx^G_{s,t}$ is a well defined finite index congruence. This congruence is quite close to $\sim^G$ defined in Chapter 2: where in $\sim^G$ we were primarily concerned with the first and last occurrence of each letter, we look here at the first and last t occurrences of each letter. Note however that $\approx^G_{s,1}$ is *not* equal to $\sim^G_{|A|,s}$ because, the induction base and the recursion on the prefixes and suffixes is slightly different in the two definitions. For example, one can verify that for any commutative G we have $ab \approx^G_{1,1} ba$ but $ab \not\approx^G_{2,1} ba$.

Not so surprisingly, given the similarity to $\sim^G$, we can show that $\approx^G_{s,t}$ also parametrizes **DO**:

Theorem 4.28 *Let $M = A^*/\gamma$, with $|A| = n$ and let $\mathbf{H}$ be a variety of groups. Then $M \in \mathbf{DO} \cap \overline{\mathbf{H}}$ if and only if $\approx^G_{s,t} \subseteq \gamma$ for some s, t and some $G \in \overline{\mathbf{H}}$.*

Proof. To show that $A^*/\approx^G_{s,t}$ is in $\mathbf{DO} \cap \overline{\mathbf{H}}$, it suffices to note that the $\approx^G_{s,t}$ classes are all unambiguous concatenations of languages with syntactic monoids in $\mathbf{J_1} \vee \mathbf{H}$.

The converse follows from Theorem 2.20 since, from the definitions of $\approx^G$ and $\sim^G$, we have $\approx^G_{ns,1} \subseteq \sim^G_{n,s}$. □

For any group G, and positive integers s, t, we will denote by $\mathbf{V}_{\approx^{\mathbf{G}}_{\mathbf{s,t}}}$ the variety of monoids $M = A^*/\gamma$ such that $\approx^G_{s,t} \subseteq \gamma$. The following lemma motivates the introduction of the $\approx^G_{s,t}$ congruences in the context of multiparty communication complexity.

Lemma 4.29 *Let G be a nilpotent group of class d. If M lies in $\mathbf{V}_{\approx^{\mathbf{G}}_{\mathbf{s,t}}}$ then $D_{(s+d)}(M) = O(1)$.*

Proof. It is sufficient to establish the upper bound for $A^*/\approx^G_{s,t}$ for any alphabet A and any s, t and we will do so using induction on s. In the case of $s = 0$ there is nothing to prove.

Suppose now $s \geq 1$ and take a set of representatives $[u_i]$ of $A^*/\approx^G_{s,t}$. For each u_i, the players will check whether $w \approx^G_{s,t} u_i$. First, they need to verify that w and u_i are G-equivalent and since G is nilpotent of class d, this requires counting the number of occurrences modulo some p of subwords of length at most d in w[5]. Since the number of players involved is $(s+d) \geq (d+1)$, this can be done at constant cost (see Theorem 4.27). Next, the players can exchange $O(t) = O(1)$ bits to insure that $\alpha_t(w) = \alpha_t(u_i)$. If $s = 1$, we are done because there is no recursive condition to check. If $s \geq 2$, suppose $u_i = v_0 a v_1$ with $|v_0|_a \leq t$. There exists a factorization $w = w_0 a w_1$, with $|v_0|_a = |w_0|_a$ but in order to handle the recursion, the players first need to identify the exact location of the j^{th} a in w (where $|v_0|_a = j - 1$).

To achieve this, each player sends a list of identities of the players they think *ignore* the first j a's. This requires only $O((s + d + 1) \log t) = O(1)$ communication. Of course, only the player who has the first a on his forehead will incorrectly identify the first member in that list (say this player is Player l), while all others will agree on designating him as the one ignoring the first a. Since $s \geq 2$, there are at least 3 players involved in the protocol, so Player l can indeed be identified as the only one disagreeing with the majority. We can correct his list by adding an l in the first position and shifting the rest of his list right. Now, the second positions in the lists of all but one of the players agree and we can repeat this procedure for j rounds. In the end, all players know which player ignores the j^{th} a and all except that player know the location of that a. The protocol can now sideline this player and let the other $s + d - 1$ players check whether $v_0 \approx^G_{s-1,t} w_0$ and $v_1 \approx^G_{s-1,t} w_1$. By induction this is doable with $O(1)$ communication. Left-right symmetry completes the protocol. □

Consequently, any language in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ has constant communication complexity for some large enough (but fixed) k. This in fact characterizes $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ for, as we will see next, both B_2 and U can be shown to have $\omega(1)$ communication complexity in every k-party model.

[5] Alternatively the players could do this by comparing the images of w and u_i under all possible morphisms from A^* to G. This strategy can be implemented at constant cost from Theorem 4.27.

Lower bounds for U were proved in [RTT98] using the universality of U. We will present an alternative proof in the same spirit which, in addition, explicitly establishes, to our knowledge, the first multiparty lower bound for a natural generalization of the Disjointness function. For n-bit vectors $x_1, \ldots, x_k$, we define the k-WISE EMPTY INTERSECTION PROBLEM EI_k as the Boolean function such that $EI_k(x_1, \ldots, x_k) = 0$ if and only if the $k \times n$ matrix with rows x_i has no all-1 column. In other words, if we think of the x_i's as representing subsets of $\{0, 1\}$, then $EI_k(x_1, \ldots, x_k) = 1$ if and only if $x_1 \cap x_2 \cap \ldots \cap x_k = \emptyset$. In particular $EI_2 = DISJ$. We will also denote by NEI_k the complement of EI_k.

Lemma 4.30 *For all fixed k we have $D_k(U) = \Omega(\log n)$.*

Proof. Simple counting arguments (see e.g. [Gro93, Gro97]) can show that there exists a language L in $(\{0, 1\}^n)^k$ with k-party communication complexity at least $n - \log n - \log k$. We claim that any such language has a k-dimensional hyper-rectangular reduction of size $|L|$ to NEI_k. Indeed, given some element $\bar{x} = (x_1, \ldots, x_k)$ of $(\{0, 1\}^n)^k$, the k players can easily apply an hyper-rectangular reduction to obtain a $k \times |L|$ boolean matrix $M_{\bar{x}}$ with columns labeled by elements $\bar{y} = (y_1, \ldots, y_k)$ of L and such that entry $(i, \bar{y}) = 1$ if and only if $x_i = y_i$. Thus, a column labeled $\bar{y}$ in this matrix consists of all 1's if and only if $\bar{x} = \bar{y}$. On the other hand $M_{\bar{x}}$ contains a column labeled $\bar{x}$ if and only if $\bar{x} \in L$.

Since L has k-party complexity at least $n - \log n - logk$ and reduces in length $|L| \leq 2^{kn}$ to NEI_k, we get $D_k(NEI_k) \geq \log n/k$.

We get the lower bound for U by noticing that for any k, the language NEI_k can be computed by a linear length program over U. □

No sub-linear upper bounds are known for the k-party communication complexity of U and EI_k and it is tempting to conjecture that our lower bounds are far from optimal.

To obtain a lower bound for the multiparty complexity of B_2, we will appeal to a Ramsey-theoretical result known as the Hales-Jewett Theorem [GRS80] and which concerns colorings of $[t]^n$ where $t \in \mathbb{N}$. We say that the vectors $v^1, \ldots, v^t \in [t]^n$ form a *combinatorial line* if at each position i they either agree (i.e. for all $1 \le j, j' \le t$ we have $v_i^j = v_i^{j'}$) or are such that $v_i^j = j$. We now state the theorem:

Theorem 4.31 (Hales-Jewett) *For any integers c, t there exists an integer n such that if all vectors in $[t]^n$ are colored with c colors then there is a monochromatic combinatorial line $v^1, \ldots, v^t$ (i.e. a line whose elements all were assigned the same color).*

The Hales-Jewett number $HJ(c,t)$ is naturally defined as the minimal such n. While the theorem's proof implicitly provides an upper bound in terms of c and t, these bounds are not primitive recursive. Although the lower bound for E_{2^n} cited earlier is not proved using the Hales-Jewett itself, it uses a Ramsey-theoretical result with a similar flavor. Chapter 29 of [Juk01] describes this and other theoretical computer science related applications of the Hales-Jewett theorem and its variants.

Lemma 4.32 *For any fixed k we have $D_k(B_2) = \omega(1)$.*

Proof. We will in fact prove the lower bound for the function k-SET-PARTITION (or $Part_k$ for short) which we define as follows: Let $S_1, \ldots, S_k$ be subsets of $[n]$ represented as $[n]$-bit vectors, then $Part_k(S_1, \ldots, S_k) = 1$ if and only if these sets are a partition of $[n]$, i.e. if the bitwise sum of the vectors is the all-1 vector.

Every input $(S_1, \ldots, S_k) \in \mathcal{P}([n])^k$ that is *accepted* by a protocol for $Part_k$ is such that for every $1 \le j \le n$, the element j lies in exactly one of the S_i's. Using this observation, these inputs can be put in one-to-one correspondence with n-tuples in $[k]^n$. As an example for $k = 3$ and $n = 4$, we have $Part_3(\{4\}, \{1,3\}, \{2\}) = 1$ and this input corresponds to the n-tuple $(2, 3, 2, 1)$.

Suppose that the communication complexity of $Part_k$ is bounded, for some k, by a constant c. To every input accepted by a protocol for $Part_k$, (i.e. to every element in $[k]^n$), we can assign one of 2^c colors corresponding to the communication history resulting from that particular input. If n is large enough, there must be, by the Hales-Jewett Theorem, a monochromatic combinatorial line $v^1, \ldots, v^k$, where the v^i's are in $[k]^n$. If we let $\emptyset \neq T \subseteq [n]$ be the set of coordinates on which the v^i's disagree, we get that there are sets $S_1, \ldots, S_k$ such that $T \cup S_1 \cup \ldots \cup S_k = [n]$ and all the inputs $(S_1 \cup T, S_2, \ldots S_k)$, $(S_1, S_2 \cup T, \ldots S_k)$, $\ldots, (S_1, S_2, \ldots S_k \cup T)$ induce the same communication history. However this is a contradiction: By the star property mentioned earlier, the input $(S_1, S_2, \ldots S_k)$ *also* induces that same communication although it should be rejected since $S_1 \cup \ldots \cup S_k = [n] - T \neq [n]$.

We complete the proof by showing an easy reduction from $Part_k$ to B_2: the reduction is obtained by concatenating n blocks of $k+2$ elements of B_2, such that the j^{th} block is $am_{1,j} \ldots m_{k,j}ab$ where $m_{i,j}$ is b if j lies in S_i but $m_{i,j}$ is 1 otherwise. It is easy to see that the j^{th} block thus created will evaluate to ab if j lies in exactly one S_i and to 0 otherwise. □

In contrast to U, there are known non-trivial upper bounds on the multiparty communication complexity of B_2. K. Reinhardt has shown upper bounds [Rhe01] which are good examples of the surprising possibilities offered by the multiparty model and we sketch one of these bounds for completeness.

Lemma 4.33 (Reinhardt) *- $D_4(B_2) = O(\sqrt{n} \log n)$;*
- $D_5(B_2) = O(\log n)$.

Proof. We exhibit a $O(\sqrt{n} \log n)$ 4-player protocol for the language $L = (c^*ac^*bc^*)^*$ which is sufficient since its syntactic monoid is B_2. The 5-party protocol is based on the same ideas. We can assume without loss of generality that the players receive an input word belonging to $c^*aA^*bc^*$ and thus only need verify that there exists an occurrence of b between any two consecutive occurrences of a and vice-versa. In particular, if the input is to be accepted, then there must be, between any two occurrences of a a number of b's exceeding the number of a's by exactly one.

Players 1 and 2 consider the intervals defined by two a's occurring on the foreheads of players 3 and 4 and write a list[6] $i_1, \ldots, i_t$ of possible lengths for the intervals. If there are d intervals, the maximum of all i_j times d must exceed n and so $t = O(\sqrt{n})$. Next, for each i_j, they exchange $O(\log n)$ bits to determine the number of b's minus the number of a's that they see occurring in such intervals of length i_j. If they find that this number is not equal to the number of intervals of length i_j the protocol halts and rejects.

A similar procedure is repeated for intervals defined by two b's held by player 3 or 4 and for every pair of players. If all these steps are completed successfully, the protocol accepts.

It is clear that no rejected input x is in the language. Conversely, suppose that x is not in the language, i.e. that x contains a subsegment of the form ac^*a (or bc^*b), and let h be the minimal length of such a segment. If the protocol accepts x nonetheless, it must be that some other segment w of length h and delimited by two a's contained at least two more b's than a's. It must therefore be that w contained a segment of the form bc^*b, but this contradicts the minimality of h. □

As a corollary to Theorem 4.37, we know $D_3(B_2) = \Omega(\log\log\log n)$ but for $k \geq 4$, there is a huge gap between these upper bounds and our best lower bounds (non primitive recursive) for the k-party communication of B_2.

Our results thus far allow us to give an algebraic characterization of languages for which there exist constant cost protocols when enough players are involved.

[6]This need not be done explicitly but it simplifies the protocol's description and analysis.

Theorem 4.34 *There exists a constant k such that $D_k(M) = O(1)$ if and only if $M \in \mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$.*

Proof. The upper bound is provided by Lemma 4.29. The converse is obtained by noticing that any M *not* in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ can be divided either by a non-nilpotent group (Lemma 4.27), by U (Lemma 4.30) or by B_2 (Lemma 4.32). □

This theorem unfortunately fails to provide algebraic characterizations of monoids with bounded complexity in the k-party game for specific values of k. We are guaranteed by Theorem 4.25 that such characterizations exists and we have already established one for $k = 2$ in Section 4.2.

Let us first consider the case $k = 3$. By Lemma 4.29, we can evaluate in constant 3-party complexity

- the product in any monoid M in $\mathbf{V}_{\approx^{\mathbf{G}}_{\mathbf{0,t}}}$ when G is a nilpotent group of class 2;
- the product in any monoid N in $\mathbf{V}_{\approx^{\mathbf{H}}_{\mathbf{1,t}}}$ for a commutative group H;
- the product in any $M \times N$ where M, N are as above.

We will show that this in fact captures exactly the limits of the 3-player game. Intuitively, the class of languages we have just implicitly defined is captured by the following congruence on A^*: we set[7] $x \sim_{t,p} y$ if and only if:

1. $\alpha_t(x) = \alpha_t(y)$;
2. the number of subwords of length at most 2 in x and y coincide modulo p;
3. For any $x = x_0 a x_1$ with $|x_0| < t$, there is $y = y_0 a y_1$ with $|x_0|_a = |y_0|_a$ and such that $\alpha_{t,p}(x_0) = \alpha_{t,p}(y_0)$ and $\alpha_{t,p}(x_1) = \alpha_{t,p}(y_1)$.

[7]The congruence $\sim$ defined here is *not* the same as the $\sim^G$ congruence defined in Chapter 2 and used to parametrize **DO**.

4. For any $x = x_0ax_1$ with $|x_1| < t$, there is $y = y_0ay_1$ with $|x_1|_a = |y_1|_a$ and such that $\alpha_{t,p}(x_0) = \alpha_{t,p}(y_0)$ and $\alpha_{t,p}(x_1) = \alpha_{t,p}(y_1)$.

Let $\mathbf{B_3}$ be the variety of monoids M satisfying

1. $M \in \mathbf{DO}$;
2. $M \in \overline{\mathbf{G_{nil,2}}}$;
3. for all w lying $\mathcal{J}$-above idempotents e, f and any u, v holds
$$euw^\omega vf = euvf;$$
4. for all x, y lying $\mathcal{J}$-above idempotents e, f and any z holds
$$ezx^{\omega-1}y^{\omega-1}xyf = ex^{\omega-1}y^{\omega-1}xyzf.$$

The congruence $\sim_{t,p}$ and the variety $\mathbf{B_3}$ are quite similar to respectively the congruence $\approx_{t,p}$ and $\mathbf{W}$ defined in Section 4.2. In fact it is clear that $\mathbf{W} \subseteq \mathbf{B_3}$.

Lemma 4.35 *Let $M = A^*/\gamma$, then $M \in \mathbf{B_3}$ if and only if $\sim_{t,p} \subseteq \gamma$ for some t, p.*

Proof. The proof is similar to the one of Lemma 4.8: if $M = A^*/\sim_{t,p}$, then M clearly satisfies conditions 1 and 2 for membership in $\mathbf{B_3}$. Furthermore, to check 3 and 4 one can easily verify that the words $q_1 = (swt)^{tp}uw^{tp}v(xwy)^{tp}$ and $r_1 = (swt)^{tp}uv(xwy)^{tp}$ are $\sim_{t,p}$-equivalent and that $q_2 = (sxyu)^{tp}zx^{p-1}y^{p-1}xy(vxyw)^{tp}$ and $r_2 = (sxyu)^{tp}x^{p-1}y^{p-1}xyz(vxyw)^{tp}$ are $\sim_{t,p}$-equivalent. So M lies in $\mathbf{B_3}$.

For the converse, assume M lies in $\mathbf{B_3}$. We need to show that there exist t, p such that for any morphism $\phi : A^* \to M$ we have $\phi(q) = \phi(r)$ for any $q \sim_{t,p} r$. We will choose p as the exponent of M and t as $|M| + 1$.

Suppose $A = \{a_1, a_2, \ldots, a_k\}$. For any $x \in A^*$, denote by $\hat{x}$ the word obtained from x by the following process:

- as a first step, we mark the first and last t occurrences in x of any letter a_i;

- next, suppose y is a segment of x lying between two consecutive marked letters. Note that every letter of $\alpha(y)$ occurs at least t times to the left and right of y. We replace y by the word

$$y' = a_1^{m_1} \dots a_k^{m_k} [a_2, a_1]^{n_{2,1}} [a_3, a_1]^{n_{3,1}} \dots [a_k, a_{k-1}]^{n_{k,k-1}}$$

  where $[a, b]$ is shorthand for the pseudo-commutator $b^{p-1}a^{p-1}ba$, and where the $0 \leq m_i \leq p-1$ and $0 \leq n_{i,j} \leq p-1$ are such that the number of subwords of length 2 in y and y' are equal modulo p. Note that we have $m_i = 0$ and $n_{i,j} = 0$ if a_i does not belong to $\alpha(y)$.

- Finally, we move every pseudo-commutator $[a_i, a_j]$ introduced in the previous step next to the leftmost marked position where the prefix contains t occurrences of both a_i and a_j and reduce the block of $[a_i, a_j]$'s thus created modulo p.

We have $x \sim_{t,p} \hat{x}$ of course. Indeed, replacing y by y' in the second step has no effect on the overall count of subwords of length 2. Furthermore, step 2 and step 3 preserve the number of occurrences (modulo p and threshold t) of each letter before or after any of the marked positions because $[a_i, a_j]$ has 0 occurrences of any letter modulo p. It can also be shown conversely that if $x \sim_{t,p} y$, then $\hat{x} = \hat{y}$.

It now suffices to prove that for any morphism $\phi : A^* \to M$ we have $\phi(x) = \phi(\hat{x})$. Suppose y is a segment of x lying between two consecutive marked letters in $x = x_0aybx_1$. Every letter of $\alpha(y)$ occurs at least t times in both x_0 and x_1 since y itself contains no marked letters. So by the argument of Lemma 4.8, we can show for any pair of letters $a_i, a_j \in \alpha(y)$ that both $\phi(x_0)$ and $\phi(x_1)$ can be written as res where e is an idempotent lying below a_i, a_j. Since M is in

$\mathbf{B_3}$, we have for any e, f idempotents below $\phi(a_i), \phi(a_j)$ and any u, v, w both $euvf = eu\phi(a_i)^p vf$ and

$$eu\phi(a_i)^{p-1}\phi(a_j)^{p-1}a_i a_j vwf = euv\phi(a_i)^{p-1}\phi(a_j)^{p-1}a_i a_j wf.$$

Supposing $y = y_0 a_i a_j y_1$ we can successively apply these rules to get:

$$\begin{aligned}
\phi(x) &= \phi(x_0 a y_0 a_i a_j y_1 b x_1) \\
&= \phi(x_0 a y_0 \mathbf{a_j^p a_i a_j y_1 b x_1}) \\
&= \phi(x_0 a y_1 a_j \mathbf{a_i^p a_j^{p-1} a_i a_j y_1 b x_1}) \\
&= \phi(x_0 a y_1 a_j a_i [a_j, a_i] y_1 b x_1) \\
&= \phi(x_0 a y_1 a_j a_i y_1 [\mathbf{a_j}, \mathbf{a_i}] \mathbf{b x_1})
\end{aligned}$$

This suffices to show that step 2 in our production of $\hat{x}$ can be done without affecting the image under ϕ. Similarly, step 3 can also be done without affecting the image under ϕ. □

We will show that monoids in $\mathbf{B_3}$ are exactly the ones with bounded 3-party communication complexity. One crucial tool for our argument will be a result of P. Pudlák [Pud03].

Theorem 4.36 *Let the In-Between function $IBF : \{0,1\}^n \times [n] \times [n] \to \{0,1\}$ be defined as*

$$IBF(x, r, s) = \begin{cases} 1 & \text{if } r < s \text{ and one of the bits } x_{r+1}, \ldots, x_s \text{ is 1;} \\ 0 & \text{else.} \end{cases}$$

Then we have $D_3^{||}(IBF) = \Omega(\log\log n)$ and thus $D_3(IBF) = \Omega(\log\log\log n)$.

It is worth noting that this result implies an $\Omega(\log\log n)$ lower bound on the three-party communication complexity of B_2 whereas the techniques of Lemma 4.32 only yield $\omega(1)$. We can now prove

Theorem 4.37 *For all $M \in \mathbf{B_3}$ holds $D_3(M) = O(1)$. If $M \notin \mathbf{B_3}$ then $D_3(M) = \Omega(\log\log\log n)$.*

Proof. In one direction it suffices to prove the upper bound for the $\sim_{t,p}$-classes. Let u be some word of A^* and suppose that 3 players are given $x \in A^*$. To check $x \sim_{t,p} u$, they first need to compare $\alpha_t(x)$ and $\alpha_t(u)$, which requires only $2t \cdot |A|$ bits. Next, they need to compare the number of occurrences of subwords of length 2 modulo p, which we have already argued can be done using $3p \cdot |A|^2$ bits of communication in the three-party model. Finally, if $u = u_0 a u_1$ with $|u_0|_a = i < t$, the three parties can determine in $O(1)$ communication which of them holds the $(i+1)^{\text{th}}$ occurrence of a in x (see the proof of Lemma 4.29). The two other players thus know how to factor x as $x_0 a x_1$ with $|x_0|_a = i$ and can verify that $\alpha_{t,p}(x_0) = \alpha_{t,p}(u_0)$ and $\alpha_{t,p}(x_1) = \alpha_{t,p}(u_1)$. Left-right symmetry completes the argument.

Conversely, suppose M is not in $\mathbf{B_3}$. If M is not in $\mathbf{DS}$, then it is divided by either U or B_2 and so $D_3(M) = \Omega(\log\log\log n)$.

If M lies in $\mathbf{DS}$ but not in $\mathbf{DO}$, we claim that $GIP_{3,q}(x,y,z)$ has a linear length reduction to M. The argument is almost exactly the one in the proof of Lemma 4.11: there must exist two $\mathcal{J}$-related idempotents $e, f \in M$ such that ef is not idempotent and $efe \neq e = (ef)^\omega e$. Let q be minimal such that $(ef)^q e = e$ and suppose without loss of generality that the $|x| = |y| = |z| = n$ is a multiple of q. Our reduction produces elements $m_1 \dots m_{3n}$ where $m_{3i-2} = e$ if $x_i = 1$ and $m_{3i-2} = ef$ otherwise; $m_{3i-1} = 1_M$ if $y_i = 1$ and $m_{3i-1} = f$ otherwise; $m_{3i} = e$ if $z_i = 1$ and $m_{3i} = fe$ otherwise. In particular the product $m_{3i-2}m_{3i-1}m_{3i}$ is e if and only if $x_i = y_i = z_i = 1$ and is efe otherwise so the product $m_1 \dots m_{3n}$ equals $(efe)^{n - \sum_{1 \le i \le n} x_i y_i z_i} e$ which equals e if and only if $GIP_{3,q}(x,y,z) = 1$. Since $D_3(GIP_{3,q}) = \Omega(n)$, we have $D_3(M) = \Omega(n)$.

If M is not in $\overline{\mathbf{G_{nil,2}}}$, then by Theorem 4.27 we have $D_3(M) = \Omega(n)$.

Finally, suppose M lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil,2}}}$ but not in $\mathbf{B_3}$. We consider two cases:

suppose first that for some w lying $\mathcal{J}$-above idempotents e, f and for some u, v we have $euw^\omega vf \neq euvf$. Then $IBF(x,r,s)$ admits the following linear length reduction to M. The reduction gives elements $m_1, \ldots, m_{3n+3}$:

$$m_i = \begin{cases} e & \text{for } i = 1 \\ f & \text{for } i = 3n+3 \\ eu & \text{for } i = 3r+2; \\ ve & \text{for } i = 3s+1; \\ 1_M & \text{for all other } i \text{ with } i \not\equiv 0 \pmod 3; \\ w^\omega & \text{for } i = 3j \text{ such that } x_j = 1; \\ 1_M & \text{for even } i = 3j \text{ such that } x_j = 0; \end{cases}$$

We have $w^\omega w^\omega = w^\omega$ and, since M lies in $\mathbf{DO}$, both $ew^\omega e = e$ and $fw^\omega f = f$. Therefore, the product $m_1 \ldots m_{3n+3}$ is equal to $euvf$ if and only if all of $m_{3r+3}, \ldots, m_{3s}$ are 1_M which occurs if and only if $x_{r+1}, \ldots, x_s$ are 0, i.e. if and only if $IBF(x,r,s) = 0$. Otherwise, $m_1 \ldots m_{3n+3}$ is $euw^\omega vf$. From the IBF lower bound we thus get $D_3(M) = \Omega(\log\log\log n)$.

Finally, suppose that M satisfies $euw^\omega vf = euvf$ for each w lying $\mathcal{J}$-above e and f but there exist $u, v \in M$ lying $\mathcal{J}$-above idempotents e, f and z such that

$$eu^{\omega-1}v^{\omega-1}uvzf \neq ezu^{\omega-1}v^{\omega-1}uvf.$$

In that case, we claim, that for some prime q, $TGIP_{2,q}$ has a linear length reduction to M.

Let q be the smallest positive integer such that $e[v,u]^q = e$ and $[v,u]^q f = f$ and assume without loss of generality that q is prime[8]. Because M is in $\mathbf{DO} \cap \overline{\mathbf{G_{nil,2}}}$, the idempotent e and the pseudo-commutator $[v,u]$ commute with respect to their action on the $\mathcal{R}$-class of e and in particular, $e[v,u] = ee[v,u] = e[v,u]e$. So if $e[v,u]zf \neq ez[v,u]f$, then for all $1 \leq i \leq q-1$ we have

[8]If q is not prime, the proof can be carried through using a reduction from $TGIP_{2,p}$ for some prime divisor of q.

$e[v,u]^i z[v,u]^{q+1-i} f \neq ez[v,u]f$. Indeed, if we assume otherwise, we have:

$$\begin{aligned} ez[v,u]f &= e[v,u]^i z[v,u]^{q+1-i} f \\ &= e[v,u]^i ez[v,u]f[v,u]^{q-i} f \\ &= e[v,u]^i e[v,u]^i z[v,u]^{q-i+1} f[v,u]^{q-i} f \\ &= e[v,u]^{2i} z[v,u]^{q+1-2i} f \end{aligned}$$

By repeating this manipulation, we in fact have for any $k \geq 1$:

$$ez[v,u]f = e[v,u]^{ki} z[v,u]^{q+1-ki} f$$

which leads to a contradiction for $k \geq 1$ such that $ki \equiv 0 \pmod q$.

We now construct the reduction from $TGIP_{2,q}(x,y,s)$ to M and assume for simplicity that the inner product (not truncated) of x and y is equal to 1 modulo q. We build from (x,y,s) the word $e(m_1 \dots m_{6n})f$ in M^* where m_i depends on x if $i \equiv 1 \pmod 3$, on y if $i \equiv 2 \pmod 3$ and on s if $i \equiv 0 \pmod 3$. Specifically, we set

- $m_{3i} = z$ if $i = 2s$ and $m_{3i} = 1_M$ otherwise;
- $m_{6i-5} = u^{\omega-1}$ and $m_{6i-2} = u$ if $x_i = 1$ but $m_{6i-5} = 1_M$ and $m_{6i-2} = 1_M$ if $x_i = 0$;
- $m_{6i-4} = u^{\omega-1}$ and $m_{6i-1} = u$ if $y_i = 1$ but $m_{6i-4} = 1_M$ and $m_{6i-1} = 1_M$ if $y_i = 0$.

One can easily verify that if $t = \sum_{1 \leq i \leq s} x_i y_i$ then the product $e(m_1 \dots m_{6n})f$ evaluates to $e[v,u]^t z[v,u]^{qt+1-t} f$. From our previous remarks, this product is equal to $ez[v,u]f$ if and only if $TGIP_{2,q}(x,y,s) = 1$. This reduction shows that $D_3(M) = \Omega(\log n)$ in this case. □

Characterizations of monoids with bounded k-party complexity for any $k \geq 4$ seem out of reach for now. A first step would be to characterize the class of aperiodic monoids with bounded k-party complexity and we propose the following conjecture.

Conjecture 4.38 *A star-free language L has bounded k-party communication complexity if and only if it is the disjoint union of $\approx^I_{k-1,t}$ for some t where the superscript I denotes the trivial group.*

Since the trivial group is of nilpotency class 1, we can apply Lemma 4.29 to obtain the "if" direction of this conjecture. From the definition of this congruence, we see that $\approx^I_{1,t}$ captures exactly languages with aperiodic and commutative syntactic monoids. This proves our conjecture for $k = 2$ and in the case $k = 3$, it follows as a simple corollary to Theorem 4.37.

A straightforward induction on k shows that for any k, t and any $s \leq k$ the words $x = (a^t_{k+1}a^t_k \dots a^t_1)^s$ and $y = (a^t_{k+1}a^t_k \dots a^t_1)^{s+1}$ are $\approx^I_{s,t}$-equivalent. For $k = 0$, this is trivially true. For $k \geq 1$ it suffices to prove the equivalence for $s = k$. If $x = x_0a_ix_1$ and $y = y_0a_iy_1$ with $|x_0|_{a_i} = |y_0|_{a_i} \leq t$ then in fact $x_0 = y_0$ and there is u such that $x_1 = u(a^t_{k+1}a^t_k \dots a^t_1)^{k-1}$ and $y_1 = u(a^t_{k+1}a^t_k \dots a^t_1)^k$. By our induction hypothesis $(a^t_{k+1}a^t_k \dots a^t_1)^{k-1}$ and $(a^t_{k+1}a^t_k \dots a^t_1)^k$ and thus x_1 and y_1 are $\approx^I_{k-1,t}$ equivalent. Left-right symmetry completes the argument.

In particular, the piecewise-testable language $A^*a_1A^* \dots A^*a_{k+1}A^*$ is not the union of $\approx^I_{k,t}$ classes because $(a^t_{k+1}a^t_k \dots a^t_1)^{k+1}$ contains a subword $a_1a_2 \dots a_{k+1}$ while $(a^t_{k+1}a^t_k \dots a^t_1)^k$ does not. In our communication game, it is of course easy for $k + 1$ players to identify the existence of a subword of length k at constant cost since any occurrence of it will be seen entirely by at least one player. In fact, a lot of our intuition about the power of the $k + 1$-party game revolves around this "free access" to subwords of length k. This leads us to separately formulate a weak special case of Conjecture 4.38:

Conjecture 4.39 *The k-party communication complexity of the regular language $A^*a_1A^* \dots A^*a_kA^*$ is non-constant.*

This statement has been established by P. Pudlák [Pud03] for the cases $k = 2, 3, 4, 5$ (see also [Gas02a] for an explicit treatment of the case $k = 3$) using a Ramsey-theoretical result known as Hindman's Theorem. Its full resolution would be a major step in our understanding of the fundamental limits of the k-party game and would nicely complement Theorem 4.27 which shows that counting, modulo p, the occurrences of subwords of length k can be done at constant cost by $k+1$ players or more but requires $\Omega(n)$ bits of communication for k players or less.

4.4 Applications to Program and Circuit Lower Bounds

We have shown that our algebraic point of view on communication complexity is a fruitful one. Bounds on the communication complexity of monoids allow us on one hand to gain some insight on the relative power of various communication models and, on the other hand, it allows us to identify, as in Conjecture 4.39, concrete functions for which communication complexity lower bounds are most susceptible of being particularly meaningful.

While algebraic tools help in the analysis of communication models, communication complexity results can, in turn, be used to formalize certain arguments in the study of programs over monoids. For instance we can use results of this Chapter to obtain the following:

Theorem 4.40 *The varieties* **Com**, **W**, **DA**, $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ *and* $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ *are all program-varieties.*

Proof. We first prove this for **W**. Suppose M is such that for all $m \in M$, the subset M_m of M^* given by $M_m = \{m_1 \dots m_n : eval_M(m_1, \dots, m_n) = m\}$ can be recognized by polynomial-length programs over **W**. These programs constitute a polynomial length rectangular reduction from the word problem of M to the word problem of some monoid in $\mathbf{W}$. We can therefore conclude that $D^{||}(M) = O(\log n)$ and thus $M \in \mathbf{W}$. Using respectively constant two-party complexity, logarithmic two-party deterministic complexity and constant multiparty complexity, we obtain similar results for **Com**, $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ and $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$.

To show that **DA** is a program-variety, we need to combine these ideas with the fact that aperiodics form a program-variety. □

This fact was already known for **Com** (folklore) and **DA** (proved using a completely different idea in [MPT91]) but the technique used here is very general and the following lemma (first proved in [RTT98]) can be used to obtain such results:

Lemma 4.41 *Let $f = O(\log^r n)$ for some $r > 0$. For any $k \geq 2$, the class of monoids with k-party deterministic communication complexity (resp. probabilistic, simultaneous, MOD_p-counting) forms a program-variety.*

V. Grolmusz implicitly exploited this idea to prove the following result about modular circuits:

Theorem 4.42 ([Gro92]) *For any prime p and any composite integer m that is not a prime power, there exists an explicitly constructible function f computable by depth-2 MOD_m circuits but not computable by any constant depth MOD_p circuit.*

In fact, this result can be obtained as a corollary to the following theorem of D. Barrington, H. Straubing and D. Thérien:

Theorem 4.43 ([BST90]) *Let p be a prime and G a group in $\mathbf{G_p} * \mathbf{Ab}$. There is a constant $c_G > 1$ such that any G-program computing the* AND *of n variables has length $\Omega(c_G^n)$.*

The results obtained previously in this Chapter allow us to obtain a theorem slightly stronger than Grolmusz's but weaker than the latter.

Theorem 4.44 *Let p be a prime and G a group in $\mathbf{G_p} * \mathbf{Ab}$ (or, more generally, any monoid M in $\mathbf{LG_p} Ⓜ \mathbf{Com}$). There is a constant $c_G > 1$ such that any G-program computing $DISJ$ has length $\Omega(c_G^n)$.*

Proof. Suppose for contradiction that $DISJ$ can be recognized by a G-program of sub-exponential length $f(n)$. This program constitutes a length $f(n)$ rectangular reduction from $DISJ$ to G and, since G has MOD_p-counting two-party complexity $O(\log n)$, it allows us to build a $O(\log(f(n))) = o(n)$ MOD_p-counting two-party protocol for $DISJ$. This is in contradiction with the lower bound stated in Table 4.1. □

In their paper, Barrington, Straubing and Thérien propose the constant-degree hypothesis, a conjecture generalizing Theorem 4.43 mentioned above.

Conjecture 4.45 (Constant-degree hypothesis [BST90]) *Let p be prime. If G is a group in $\mathbf{G_p} * \mathbf{G_{nil}}$ then any G-program computing the* AND *function has length $2^{\Omega(n)}$.*

This conjecture is proposed as a first step towards the more ambitious goal of proving that AND cannot be computed in sub-exponential length by programs over any solvable group or, equivalently, in sub-exponential size by CC^0 circuits. Such lower bounds would be the dual of the exponential-size lower bounds for AC^0 circuits computing MOD_p.

Progress towards the constant-degree hypothesis has proved to be quite difficult: circuit lower bounds of V. Grolmusz and G. Tardos [GT00, Gro98] (see also [ST]) can be reformulated to cover very special cases and attempts have been made for the case where the group is the wreath product of a p-group and a nilpotent group of class two [BTT02]. If we seek lower bounds for the

AND function, the communication complexity approach is probably doomed to fail since AND has extremely low communication complexity in every reasonable model but as Theorem 4.44 suggests it might prove fruitful if we target a slightly more complicated function.

Conjecture 4.46 *For any fixed k, the MOD_p-counting k-party communication complexities of EI_k and GIP_q^k are $\Omega(n)$.*

Such lower bounds would immediately imply:

Corollary 4.47 (Assuming 4.46) *For any $k \geq 1$, there is a function which can be computed by a polynomial length program over a group in $\mathbf{G_p} * \mathbf{G_{nil,k+1}}$ but cannot be computed by a sub-exponential length program over any group in $\mathbf{G_p} * \mathbf{G_{nil,k}}$.*
There is a function in AC_2^0 which cannot be computed by any program of sub-exponential length over a group G in $\mathbf{G_p} * \mathbf{G_{nil}}$.

Proof. [sketch] The first observation to make of course is that any group G in $\mathbf{G_p} * \mathbf{G_{nil,k}}$ lies in $\mathbf{LG_p} \circledM \mathbf{G_{nil,k}}$. Since any G' in $\mathbf{G_{nil,k}}$ is such that $D_k(G') = O(1)$, we must have $N_k^{Mod_p}(G) = O(\log n)$. Thus, no sub-exponential length program over G can recognize a function of super-logarithmic k-party MOD_p-counting complexity.

Assuming the conjectured lower bounds, we conclude that EI_k and GIP_q^k cannot be recognized in sub-exponential length by G-programs, when $G \in \mathbf{G_p} * \mathbf{G_{nil,k}}$ while it is an easy exercise to show that they are recognizable in polynomial length by a program over a group in $\mathbf{G_p} * \mathbf{G_{nil,k+1}}$. Furthermore, for every k the function EI_k can be computed by a linear length program over U and thus $N_k^{Mod_p}(U) = \Omega(n)$ for all constant k. Hence U cannot be recognized by a sub-exponential length program over a group in $\mathbf{G_p} * \mathbf{G_{nil}}$. $\square$

While length lower bounds for G-programs AND remain the ultimate goal, it should be noted that no super-linear bounds exist for an explicit function in NP if G is solvable but not in $\mathbf{G_p} * \mathbf{Ab}$.

4.5 Conclusion and Open Problems

We have established a number of classification results for the communication complexity of regular languages and monoids and have shown their importance as means of understanding both the power of communication complexity models and the power of monoids as language recognizers. Our results further stress the importance of the varieties $\mathbf{DO}$ and its subclasses, in particular $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ and $\mathbf{DO} \cap \overline{\mathbf{Ab}}$ which we will again encounter in the context of our next chapter.

We believe that this algebraic approach to communication complexity could and should be explored further and present some open questions pertaining to it.

4.5.1 Towards a Multiparty Analog of Szegedy's Theorem

The *worst-case partition* k-party communication complexity of a language $K \subseteq A^n$ is, as the terminology suggests, the maximum over all k-partitions of $[n] = S_1 \dot\cup S_2 \dot\cup \ldots \dot\cup S_k$ of the k-party communication complexity of determining if $w \in A^n$ belongs to K when player P_i has the letter of w indexed by S_i written on his forehead. We mentioned in this chapter's introduction the following spectacular theorem due to M. Szegedy:

Theorem 4.48 ([Sze93]) *A language K has bounded worst-case partition two-party communication complexity if and only if it can be recognized by a program over a commutative monoid.*

Such a connection between programs and communication complexity is completely unexpected and the proof of this result is difficult. On the other hand, its content is so rich that it is important to consider possible extensions of the theorem.

It seems hopeless to find a "program-over-monoid" characterization of languages with, say, logarithmic two-party communication complexity since MAJORITY has logarithmic communication complexity but presumably cannot be computed by a program over any solvable monoid and provably cannot be computed by a program over a monoid $\mathbf{DO} \cap \overline{\mathbf{Ab}}$. It is quite possible on the other hand that a multiparty analog of Szegedy's Theorem exists. In the rest of this subsection, we want to argue in favor of the following conjecture.

Conjecture 4.49 *For any $L \subseteq A^*$, there exists a constant k such that $D_k(L) = O(1)$ in the worst-case partition if and only if there exists r such that L can be recognized by an r-program over some M in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$.*

One direction of Szegedy's Theorem is quite straightforward: if L can be recognized by a program over a commutative monoid then, regardless of the input partition, Alice can compute the product of the outputs of instructions querying input letters that she has access to. Sending this value to Bob requires only $\log |M|$ bits and this is sufficient information for Bob to determine the value of the program's output. Note that it is crucial to consider only 1-programs for otherwise certain instructions might be querying both input letters known only to Alice and input letters known only to Bob.

Similarly, we can establish the easy half of our conjecture:

Lemma 4.50 *If there exists r such that $L \subseteq A^*$ can be recognized by an r-program over some M in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ then there exists a constant k such that $D_k(L) = O(1)$ in the worst-case partition.*

Proof. Let ϕ be the r-program over M recognizing L. Since M lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$, we know that there exist t such that the program accepts x if and only if $\phi(x)$, when viewed as a word in M^*, belongs to some disjoint union of $\sim^G_{|M|,t}$

classes for some nilpotent group G of class d. In the case $r = 1$, we can use a variant of the protocol described in the proof of Lemma 4.29 to do this but if $r > 1$ then the value of a particular instruction may be unknown to as many as r players and we have to be more careful in our implementation of this strategy.

We choose a number of players $k = (|M|+t+1)\cdot r\cdot d+1$. Let us consider M as an alphabet and show that for any $w \in M^*$ and any $x \in A^*$, these k players can check in $O(1)$ communication if $w \sim^G_{|M|,t} \phi(x)$. We argue by induction on $s = |M| + t$. This is trivially true for $s = 1$. If $s > 1$, we distinguish two cases. First if $t = 0$, then the players need to verify that w and $\phi(x)$ are G-equivalent. Since G is nilpotent of class d, this can be done by counting, modulo some q, occurrences of subwords of length d in $\phi(x)$. Any such occurrence is the result of at most d instructions and its existence is thus known to all but $r \cdot d$ players, regardless of the input's partition. Since k is greater than $r \cdot d$, the counting of these occurrences modulo q can be done using only $O(1)$ communication.

If $t > 0$, the players can check in $O(1)$ communication that $w \sim^G_{|M|,t-1} \phi(x)$ (by induction) and that $\alpha(w) = \alpha(\phi(x))$ (because $k > r$). The difficulty, of course, lies in verifying that for any $m \in w$ if $w = w_0mw_1$ is the m-left decomposition of w, then the m-left decomposition of $\phi(x) = v_0mv_1$ is such that $w_0 \sim^G_{|M|-1,t} v_0$ and $w_1 \sim^G_{|M|,t-1} v_1$. For a given input x to the program and a given partition of this data, our k players can vote on the set of players which hold (on their forehead) one of the input letters queried by the first instruction in ϕ that, in their opinion, outputs a given $m \in \alpha(\phi(x))$. Note that this vote only requires each player to send $r \cdot \lceil \log k \rceil = O(1)$ bits.

The players that *do not* hold any of these r letters will, of course, all agree. Because $k > 2r + 1$, a majority of players (at least $k' = (|M| + t) \cdot r \cdot d + 1$ of them) will thus identify a subset of at most r players with these letters of x written on their forehead and will know that the m-left decomposition of $\phi(x)$ is v_0mv_1. These k' players need to verify $w_0 \sim^G_{|M|-1,t} v_0$ and $w_1 \sim^G_{|M|,t-1} v_1$ and this can be done, by our induction hypothesis, at $O(1)$ cost since u_0 and v_0 are also outputs of an M-program. Left-right symmetry completes the proof. $\square$

The technicality of this proof might be seen as a bad omen: Surely if the "easy" half or our conjecture is difficult to establish then we should expect that proving the second half, if it is true at all, will be extremely hard, if not out of reach. Such pessimism can be tempered: first our last proof is, at least conceptually, not so complicated. Secondly, our results of Chapter 3 show that $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ is reasonably well-behaved with respect to programs. In particular, this program-variety has the polynomial-length contraction property and this could be a useful tool. Furthermore, we have the advantage of knowing very good combinatorial descriptions of languages recognized (via morphisms) by this variety. It is worth noting that we know of no language with bounded k-party communication complexity which cannot be easily shown to be recognized by an r-program over $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$.

As a first step towards this conjecture, it might be easier to establish the conjecture in the restricted case where L has a neutral letter. This would follow if we could show, for instance, that r-programs over $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ have the Crane-Beach property. It is clear that progress towards this conjecture will require a very good understanding of the combinatorics of the multiparty model.

4.5.2 Further Bounds for Regular Languages

Another very intriguing open question concerns the non-deterministic communication complexity of regular languages. Attacking this question from an algebraic angle will require a refinement of our techniques since some regular languages have a non-deterministic communication complexity exponentially smaller than their complement. This means that we cannot find tight bounds for $N^1(L)$ by simply looking at the algebraic properties of $M(L)$. However, it is easy to show that the class of regular languages with non-deterministic communication complexity $O(f)$ is closed under positive Boolean operations, inverse homomorphisms and left-right quotients, i.e. that it forms a positive variety of languages (see e.g. [Pin97]). Correspondingly, $N^1(L)$ is thus determined by algebraic properties of L's *ordered* syntactic monoid. We have already established that, in the two-party case, a regular language has logarithmic deterministic communication complexity if and only if it is a disjoint union of unambiguous concatenations of languages with commutative syntactic monoids and that it has logarithmic MOD$_p$-counting complexity if it is a disjoint union of products with p-counters of languages with commutative syntactic monoids. It is tempting to conjecture the following characterization for the non-deterministic two-party model:

Conjecture 4.51 *Let $L \subseteq A^*$ be a regular language with $M = M(L)$. Then*

$$N^1(L) = \begin{cases} O(1) & \text{if and only if } M(L) \text{ is commutative;} \\ \Theta(\log n) & \text{if and only if } M(L) \text{ is non-commutative} \\ & \text{but } L \text{ is the disjoint union of languages} \\ & L_0 a_1 L_1 \dots a_k L_k \text{ with } M(L_i) \text{ commutative;} \\ \Theta(n) & \text{otherwise.} \end{cases}$$

Because Greater Than has linear non-deterministic two-party complexity, it is possible to show $N^1(L) = \Omega(\log n)$ whenever $M(L)$ is non-commutative. The non-deterministic upper bound for languages of the form $L_0 a_1 L_1 \dots a_k L_k$ with $M(L_i)$ commutative can be obtained in a way similar to the protocol given in the proof of Lemma 4.10: God first proposes factorization of the input w as $u_0 a_1 u_1 \dots a_k u_k$ by sending, at logarithmic cost, the positions of the k bookmarks and Alice and Bob then check the validity of this factorization by verifying, at constant cost, that $u_i \in L_i$ for each i. On the other hand, the linear lower bound probably requires both subtle algebraic calculations and non-deterministic communication complexity lower bounds that are of a different nature than the ones for Inner Product and Disjointness. A curious corollary of this conjecture would be that if L is regular then $D(L) = \Theta(\max\{N^1(L), N^0(L)\})$.

There are of course many questions left open in the multiparty case. We have already mentioned open problems about the k-party communication complexity of piecewise testable languages, the exact k-party complexity of B_2 and U among others. All our results focus on the case where the number of players is fixed independently of the input's length but it is of course natural to consider the case where k is a function of n. Currently, no non-trivial bound is known for the communication complexity of an explicit function when k is polylogarithmic in n. Such bounds would be extremely interesting since it has been observed [HG90] that, from the results of [Yao90, BT94], any $f \in ACC^0$ has polylogarithmic simultaneous multiparty communication complexity if the number of players is polylogarithmic.

Finally, it is possible that the algebraic approach will also be fruitful in the study of quantum communication complexity. This model introduced by A. Yao [Yao93] generalizes the probabilistic model by allowing Alice and Bob to exchange qubits. It has attracted considerable attention in the last ten years and it is still unclear how its power differs from the classical communication models [TS99]. It is known that Inner Product modulo 2 still has linear communication complexity in this model [CvDNT99] and it would be interesting to translate this to a linear lower bound for any regular language L with $M(L)$ in $\mathbf{DS} - \mathbf{DO}$ or outside $\overline{\mathbf{Ab}}$ using the methods developed in Section 4.2.

Chapter 5

Satisfiability of Equations over Semigroups

Algorithmic questions concerning the resolution of equations over finitely presented groups is a central concern in combinatorial group theory. For instance, a recent result of C. Gutiérrez shows that the problem of checking the satisfiability of an equation over the free group lies in PSPACE [Gut00]. One may also view the famous DISCRETE LOGARITHM problem as a simple group equation. In each of these contexts, the group is, at least implicitly, given as part of the input.

On the other hand, M. Goldmann and A. Russell studied in [GR99] the relationship between the algebraic properties of a finite group and the complexity of determining the solvability of an equation or a system of equations over that fixed group. They showed that determining whether a system of equations over G has a solution is NP-complete for any non-Abelian G and polynomial time computable for any Abelian G. For the case of a single equation, however, they could not establish a complete dichotomy: they proved on one hand the NP-hardness of determining the solvability of an equation over a fixed non-solvable group and, on the other hand, showed that the problem was polynomial-time computable, and in fact computable in ACC^0, for nilpotent groups. The case of solvable, non-nilpotent groups, however, was left open. Interestingly, the upper bound for nilpotent groups is obtained by studying the complexity of determining whether a given n-input G-program outputs a specific element g of G for some input $x \in \{0,1\}^n$.

In this chapter, we extend this work in a number of directions: we investigate the complexity of checking the satisfiability of programs, equations and systems of equations over monoids and, more generally, of semigroups. The results in the case of programs and single equations complete the picture sketched by Goldmann and Russell by further uncovering the tight relationship between these questions and the ones about the expressive power of programs and expressions over particular varieties of monoids. The case of systems of equations, on the other hand, is tightly connected to the constraint satisfaction problems, a well-studied framework used to analyze the complexity of a wide range of combinatorial problems.

We can also relate our work to that of O. Klíma and J. Srba about the complexity of Unification and Matching in idempotent semigroups [KS00, Klí02, Klí03a]. These problems are equivalent to testing the satisfiability of certain equations over a free idempotent semigroup satisfying some fixed set of identities. More recently, O. Klíma has considered the problem of solving certain systems of *two* equations over a fixed monoid [Klí03b] in order to understand the complexity of checking if a monoid satisfies a given identity.

The chapter consists of two independent parts. Results of Section 5.1 concern satisfiability problems for single equations and programs over monoids. They were, for the most part, published in [BMM+00] and obtained in collaboration with D. Barrington, P. McKenzie, C. Moore and D. Thérien. The results of Section 5.2 concern systems of equations over monoids and semigroups and arose from further work with C. Moore and D. Thérien [MTT01] and subsequent collaboration with Ondřej Klíma [KTT03]. I am particularly indebted to Ondřej for Lemma 5.26.

5.1 Single Equations and Programs

5.1.1 Introduction

Formally, an equation over a finite monoid M is given as:

$$c_0 X_{i_o} c_1 \dots c_{n-1} X_{i_n} = d_0 X_{j_o} d_1 \dots d_{m-1} X_{j_m}$$

where $c_i, d_j \in M$ are constants and the X_i's are variables, not necessarily distinct. The EQUATION SATISFIABILITY problem for M (which we will denote by EQN_M) is to determine whether a given M-equation has a solution i. e. to determine whether one can assign values in M to the variables such that the equation is satisfied. Similarly the TARGET-EQUATION SATISFIABILITY problem for M (denoted T-EQN_M) is the special case of EQN_M where the right-hand side of the equation is free of variables and thus consists of a single constant which we call the target (we will refer to these as equations *with targets*). Clearly, T-EQN_M and EQN_M are equivalent problems when M is a group.

We will be considering M-programs over a binary input alphabet. In this case, we write instructions in our programs as (i, m_0, m_1) with $i \in [n]$ and $m_0, m_1 \in M$. Such an instruction queries input bit x_i and outputs m_{x_i}. An instance of the PROGRAM SATISFIABILITY problem for M (denoted P-SAT_M) consists of an n-input M-program

$$\phi = (i_1, f_1)(i_2, f_2) \dots (i_s, f_s)$$

and a target element $m \in M$. The problem is to determine whether there exists some $x \in \{0,1\}^n$ such that $\phi(x) = m$. Note that this is always at least as hard as determining the satisfiability of an equation with target.

Lemma 5.1 *For any M, we have*

$$\text{T-EQN}_M \leq_P \text{P-SAT}_M$$

and

$$\text{EQN}_M \leq_P^{btt} \text{P-SAT}_{M \times M}.$$

Proof. Suppose the equation $c_0X_{i_1}c_1 \dots X_{i_n}c_n = s$ has t variables. It can be satisfied if and only if the following M-program over the $t \cdot |M|$ variables $Y_1, \dots, Y_{t \cdot |M|}$ can reach the target s: we replace every constant c_i by the instruction (Y_1, c_i, c_i) and each occurrence of the variable X_i by a sequence of $|M|$ instructions querying variables $Y_{k_1}, \dots, Y_{k_{|M|}}$ of the form

$$(Y_{k_1}, 1_M, m_1)(Y_{k_2}, 1_M, m_2) \dots (Y_{k_{|M|}}, 1_M, m_{|M|})$$

where $m_1, \dots, m_{|M|}$ are the $|M|$ elements of M. Note that for any $m \in M$, there is an assignment of the Y_k's such that this sequence of $|M|$ instructions evaluates to m.

A similar construction allows the encoding of a system of two *target*-equations

$$\begin{cases} c_0X_{i_1}c_1 \dots X_{i_n}c_n = s \\ d_0X_{j_1}c_1 \dots X_{j_p}c_p = t \end{cases}$$

as a program over $M \times M$. The first half of the program encodes the first equation as in the above paragraph by using only the first copy of M: constants c_i are now replaced by $(Y_1, (c_i, 1_M), (c_i, 1_M))$ and variables become blocks of instructions of the form

$$(Y_{k_1}, (1_M, 1_M), (m_1, 1_M)) \dots (Y_{k_{|M|}}, (1_M, 1_M), (m_{|M|}, 1_M)).$$

Similarly the second half of the program uses the second copy of M. The crucial observation however is that if a variable X occurs in both equations then for any setting of the variables Y, the program segments corresponding to an occurrence of X in the first half will evaluate to $(1_M, m)$ if and only if the segments corresponding to X in the second half evaluate to $(m, 1_M)$. The system of equations is thus satisfiable if and only if the program can reach the target (s, t). To explicitly complete the reduction from EQN_M to P-SAT$_{M \times M}$, it suffices to note that $E_1 = E_2$ is satisfiable in M if and only if there is $m \in M$ such that the system

$$\begin{cases} E_1 = m \\ E_2 = m \end{cases}$$

is satisfiable. □

Despite the apparent similarity of T-EQN and P-SAT, we will see that the converse of this lemma is not true unless P equals NP and we will try to understand how and why T-EQN$_M$ and P-SAT$_M$ differ in complexity. It should be noted that all three problems defined above lie within NP since it is easy to check in polynomial-time, and in fact in NC^1, whether a particular assignment satisfies a given equation or program.

As a start, it is useful to understand whether upper bounds for program or equation satisfiability over a certain M can translate into upper bounds for satisfiability problems over divisors of M or $M \times M$.

Lemma 5.2 1. *If N is a submonoid of M then* P-SAT$_N \leq_P$ P-SAT$_M$ *and if N is a morphic image of M then* P-SAT$_N \leq_P^{btt}$ P-SAT$_M$.

2. *If N is a morphic image of M then* T-EQN$_N \leq_P^{btt}$ T-EQN$_M$.

3. *For any M, N, we have*

$$\mathrm{EQN}_{M\times N} \leq_P^T \{\mathrm{EQN}_M, \mathrm{EQN}_N\}$$

and

$$\text{T-EQN}_{M\times N} \leq_P^T \{\text{T-EQN}_M, \text{T-EQN}_N\}.$$

Proof.

1. A program ϕ over a submonoid N of M is simply a program over M where the output of each instruction lies in N and so any algorithm for P-SAT$_M$ *is* an algorithm for P-SAT$_N$.

   Suppose now that there is a surjective morphism $\psi : M \to N$. For a given an n-input N-program ϕ, we can obtain a (not uniquely defined) M-program ϕ' by replacing the elements of N output by instructions of ϕ

by arbitrary pre-images of them under ψ. Thus, for each $x \in \{0,1\}^n$, we have $\psi(\phi'(x)) = \phi(x)$ and so there exists x with $\phi(x) = t$ if and only if for some $m \in M$ where $\phi(m) = t$ there exists x with $\phi'(x) = m$.

2. A similar argument can be used: if $\psi(M) = N$, then given an N-expression E, we can obtain an M-expression E' by replacing every constant in E by some arbitrary pre-image. Then $E = t$ is satisfiable if and only if there is a pre-image $m \in M$ of t such that $E' = m$ is satisfiable.

3. This last part simply follows from the observation that an equation over the direct product $M \times N$ is simply a pair of completely independent equations over M and N respectively.

□

Consequently, the class $\mathcal{M}_P = \{M : \text{P-SAT}_M \in P\}$ is closed under division, $\mathcal{M}_{TE} = \{M : \text{T-EQN}_M \in P\}$ is closed under direct product and morphic images and $\mathcal{M}_E = \{M : \text{EQN}_M \in P\}$ is closed under direct product. As we will see, the latter two classes do not form varieties. Intuitively, $\mathcal{M}_E$ might not be closed under submonoids because when we can check satisfiability of equations over the larger monoid M no mechanism can guarantee that the variables are assigned only values in the submonoid. This motivates the following definition:

Definition 5.3 *A subset T of M is said to be* inducible *if there exists an M-expression E in k variables such that the image of E (that is $\{m$: for some $\bar{x} \in M^k, E(\bar{x}) = m\}$) is T.*

The following is an easy observation due to [GR99]:

Lemma 5.4 *If N is an inducible submonoid of M, then* $\text{EQN}_N \leq_P \text{EQN}_M$ *and* $\text{T-EQN}_N \leq_P \text{T-EQN}_M$.

On the other hand, $\mathcal{M}_P$ is closed under division but might not be closed under direct product. We can certainly view a program over $M \times N$ as a pair of programs on M and N respectively which are both satisfiable if the original program is, but, conversely, there is no obvious way to check whether the sets of satisfying assignments for each of them are disjoint or not. We will come back to these issues in Subsection 5.1.4.

5.1.2 Groups

As we stated earlier, it is shown in [GR99] that T-EQN$_G$ (and thus EQN$_G$) are NP-complete for any non-solvable group G. As an immediate corollary, we also obtain the NP-completeness of P-SAT$_G$ for non-solvable G. The latter is not surprising in light of Barrington's Theorem: the satisfiability problem for programs over a non-solvable group is equivalent to the satisfiability problem for NC^1 circuits, which is NP-complete. In fact, the problem is already NP-complete for depth two AC^0-circuits (by Cook's Theorem). Similarly, other results about the complexity of P-SAT for restricted classes of monoids can be interpreted as results about the complexity of checking satisfiability for circuits of a corresponding class.

Goldmann and Russell also showed that for a nilpotent group G, P-SAT$_G$ (and thus T-EQN$_G$) was computable in polynomial time. Their proof is centered around the following fact:

Proposition 5.5 *[PT88] Let ϕ be an n-input program over a nilpotent group G and g an arbitrary element of G. There exists a constant d_G such that if there exists some $x \in \{0,1\}^n$ where $\phi(x) = g$ then there exists a $y \in \{0,1\}^n$ of weight at most d_G such that $\phi(y) = g$.*

In other words, if a P-SAT$_G$ instance is satisfiable, then it can be satisfied by one of the $\sum_{i=1}^{d_G} \binom{n}{d_G} = O(n^{d_G})$ n-bit strings of weight at most d_G. Since this set has polynomial size a "brute force search" approach can be used to check satisfiability in polynomial time.

This proposition also shows that a program over a nilpotent group G cannot compute the AND of more than d_G variables. We mentioned in the introduction and in Chapter 3 that the AND-function is not believed to be computable even by a program of sub-exponential length over any solvable group although this conjecture has only been proved for a small subvariety of solvable groups (see Chapter 3 and Section 4.4).

Definition 5.6 *A finite group G is AND-*strong *if there exists a G-program of polynomial length computing* AND *and is AND-*weak *if any G-program computing the* AND *of n variables requires length $\Omega(c^n)$ for some $c > 1$.*

From Barrington's Theorem we know that all non-solvable groups are AND-strong while the results of [BST90] show that the wreath product of a p-group and an Abelian group is always AND-weak. The following shows that, similarly to the nilpotent case, the lack of expressiveness of AND-weak groups can be exploited to obtain good algorithms for P-SAT.

Theorem 5.7 *If G is AND-weak then* P-SAT$_G$ *is solvable in quasi-polynomial time.*

Proof. We claim that if a program in s variables over G can be satisfied, then it can be satisfied by an assignment of weight logarithmic in the length of the program (Recall that the weight $|x|_1$ of $x \in \{0,1\}^*$ is the number of 1's in x). Suppose that this is not the case. Let w be a satisfying assignment of minimal weight, with $|w|_1 = k$. Assume without loss of generality that the first k bits of w are 1. By fixing $X_{k+1}, \ldots, X_s$ to 0, we obtain a k-input program ψ over G. This program outputs s when all its input bits are 1, but since w was assumed to have minimal weight, the output of ψ is not s otherwise. In other words, ψ is computing the AND of k bits. Since G is AND-weak, we must have $n \geq 2^{\Omega(k)}$, so $k \leq O(\log n)$. It is thus sufficient to consider only the $O(\binom{n}{O(\log n)}) = O(n^{O(\log n)})$ assignments of weight at most k, so we have a quasi-polynomial time algorithm. □

Many solvable groups, however are not known to be AND-weak, so it would be preferable to obtain upper bounds on the complexity of P-SAT for solvable groups independently of assumptions on their computational power but the following theorem shows that the two questions are probably too closely tied to allow it.

Theorem 5.8 *If G is AND-strong, then* P-SAT$_{G \circ C_k}$ *is NP-complete for the wreath product $G \circ C_k$ for any cyclic group C_k with $k \geq 4$.*

Proof. We want to build a reduction from 3-SAT. Define the function $f_{g_0,g_1} : C_k \to G$ as

$$f_{g_0,g_1}(x) = \begin{cases} g_0 & \text{if } x = 0 \\ g_1 & \text{if } x \neq 0 \end{cases}$$

Also, denote by id the function such that $id(x) = 1_G$ for all $x \in C_k$. Consider now the following 3-input program over $G \circ C_k$

$$\begin{aligned} \phi \;=\; & (1, (id, 0), (id, 1))\ (2, (id, 0), (id, 1))\ (3, (id, 0), (id, 1)) \\ & (\mathbf{1}, (\mathbf{f_{g_0,g_1}}, \mathbf{0}), (\mathbf{f_{g_0,g_1}}, \mathbf{0}))(1, (id, 0), (id, -1)) \\ & (2, (id, 0), (id, -1))\ (3, (id, 0), (id, -1)) \end{aligned}$$

First note that the C_k component of ϕ's output will always be 0. Note also that the middle instruction's output is independent of the value of the bit queried. It is also the only instruction affecting the G^{C_k} component of the output. This component is a function f such that $f(0) = g_1$ if one of the input bits is on and $f(0) = g_0$ otherwise. To see this note that when we execute the middle instruction, the product in C_k so far is not equal to zero if and only if one of the instructions yielded a $+1$. Thus, ϕ is recognizing the OR of these three variables.

Suppose the 3-SAT instance has m clauses. By assumption there is a G-program of length m^c that computes the AND of m variables. If we replace every instruction (i, g_0, g_1) by a program over $G \circ C_k$ as above, we obtain a program of length $7 \cdot m^c$ which is satisfiable if and only if the 3-SAT instance is satisfiable. $\square$

The wreath product of two solvable group is itself solvable and the proof of Theorem 5.7 can be used to show that super-polynomial lower bounds on the length of programs recognizing the AND over a group G translate into sub-exponential upper bounds on the time complexity of P-SAT$_G$. Thus, assuming that no sub-exponential time algorithm can solve an NP-hard problem, there exists an AND-strong solvable group if and only if there exists a solvable group for which P-SAT is NP-complete.

By Lemma 5.1 we now have upper bounds for T-EQN$_G$ for AND-weak G's but it is not known whether the hardness result on P-SAT$_{G \circ C_k}$ for AND-strong G's can translate into hardness for, say, T-EQN$_{G \circ C_k}$. On the other hand, if all solvable groups are AND-weak, as we conjecture, then this is unimportant.

5.1.3 Aperiodic Monoids

In light of the group case, one might hope to prove a converse to Lemma 5.1 and show equation satisfiability and program satisfiability to be polynomially equivalent but we show in this section that there exist aperiodic monoids M such that P-SAT$_M$ is NP-complete but EQN$_M$ is in P. Furthermore, we characterize the class of aperiodics for which P-SAT is NP-complete.

Lemma 5.9 *For any monoid M in* **DA**, P-SAT$_M$, T-EQN$_M$ *and* EQN$_M$ *all lie in P (in fact in AC^0).*

Proof. By Lemma 5.1, it suffices to prove the upper bound for P-SAT. Let ϕ be an n-input M-program of length l and $F \subseteq M$ be a target set. The set $\{w : w \in M^* \text{ and } eval_M(w) = m \in F\}$ can be expressed as the finite disjoint union of unambiguous $A_0^* a_1 A_1^* \dots a_k A_k^*$ with $a_i \in M$ and $A_i \subseteq M$.

Hence it is sufficient to consider the at most $\binom{l}{k}$ k-tuples of instructions of ϕ that could be held responsible for the presence of the subword $a_1 a_2 \dots a_k$ in $\phi(x)$. For each of them, we need to check if there is an assignment such that the output of the program belongs to $A_0^* a_1 A_1^* \dots a_k A_k^*$ and that can clearly be done in linear time.

In fact, this brute force approach can easily be implemented in AC^0 since the evaluation of the product of n elements in an aperiodic monoid is computable in AC^0 [BT88]. □

If we turn our attention to aperiodics outside of **DA**, the first examples to consider are of course U and B_2.

Lemma 5.10 T-EQN$_{B_2}$ *is NP-complete.*

Proof. We use a reduction from 1-3SAT. Each variable v_i in the 1-3SAT instance is represented by two variables v_i^+ and v_i^- representing v_i and its complement in the equation. We build the following equation with target ab. First, we concatenate, for each i the segments $abv_i^+v_i^-bv_i^-v_i^+b$ and for each clause e.g. $(v_i, \bar{v}_j, v_k)$ we concatenate $abv_i^+v_j^-v_k^+b$.

It is easy to see that the first half of the equation forces us to choose one of v_i^+, v_i^- as 1 and the other as a. If we now interpret a as TRUE and 1 as FALSE, the equation is satisfiable if and only if we can choose assignments to the v_i such that in every segment e.g. $abv_i^+v_j^-v_k^+b$ exactly one of the variables is set to a. □

Thus, P-SAT$_{B_2}$ is also NP-complete. Furthermore:

Lemma 5.11 P-SAT$_U$ *is NP-complete.*

Proof. We essentially use the universality of U discussed in Chapter 3. We have already seen that the program $\phi_i = (i_1, b, ba)(i_2, 1, a)(i_3, 1, a) \ldots (i_k, ba, a)$ (over U) outputs ba if one of the X_{i_j}'s is set to 1 and 0 otherwise. By concatenating such ϕ_i's we get a program whose output is ba if all ϕ_i's have one of their input variables set to 1 and 0 otherwise. So we can simulate a CNF formula and obtain a reduction from SAT. □

Using Lemma 5.2, we thus obtain the following dichotomy:

Theorem 5.12 *If M is aperiodic then* P-SAT$_M$ *lies in P if M is in* **DA** *and is NP-complete otherwise.*

It is tempting in light of Lemma 5.10 to conjecture that the same dichotomy also holds for equation and target-equation satisfiability. This is however not the case:

Theorem 5.13 EQN$_U$ *and* T-EQN$_U$ *can be decided in polynomial time.*

Proof. We first provide a polynomial time algorithm for T-EQN$_U$ and crucially use the fact that, in U, $axa = a$ whenever $x \neq 0$. Intuitively, we use the fact that a's are our friends. In particular, we have that if $xyz = a$ then $xaz = a$.

We will show that for any target, if the equation is satisfiable then it can be satisfied by an assignment with a very precise structure. We are given the expression $E : c_0 X_{i_1} c_1 \ldots X_{i_n} c_n$ and a target m.

If $m = 0$, the equation is trivially satisfiable by setting any variable to 0, and if $m = 1$, it is satisfiable if and only if all the c_i's are 1. Since the equation is 0 if any of the c_i is 0, we will assume that the constants are non-zero.

If $m = a$, then E is satisfiable if and only if it is satisfied when all the variables are set to a, namely when we have both $c_0 \in \{1, a, ab\}$ and $c_n \in \{1, a, ba\}$.

If $m = ba$, and E can be satisfied, then it can be satisfied by one of the $O(n)$ assignments of the following form: all the variables occurring before some point j in the equation (which might be a constant or a variable) are set to 1, the variable at point j is set to ba and the other variables are set to a. To see this, consider any satisfying assignment to E. If the first b in the induced string comes from a constant, then all the variables occurring before it must have been set to 1. Moreover, all we have to insure now is that there are no consecutive b's in the suffix. So we can set all the variables that haven't yet occurred to a without affecting the target. If the first b occurs in a variable, the same reasoning shows that we can set this variable to ba and the variables not yet considered to a. So it is sufficient to check a linear number of possible assignments to decide satisfiability. The case $m = ab$ is handled in a similar, symmetrical way.

Finally if $m = b$, it suffices to consider the following $O(n^2)$ assignments: variables occurring before some j or after some k are set to the identity, the variable at point j is set to ba or b, the one at point k to ab or b, and all remaining variables are set to a. Again, if we now consider any satisfying assignment and call j and k the first and last occurrence of b in the induced word over M^*, then we know that all variables occurring before j or after k were set to 1. The variable or constant at point j must be b or ba, the one at point k being b or ab so we still have a satisfying assignment if we set the rest of the variables to a.

The algorithm can easily be adapted to handle equations with variables on both sides. □

Since B_2 belongs to the variety generated by U this shows that neither $\mathcal{M}_E$ nor $\mathcal{M}_{TE}$ form varieties. This also shows that B_2 is not inducible[1] as a submonoid of $U \times U$. In [GR99], the notion of inducibility was needed to complete the NP-completeness proof of T-EQN$_G$ in the case of non-solvable groups G and our result gives indication that this was a necessary evil.

[1] Strictly speaking, this is under the hypothesis that P is not NP, although this can probably be proved directly.

5.1.4 A Look at the General Case

Getting necessary and sufficient conditions for the tractability of program and equation satisfiability already seems difficult, if not impossible, in the basic cases of groups and aperiodics. In this subsection we prove some partial results for monoids in general and try to further understand what differentiates program satisfiability and equation satisfiability.

We start with an easy generalization of the hardness results for non-solvable groups.

Theorem 5.14 *If M is non-solvable then* P-SAT$_M$*,* EQN$_M$ *and* T-EQN$_M$ *are all NP-complete.*

Proof. The result for P-SAT is an immediate corollary of the NP-completeness of P-SAT$_G$ for non-solvable groups G and Lemma 5.2. Similarly, if G is a non-solvable subgroup of M with idempotent e then the expression exe induces the submonoid eMe. Any target-equation over G, say

$$c_0 X_{i_1} c_1 \dots X_{i_n} c_n = g$$

can be viewed as an equation over eMe. If it is satisfiable in eMe then a satisfying assignment must set variables on the left-hand side to values $\mathcal{J}$-above the target g and thus $\mathcal{H}$-related to e. In other words, the equation is satisfiable over eMe if and only if it is satisfiable in G. Hence we have T-EQN$_G \leq_P$ T-EQN$_{eMe} \leq_P$ T-EQN$_M \leq_P$ EQN$_M$ and so the latter two are NP-complete. $\Box$

The proof of Lemma 5.9 used crucially the combinatorial characterization of languages with syntactic monoids in **DA** and it it perhaps not too surprising that this argument can be generalized to subclasses of **DO**.

Lemma 5.15 *If $M \in \mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ then* P-SAT$_M$*,* EQN$_M$ *and* T-EQN$_M$ *all lie in P (and in fact in ACC^0).*

Proof. Because of Lemma 5.1 it is again sufficient to establish the upper bound for P-SAT$_M$. Let ϕ be an n-input M-program of length l and $F \subseteq M$ be a target set. The set $\{w : w \in M^*$ and $eval_M(w) = m \in F\}$ can be expressed as the finite disjoint union of unambiguous $L_0^* a_1 L_1^* \dots a_k L_k^*$ where the L_i are the intersection of some A_i^* with $A_i \subseteq M$ and some language K_i recognized by a nilpotent group.

For the at most $\binom{l}{k}$ k-tuples of instructions of ϕ that could be held responsible for the presence of the subword $a_1 a_2 \dots a_k$ in $\phi(x)$, we need to check if there is an assignment such that the output of the program belongs to $L_0^* a_1 L_1^* \dots a_k L_k^*$. As a first step we check for each input variable x_j whether setting it to 0 or 1 causes some instruction to throw us out of one of the A_i^*'s. This process forces an assignment on some of the variables (or possibly even proves that the target is unreachable given this particular k-tuple) and leaves other free.

What we are left with can be thought of as a system of $k+1$ programs over nilpotent groups. The k-tuple chosen previously naturally defines k segments $\phi_0, \phi_1, \dots, \phi_k$ of the program which, after our initial computation, are now n'-input programs for some $n' \leq n$. We are searching of course for some $x' \in \{0,1\}^{n'}$ such that for all $1 \leq j \leq k$ we have $\phi_i(x')$ belongs to the nilpotent group language K_i. We can reuse a trick of the proof of Lemma 5.1 to argue that this is no harder than a program satisfiability problem over the nilpotent group $M(K_0) \times M(K_1) \times \dots \times M(K_k)$ which, by the result of [GR99] is doable in polynomial time. $\square$

Indications of hardness for P-SAT$_M$ when M lies outside $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ are scarce. Just as we have shown NP-completeness for non-solvable M's, we can extend our results about aperiodics easily to show:

Lemma 5.16 *If M is not in* $\mathbf{DS}$ *then* P-SAT$_M$ *is NP-complete.*

We are however unable so far to provide any indication of hardness for either P-SAT$_M$ or EQN$_M$ when M is in $\mathbf{DS} - \mathbf{DO}$.

P is closed?	Direct products	Factors	Subs
$\mathcal{M}_{TE} \cap \mathbf{G}$	YES	YES	?
$\mathcal{M}_{TE} \cap \mathbf{A}$	YES	YES	NO
$\mathcal{M}_{TE}$	YES	YES	NO
$\mathcal{M}_{P} \cap \mathbf{G}$	?	YES	YES
$\mathcal{M}_{P} \cap \mathbf{A}$	YES	YES	YES
$\mathcal{M}_{P}$	?	YES	YES

Table 5.1: Closure properties of $\mathcal{M}_{TE}$ and $\mathcal{M}_P$.

Our results so far have also indicated that beyond their apparent similarities, the tasks of checking satisfiability for programs and for equations present different computational challenges. In particular, Table 5.1 sums up the known closure properties of the classes $\mathcal{M}_{TE} = \{M : \text{T-EQN}_M \in P\}$ and $\mathcal{M}_P = \{M : \text{P-SAT}_M \in P\}$ and they are quite different. It should be noted that although $\mathcal{M}_P \cap \mathbf{A}$ is known to be closed under direct products we have no direct proof of this fact.

5.1.5 Open Problems

Our results establish a close connection between the algebraic properties of a finite monoid M, its power as a language recognizer and the complexity of resolving equations or programs over M. Many questions remain open, however, the most important of which is the complexity of P-SAT in the case of solvable but not nilpotent groups. Of course, we have tied the full resolution of this question to lower bounds for CC^0 circuits but progress on this problem can be made in other ways. For instance, we cannot rule out that there exists a better way to use the hypothesis that a group is AND-weak in order to surpass our quasi-polynomial time upper bound P-SAT$_G$. We conjecture that this is not possible. Certainly, any indication that P-SAT$_G$ is not in P for non-nilpotent AND-weak groups would be of great interest. Note that our current algorithm puts, for instance, P-SAT$_{S_3}$ in the complexity class $NP^{[log^2 n]}$ of problems which can be decided by a polynomial time non-deterministic Turing machine that is using only $O(\log^2 n)$ bits of non-determinism (see e.g. [DF97, GLM96]). Perhaps, P-SAT$_{S_3}$ can actually be shown complete for this class or at least hard for some smaller class of bounded non-determinism.

Another outstanding problem is the complexity of P-SAT$_M$ when M is in $\mathbf{DS} - \mathbf{DO}$. We believe that the problem is NP-complete also in this case. In light of our proof that monoids in $\mathbf{DS} - \mathbf{DO}$ are universal (Lemma 3.13), this would follow from the NP-completeness of determining if, given a CNF formula, there exists a truth assignment to the variables which satisfies 0 modulo p clauses in the formula. To the best of our knowledge, no hardness result is known for this problem. Obtaining indication that P-SAT$_M$ is tractable if and only if M lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ would provide an interesting parallel with our communication complexity results.

We chose to study the satisfiability of programs. It would be reasonable to also study the equivalent of P-SAT for k-programs. The results would be radically different in the case of aperiodics since it is easy to prove this problem is NP-complete for any aperiodic if $k \geq 2$. On the other hand, our upper bounds for nilpotent groups still hold and if a group is AND-strong then the problem is NP-complete for $k \geq 3$.

Finally, one can study the complexity of counting the number of solutions for a program-equation. This was briefly discussed in [BMM$^+$00], where it is established that the problem #P-SAT is in $\#L$ for monoids in $\mathbf{DA}$ but $\#P$-complete for non-solvable monoids and for U and BA_2, in simple correspondence with the results presented above. In particular, it seems challenging to find an efficient way of counting the number of assignments satisfying a certain program over a nilpotent group G: our current algorithm for P-SAT$_G$ seems to lack the finesse presumably needed for the counting task.

5.2 Systems of Equations

We now turn to the study of systems of equations over a given semigroup. The Systems of Equations Satisfiability problem over a semigroup S (abbreviated as EQN^*_S) is that of determining whether a given set of equations over S can be simultaneously satisfied. We will also study the restriction T-EQN^*_S of EQN^*_S in which the right-hand side of each equation in the system is a constant. For finite groups, the problems are obviously equivalent: they lie in P for Abelian groups and are NP-complete otherwise [GR99].

Recall from Section 5.1 that $T \subseteq S$ is inducible if there exists an expression E over S such that the range of E is exactly T. Similar to Lemma 5.4, one can show

Lemma 5.17 *If T is an inducible subsemigroup S then* $\mathrm{EQN}^*_T \leq \mathrm{EQN}^*_S$ *and* $\text{T-EQN}^*_T \leq \text{T-EQN}^*_S$.

We will show that for any finite monoid M both EQN^*_M and T-EQN^*_M are either in P or NP-complete depending on whether M belongs to $\mathbf{J_1} \vee \mathbf{Ab}$ or not in the general case and depending on whether M belongs to $\mathbf{RB} \vee \mathbf{Ab}$ or not in the case of target-equations. We prove a similar dichotomy for T-EQN^*_S when S is a regular semigroup and prove a number of sufficient conditions on a semigroup S for the NP-completeness of T-EQN^*_S and EQN^*_S. We begin by pointing out a very interesting connection between these problems and so called constraint satisfaction problems.

5.2.1 Constraint Satisfaction Problems

Let D be a finite domain and Γ be a finite set of relations on D. To each pair D, Γ corresponds a Constraint Satisfaction Problem (CSP). An instance of CSP(Γ) is a list of constraints, i.e. of pairs $R_i(S_i)$ where $R_i \in \Gamma$ is a k-ary relation and S_i, the scope of R_i, is an ordered list of of k-variables (with possible repetitions) and we want to determine whether the variables can be assigned values in D such that each constraint is satisfied. As an example, the problem EQN^*_S can be seen as a CSP problem in which the domain is the semigroup S and Γ is the set of constraints definable as equations over S.

This class of combinatorial decision problems has received a lot of attention because of the wide variety of problems which it encompasses and because constraint satisfaction problems arise so naturally in artificial intelligence. CSP lies in NP and is easily seen to be NP-complete in general so one seeks to identify tractable restrictions of the problem. One might choose, for instance, to impose certain conditions on the structure of constraints appearing in a given instance. A lot of research has also dealt with identifying necessary and sufficient conditions on Γ to have CSP(Γ) tractable over a given domain D. This approach was pioneered by T. Schaefer [Sch78] who studied the CSP problem on Boolean domains. In this case, the problem is usually known as GENERALIZED SATISFIABILITY and Schaefer proved that this problem was NP-complete unless it was one of six tractable special cases: 2-SAT, 0-valid SAT, 1-valid SAT, affine-SAT, Horn-SAT and anti-Horn SAT. Affine-SAT is the case where each relation is the solution set of a system of equations over the cyclic group C_2. The only other 2-element monoid is U_1 of course and, interestingly, we can relate the last two of Schaefer's tractable cases to systems of equations over U_1.

Lemma 5.18 *A boolean relation is Horn or anti-Horn, i.e. expressible as tuples satisfying a conjunction of disjuncts containing each at most one un-negated (resp. negated) variable, if and only if it is the set of solutions of a system of equations over* U_1.

Proof. Identify the element 1 of U_1 with TRUE and 0 with FALSE. Then the Horn clause $X_1 \wedge X_2 \wedge \ldots X_n \to Y$ is satisfied when one of the X_i's is FALSE or when all X_i's and Y are TRUE. These are exactly the tuples which satisfy the equation

$$X_1X_2 \dots X_n = X_1 \dots X_n Y$$

over U_1.

Conversely, the equation $X_1 \dots X_n = Y_1 \dots Y_m$ corresponds to the Horn formula:

$$\bigwedge_{1 \le i \le m} (X_1 \wedge \dots \wedge X_n \to Y_i) \wedge \bigwedge_{1 \le i \le n} (Y_1 \wedge \dots \wedge Y_n \to X_i)$$

If on the other hand we choose to identify 1 with FALSE and 0 with TRUE, a similar argument shows the relationship of U_1 systems to anti-Horn formulas. □

Recently, tools from universal algebra [BKJ00, Dal00], group theory and relational database theory [FV99] have been used to identify classes of relations for which CSP is tractable and it is conjectured that for any domain D and any set of relations Γ the problem CSP(Γ) either lies in P or is NP-complete. Let us define a k-ary operation to be any function $f : D^k \to D$ and say that a relation $R \in D^t$ is *preserved* by f if for any k t-tuples

$$(d_1^1, d_2^1, \dots, d_t^1), \dots, (d_1^k, d_2^k, \dots, d_t^k)$$

all lying in R, the t-tuple

$$(f(d_1^1, \dots, d_1^k), \dots, f(d_t^1, \dots, d_t^k))$$

is also in R. The algebraic properties of the operations that preserve every relation in Γ can be studied to determine the complexity of CSP(Γ). Using this approach, A. Bulatov obtained a spectacular dichotomy theorem similar to the one of Schaefer for domains of size three [Bul02]. It has also been shown that if the domain is a semigroup S and Γ_S is the set of relations preserved by the multiplication in S then CSP(Γ_S) is tractable if S is a block group and is NP-complete otherwise [BJV02].

Although our work is a priori incomparable to the results just cited, the mechanics of some of our upper bounds can be rephrased in the universal algebra terminology. We will also use a powerful result of [FV99]:

Theorem 5.19 *If G is a group and Γ is a set of relations such that for each $R \in \Gamma$ of arity k the k-tuples in R form a coset of G^k, then over the domain G CSP(Γ) can be solved in polynomial time.*

5.2.2 Tractable Cases

We begin by presenting some polynomial time algorithms to test the satisfiability of systems of equations over simple classes of monoids and semigroups. We are faced with an inconvenient obstacle: if M and N are monoids such that M divides N we do not know how to infer upper bounds for EQN^*_M from upper bounds for EQN^*_N. In fact, as we will see later on, solving equations over certain *semigroups* might be easier than solving equations over some of their divisors. As usual, we will first separately treat the group case and the aperiodic case before combining them to get upper bounds in the general case. We first recall:

Lemma 5.20 ([GR99]) *If G is Abelian, then* $\mathrm{T\text{-}EQN}^*_G$ *and* EQN^*_G *are solvable in polynomial time.*

Remark 5.20. The proof of Goldmann and Russell uses simple Gaussian elimination techniques, but this lemma can also be obtained as a corollary to Theorem 5.19. Indeed, it can be shown that a subset T of a group G is a coset if and only if $uv^{-1}w$ lies in T for all $u, v, w \in T$. If an equation over a commutative group $x_{i_1} \dots x_{i_s} = g$ has solutions $(u_1, \dots, u_n)$, $(v_1, \dots, v_n)$ and $(w_1, \dots, w_n)$ then

$$\begin{aligned} u_{i_1} v_{i_1}^{-1} w_{i_1} \dots u_{i_s} v_{i_s}^{-1} w_{i_s} &= u_{i_1} \dots u_{i_s} v_{i_s}^{-1} \dots v_{i_1}^{-1} w_{i_1} \dots w_{i_s} \\ &= g g^{-1} g \\ &= g \end{aligned}$$

and so the solutions form a coset of G^n. In fact, this is true even of more complicated relations: if H is a subgroup of G and $\phi_1 \dots \phi_s$ are morphisms from G into H, then the relation defined by $\phi_1(x_{i_1}) \dots \phi_s(x_{i_s}) = h$ also forms a coset in G^n. This observation will be important in two of our algorithms.

Next, we consider systems over idempotent and commutative semigroups.

Lemma 5.21 *If S is an idempotent and commutative semigroup then* EQN^*_S *lies in P.*

Proof. Such semigroups are $\mathcal{J}$-trivial and the $\mathcal{J}$-ordering defines a semilattice on S. Our algorithm will rely on the following observation: if $(u_1, \dots, u_n)$ and $(v_1, \dots, v_n)$ are solutions to a system of equations $\mathcal{E}$ in n variables over S, then $(u_1v_1, \dots, u_nv_n)$ is also a solution to $\mathcal{E}$. Indeed, using idempotency and commutativity, any equation in $\mathcal{E}$ can be rewritten as

$$cx_{i_1} \dots x_{i_k} = dx_{j_1} \dots x_{j_t}$$

whence we can conclude $cu_{i_1} \dots u_{i_k} = du_{j_1} \dots u_{j_t}$ and $cv_{i_1} \dots v_{i_k} = dv_{j_1} \dots v_{j_t}$. Using idempotency and commutativity again, we thus get

$$cu_{i_1}v_{i_1} \dots u_{i_k}v_{i_k} = du_{j_1}v_{j_1} \dots u_{j_t}v_{j_t}.$$

Note that $(u_1v_1, \dots, u_nv_n)$ is the meet of $(u_1, \dots, u_n)$ and $(v_1, \dots, v_n)$ in the semilattice S^n.

Our algorithm maintains a lower bound $\overline{Y} = (y_1, \dots, y_n)$ for the minimal solution to $\mathcal{E}$. We initialize $\overline{Y}$ as $(0, \dots, 0)$ and update it as follows. In each step, if $(y_1, \dots, y_n)$ is a solution to $\mathcal{E}$, the algorithm halts. If not, there must be some equation[2] in $\mathcal{E}$, say $cx_{i_1} \dots x_{i_k} = dx_{j_1} \dots x_{j_t}$, such that $cy_{i_1} \dots y_{i_k} \neq dy_{j_1} \dots y_{j_t}$.

[2]Technically, since S is not necessarily a monoid, we cannot assume that constants c and d appear in this equation. This is however unimportant in our argumentation.

Since we are maintaining $\overline{Y}$ as a lower bound to any assignment satisfying $\mathcal{E}$, we know that, for any satisfying assignment, the right-hand side product is bounded below by $dy_{j_1} \ldots y_{j_t}$. Thus, if there is some y_{i_s} occurring on the left-hand side which is not $\mathcal{J}$-above $dy_{j_1} \ldots y_{j_t}$ then we can update our lower bound by setting $y_{i_s} := y_{i_s} \vee (dy_{j_1} \ldots y_{j_t})$, i.e. the $\mathcal{J}$-minimal element of S lying above both y_{i_s} and the right-hand side product[3]. We do similar updates on variables occurring on the left-hand side which do not lie $\mathcal{J}$-above the current product on the right-hand side.

We iterate this until we reach a fixed point for $\overline{Y}$. The process terminates in at most $n \cdot |S|$ steps since the value of $\overline{Y}$ always increases in the semilattice S^n. If the fixed point is not a solution to the system, then we conclude that $\mathcal{E}$ is unsatisfiable for in this case we must have an equation such that $cy_{i_1} \ldots y_{i_k} = c$ and $dy_{j_1} \ldots y_{j_t} = d$ but $c \neq d$. Obviously no solution to $\mathcal{E}$ can then exist above $\overline{Y}$ so $\mathcal{E}$ has no solution. $\square$

In fact, this algorithm can be viewed as an instance of a classical result from the CSP literature [JCG97] showing the tractability of CSP when the relations are preserved by an associative, commutative and idempotent operation. As we noted in the first paragraph of our proof, the multiplication in S preserves the relations defined as equations over S.

Next, we look at upper bounds for the resolution of systems of target-equations. This restriction allow some more leeway and we will give an algorithm to solve T-EQN$^*_S$ over a regular band (i.e. an idempotent semigroup satisfying the identity $abaca = abca$) and use the following technical result:

Lemma 5.22 *Let S be a regular band and suppose $x_1 \ldots x_k = s$ and $y_1 \ldots y_l = s$ for some $x_i, y_i, s \in S$. For all shuffles K of $x_1 \ldots x_k$ with $y_1 \ldots y_l$, we have $K = s$.*

[3] If no such element exists, we conclude that the system is unsatisfiable.

Proof. In any idempotent semigroup, the product of two elements $\mathcal{J}$-above some $u \in S$ is also $\mathcal{J}$-above u. Hence we have $K \geq_{\mathcal{J}} s$ since each x_i, y_i lies $\mathcal{J}$-above s. On the other hand, since all x_i's appear in K, we can use the relation $abaca = abca$ to get $KsK = Kx_1 \cdots x_k K = K^2 = K$. Thus, $s \geq_{\mathcal{J}} K$ and so $s \mathcal{J} K$. Furthermore, every prefix of $x_1 \ldots x_k$ is $\mathcal{R}$-above s and so $x_1 \ldots x_i s = s$ for all $i \leq k$.

We claim that $K \geq_{\mathcal{R}} s$. Indeed, we have $Ks = Ksx_1 \ldots x_k y_1 \ldots y_l$. Using again the relation $abaca = abca$, we can replace the occurrence of x_i in K on the right-hand side of this equation with the prefix $x_1 \ldots x_i$ since all the x_j with $j \leq i$ appear both before and after x_i. Hence Ks can be written as a product of prefixes of $x_1 \cdots x_k$ or $y_1 \ldots y_l$ times s. Thus $Ks = s$ and $K \geq_{\mathcal{R}} s$.

By a symmetric argument, $K \geq_{\mathcal{L}} s$. Since $s \mathcal{J} K$, we have $s \mathcal{H} K$ and $s = K$ by aperiodicity. □

We can now prove:

Lemma 5.23 *If S is a regular band then* T-EQN$^*_S$ *lies in P.*

Proof. Our algorithm works by shrinking a list of possible values for each variable and implicitly uses the fact that the relations defined by equations over S are closed under a set function [Dal00].

For each variable x_i, $1 \leq i \leq n$, we initialize a set $A_i = S$ of "possible values" for x_i and repeat the following until either the A_i are fixed or some $A_i = \emptyset$: for all i from 1 to n, for each equation E involving x_i, and each $a_i \in A_i$, if there exists no n-tuple $(a_1, \ldots, a_i, \ldots a_n)$ with $a_j \in A_j$ that satisfies E, then we set $A_i := A_i - \{a_i\}$.

If some A_i is empty, the system clearly has no solution. Conversely, we are left with sets A_i such that for all $a_i \in A_i$ and all equations E in the system, there are $a_j \in A_j$ for all $i \neq j$ such that the n-tuple $(a_1, \ldots, a_n)$ satisfies E. We claim that this guarantees the existence of a solution to the system.

Indeed, let t_i be the product in S of all elements of $A_i = \{a_i^{(1)}, \ldots, a_i^{(s_i)}\}$ in some arbitrary order. Then $(t_1, \ldots, t_n)$ satisfies all equations in the system. To see this, consider some equation $E = x_1 x_2 \ldots x_k = s$. The product $t_1 t_2 \ldots t_k$ is a shuffle of solutions to this equation by definition of the A_i's, so by Lemma 5.22, the tuple $(t_1, \ldots, t_n)$ also satisfies the equation.

It remains to show that our algorithm runs in polynomial time. It is sufficient to show that we can efficiently test whether a given equation $x_1 \ldots x_k = s$ has a solution $\bar{a} = (a_1, \ldots, a_i, \ldots, a_n)$ where a_i is given and for each $j \neq i$ we have $a_j \in A_j$.

We will use a variant of our algorithm for EQN_M for monoids in **DA**: since S belongs to **DA**, we know that sets $S_s = \{w \in S^* | eval_S(w) = s\}$ are the disjoint union of unambiguous concatenations $S_0^* b_1 S_1^* \ldots b_t S_t^*$ with $S_i \subseteq S$. So to test for the existence of an $\bar{a}$ as above, we need only consider, for each $S_0^* b_1 S_1^* \ldots b_t S_t^*$, the $\binom{k}{t}$ ways of placing the b_i's among the k variables $x_1, \ldots, x_k$ occurring in the equation. To validate this choice, it now suffices to check that we have $b_i \in A_j$ if we set $x_j = b_i$, and, for all other variables, that the corresponding A_i contains at least one element which belongs to the right S_j's. $\square$

We can combine the result of Lemmas 5.20, 5.21, 5.23 to solve, in polynomial time, equations over the direct product of a commutative band and an Abelian group and target-equation over the direct product of a regular band and an Abelian group. The tractability of these problems can also be shown for divisors of such semigroups but this requires some additional work. We introduce two definitions:

Definition 5.24 *The semigroup S is a* strong semilattice of Abelian groups *if there exists a semilattice E (i.e. a commutative band), a family of disjoint Abelian groups $\{G_e | e \in E\}$ and for every $e, f \in E$ such that $e \geq_{\mathcal{J}} f$ a group homomorphism $\phi_{e,f} : G_e \to G_f$ such that:*

1. *S is the union of the G_e;*

2. *$\phi_{e,e} = id_{G_e}$ for all $e \in E$;*

3. *for any $e \geq_{\mathcal{J}} f \geq_{\mathcal{J}} d$ we have $\phi_{f,d} \circ \phi_{e,f} = \phi_{e,d}$;*

4. *the multiplication in S is given by the formula*

$$x \cdot y = \phi_{x^\omega, x^\omega y^\omega}(x) \cdot \phi_{y^\omega, x^\omega y^\omega}(y).$$

Similarly, we say that the semigroup S is a *strong regular band of Abelian groups* if there exists a regular band E, a family of disjoint Abelian groups $\{G_e | e \in E\}$ and for every $e, f \in E$ such that $e \geq_{\mathcal{J}} f$ a group homomorphism $\phi_{e,f} : G_e \to G_f$ satisfying the same properties.

Lemma 5.25 *Let S be a semigroup. The following are equivalent:*

1. *S is a strong semilattice of groups;*

2. *S lies in $\mathbf{J_1} \vee \mathbf{Ab}$;*

3. *S is a union of Abelian groups and $\mathcal{J} = \mathcal{H}$ over S.*

4. *S is a commutative union of groups.*

This follows from well-known facts about unions of groups (see, e.g. [How76]). Yet, we will sketch part of this proof because some of the mechanics involved will be of use later on and because it is a good warm-up for the slightly more technical proof of Lemma 5.26.

Proof.

($1 \Rightarrow 2$) Suppose S is a strong semilattice of Abelian groups with k $\mathcal{J}$-classes. Let $G = \prod_{e \in E} G_e$ and consider the subsemigroup T of $E \times G$ consisting of elements $(f, g_{e_1}, \ldots, g_{e_k})$ such that $g_{e_i} = 1$ unless $e_i \geq_{\mathcal{J}} f$. We claim that S is a morphic image of T.

Indeed, define $\psi : T \to S$ as

$$\psi(f, g_1, \ldots, g_k) = \prod_{e_i \geq_{\mathcal{J}} f} \phi_{e_i,f}(g_i).$$

Obviously, ψ is surjective. Moreover, it is a well-defined morphism since we can show that $\psi(f, g_1, \ldots, g_k) \cdot \psi(f', g'_1, \ldots, g'_k)$ is:

$$
\begin{aligned}
&= \prod_{e_i \geq_{\mathcal{J}} f} \phi_{e_i,f}(g_i) \cdot \prod_{e_i \geq_{\mathcal{J}} f'} \phi_{e_i,f'}(g'_i) && (5.1)\\
&= \prod_{e_i \geq_{\mathcal{J}} f} \phi_{e_i,f'}(g_i) \cdot \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g'_i) && (5.2)\\
&= \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g_i) \cdot \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g'_i) && (5.3)\\
&= \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g_i g'_i) && (5.4)\\
&= \psi(ff', g_1 g'_1, \ldots, g_k g'_k) && (5.5)
\end{aligned}
$$

We have (5.2) by properties 3 and 4 of Definition 5.24 and (5.3) because membership in T guarantees that $g_i = 1$ unless $e_i \geq f$. For (5.4), we use that $G_{ff'}$ is Abelian and that the ϕ's are morphisms.

$(2 \Rightarrow 3)$ is a simple exercise.

$(3 \Rightarrow 1)$ can be obtained as in the proof of Lemma 5.26.

$(4 \Leftrightarrow 1)$ is part of semigroup theory folklore (see e.g. [How76]). □

Lemma 5.26 *For a semigroup S, the following are equivalent:*

1. *S is a strong regular band of Abelian groups;*

2. *S belongs to* $\mathbf{RB} \vee \mathbf{Ab}$.

3. *S is an orthodox union of Abelian groups such that $E(S)$ is a regular band and $\mathcal{H}$ is a congruence;*

Proof.

(1 $\Rightarrow$ 2) This is similar to the corresponding implication in the proof of Lemma 5.25. Suppose S is a strong regular band of Abelian groups. From every $\mathcal{H}$-class we pick the idempotent e_i in S and the corresponding subgroup G_i. Let $G = G_1 \times \ldots \times G_k$ be the product of all such groups. Our claim is that S is a divisor of the semigroup $E(S) \times G$. Let T be the subsemigroup of S consisting of elements $(e, g_1, \ldots, g_s)$ where $g_i = 1$ whenever e_i is not $\mathcal{J}$-above e. Let $\psi : T \to S$ be defined as

$$\psi(t, g_1, \ldots, g_k) = \prod_{e_i \geq_{\mathcal{J}} t} \phi_{e_i,t}(g_i).$$

It is obvious that ψ is surjective. Moreover, it is a well defined morphism because we can show $\psi(f, g_1, \ldots, g_k) \cdot \psi(f', g_1', \ldots, g_k')$ is:

$$\begin{aligned}
&= \prod_{e_i \geq_{\mathcal{J}} f} \phi_{e_i,f}(g_i) \cdot \prod_{e_i \geq_{\mathcal{J}} f'} \phi_{e_i,f'}(g_i') && (5.6)\\
&= \prod_{e_i \geq_{\mathcal{J}} f} \phi_{e_i,f'}(g_i) \cdot \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g_i') && (5.7)\\
&= \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g_i) \cdot \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g_i') && (5.8)\\
&= \prod_{e_i \geq_{\mathcal{J}} ff'} \phi_{e_i,ff'}(g_i g_i') && (5.9)\\
&= \psi(ff', g_1 g_1', \ldots, g_k g_k') && (5.10)
\end{aligned}$$

(2 $\Rightarrow$ 3) The direct product T of a regular band and an Abelian group certainly has these properties. They are clearly preserved under morphic images. If S is a subsemigroup of T, it is an orthodox union of Abelian groups such that $E(S)$ is a regular band. Furthermore and because T is a union of groups, if $u\,\mathcal{H}_S\,v$ in S, then for any $s \in S$ we have $us\mathcal{H}_T vs$. In particular $(us)(us)^{\omega-1}vs = vs$ and since $(us)^{\omega-1}vs \in S$ we get $us\,\mathcal{H}\,vs$.

(3 $\Rightarrow$ 1) This is the hardest of the three implications. Let S be an orthodox union of Abelian groups such that $E(S)$ is a regular band and $\mathcal{H}$ is a congruence.

Then for any $x, y, e \in S$ with e idempotent we have:

$$exeye = ex(ex)^\omega e(ye)^\omega ye = ex(ex)^\omega (ye)^\omega ye = exye \quad (1)$$

since $e, (ex)^\omega, (ye)^\omega$ are elements of the regular band $E(S)$.

For any idempotents $e, f \in S$, let G_e, G_f be the maximal subgroups of S containing e and f respectively and define for any $e \geq_{\mathcal{J}} f$ the morphism $\phi_{e,f} : G_e \to G_f$ as $\phi_{e,f}(x) = fxf$. Because of (1), $\phi_{e,f}$ is a well-defined group homomorphism. Clearly, $\phi_{e,e} = id_{G_e}$ and for idempotents $d \geq_{\mathcal{J}} e \geq_{\mathcal{J}} f$ we have for any $x \in G_d$:

$$\phi_{e,f} \circ \phi_{d,e}(x) = fexef = fefxfef = fxf = \phi_{d,f}(x).$$

Clearly, S is the union of the Abelian groups G_e. Since $\mathcal{H}$ is a congruence on S, we have $xy \,\mathcal{H}\, x^\omega y^\omega$ and so $xy = x^\omega y^\omega xy x^\omega y^\omega$. By (1) we get that the multiplication in S is given by

$$x \cdot y = x^\omega y^\omega x x^\omega y^\omega y x^\omega y^\omega = \phi_{x^\omega, x^\omega y^\omega}(x) \cdot \phi_{y^\omega, x^\omega y^\omega}(y).$$

□

These structural results allow us to prove:

Lemma 5.27 *If S is a semigroup in the variety $\mathbf{J_1} \vee \mathbf{Ab}$ then EQN^*_S lies in P.*

Proof. We know that S is a strong semilattice of Abelian groups. Let $\mathcal{E}$ be a system of equations over S in n variables. If $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_n)$ are two solutions to $\mathcal{E}$, then one can easily verify that $(u_1^\omega v_1, \ldots, u_n^\omega v_n)$ is also a solution to $\mathcal{E}$. If $\mathcal{E}$ is satisfiable over S, then it must be also be satisfiable in $S/\mathcal{H} = E(S)$ (note that $\mathcal{H} = \mathcal{J}$ is a congruence over S since S is a union of groups) and since $E(S)$ is a semilattice, we can find in polynomial time the $\mathcal{J}$-minimal solution of this system.

In other words, we can find in polynomial time idempotents $t_1, \dots t_n$ such that if $\mathcal{E}$ has a solution then it has a solution $(y_1, \dots, y_n)$ such that $y_i^\omega = t_i$ for each $1 \leq i \leq n$. We next check if such $(y_1, \dots, y_n)$ exists.

Recall from our previous proof of Lemma 5.25 that if $G = \prod_{e \in E} G_e$ and T is the subsemigroup of $E \times G$ consisting of elements $(f, g_{e_1}, \dots, g_{e_k})$ such that $g_{e_i} = 1$ unless $e_i \geq_{\mathcal{J}} f$, then $S = \psi(T)$ where

$$\psi(f, g_1, \dots, g_k) = \prod_{e_i \geq_{\mathcal{J}} f} \phi_{e_i, f}(g_i).$$

Consider now the following constraint satisfiability problem over this group G with variables $X_1, \dots, X_n$. First, we constrain every X_i so that (t_i, X_i) belongs to T. Next, we insure that the X_i are such that setting $z_i = \psi(t_i, X_i)$ we get a solution to $\mathcal{E}$. It is clear that these constraints can be satisfied in G if and only if $\mathcal{E}$ has a solution.

Now observe that, as we pointed out in our remark following Lemma 5.20, every relation of arity k used to build the above constraints forms a coset of G^k. Thus, using Theorem 5.19, we can test for the existence of such X_i. □

As in the case of EQN_S^*, we can exploit further the ideas presented in the algorithm of Lemma 5.23 to obtain an upper bound for a larger class with the help of our structural result.

Lemma 5.28 *If S is a semigroup in* $\mathbf{RB} \vee \mathbf{Ab}$ *then* T-EQN_S^* *lies in P.*

Proof. We proceed exactly as in the proof of Lemma 5.27. We can adapt Lemma 5.22 to show that if we have $x_1 \dots x_k = s$ and $y_1 \dots y_l = s$ then for any shuffle K of $x_1 \dots x_k$ and $y_1^\omega \dots y_l^\omega$ we have $K = s$.

In particular, if $\mathcal{E}$ is a system of equations over S in n variables, we can use the algorithm of Lemma 5.23 to find idempotents $t_1, \dots t_n$ such that if $\mathcal{E}$ has a solution then it has a solution $(y_1, \dots, y_n)$ such that $y_i^\omega = t_i$ for each $1 \leq i \leq n$. Once again, it remains to check if such $(y_1, \dots, y_n)$ exist.

As for Lemma 5.27, we can formulate this question as a constraint satisfiability problem over the product of all subgroups of S. Once more, every relation used forms a coset in some power of G so this problem can be solved in polynomial time. □

5.2.3 Hardness Results

Next, we obtain a number of NP-completeness results which, in some cases, will be combined with the upper bounds presented above to provide complete dichotomies for the complexity of EQN* and T-EQN*. Recall that in order to establish a hardness result on the complexity of equation satisfiability for a semigroup S, it is sufficient to prove the hardness results for inducible subsets of S. We will make extensive use of this fact in the arguments below. In particular, for any monoid M, the set of elements $\mathcal{J}$ (resp. $\mathcal{L}$, $\mathcal{R}$) below an element m is inducible by the expression x_1mx_2 (resp. x_1m, mx_1). Also, for any idempotent e in a semigroup S, the expression exe induces the submonoid eSe and the set of idempotents can be induced by the expression x^ω. Consequently, we will loosely use sentences such as "we restrict the variable x_i to be idempotent and $\mathcal{J}$-below m" to improve the readability of our reductions.

If m is a regular element of a semigroup S. The target equation $x_1yx_2 = m$ can be satisfied if and only if y is $\mathcal{J}$-above m. To stress the intended meaning of certain equations, we will sometimes write $y \geq_{\mathcal{J}} m$ in place of such an equation.

We will use reductions from the following NP-complete problems: 3SAT, 1-3SAT, NAE 3SAT, MONOTONE NAE3SAT and GRAPH k-COLORABILITY. The NP-completeness of the first four are guaranteed by Schaefer's Theorem. MONOTONE NAE3SAT is perhaps the lesser known problem in the list: it is the variant of NAE3SAT in which no clause contains a negated literal. It is sometimes presented as the MONOCHROMATIC TRIANGLE problem [GJ79].

In the rest of this section we will systematically use M to denote a finite monoid and S to denote a finite semigroup in the statement of lemmas.

Lemma 5.29 *If S is not in $\overline{\mathbf{Ab}}$ then* T-EQN$_S^*$ *is NP-complete.*

Proof. This is a simple corollary of the already oft-cited NP-completeness of T-EQN$_G^*$ for non-Abelian G proved in [GR99]. If G is a maximal non-Abelian subgroup of S with idempotent e, the expression eSe induces a submonoid in which every element is either in G or lying $\mathcal{J}$-below e. So any system of target-equations over with targets in G is satisfiable in G if and only if it is satisfiable in eSe. □

In contrast to the group case, there are commutative aperiodic monoids for which T-EQN$_M^*$ is already NP-complete.

Lemma 5.30 *If M is aperiodic but not idempotent, then* T-EQN$_M^*$ *is NP-complete.*

Proof. Let $m \neq m^2$ be a $\mathcal{J}$-maximal non-idempotent element of M.

We use the following reduction from 1-3SAT: for each Boolean variable X_i in the formula, we create two variables x_i, $\bar{x}_i$ for the system and create equations

$$(1) \quad x_i\bar{x}_i = m \quad \text{and} \qquad (2) \quad \bar{x}_i x_i = m.$$

Moreover, for each clause of the formula, e.g. $(X_1 \vee \bar{X}_2 \vee X_3)$ we add the equation

$$(3)\ x_1\bar{x}_2x_3 = m.$$

Suppose first that the 1-3SAT formula is satisfiable. Then we can satisfy the resulting system of equations by setting $x_i = m$ and $\bar{x}_i = 1$ whenever X_i is TRUE, and $x_i = 1$ and $\bar{x}_i = m$ whenever X_i is FALSE. It is easy to see that this satisfies the sets of Equations (1), (2) and (3).

Conversely, suppose that this system of equations is satisfiable. Note first that Equations (1) and (2) force x_i (resp. $\bar{x}_i$) to be both $\mathcal{R}$-above and $\mathcal{L}$-above

m. So if x_i (say) lies in m's $\mathcal{J}$-class, it must be $\mathcal{R}$ and $\mathcal{L}$ equivalent and hence $\mathcal{H}$-equivalent to m. By aperiodicity, this implies in fact $x_i = m$. It follows that at least one of x_i or $\bar{x}_i$ lies strictly $\mathcal{J}$-above m; otherwise we would have $x_i = \bar{x}_i = m$, and since m is not idempotent this would violate Equation (1). Moreover, since m is $\mathcal{J}$-maximal among the non-idempotent elements of M, whichever one of x_i, $\bar{x}_i$ is strictly $\mathcal{J}$-above m must be some idempotent e.

Therefore, suppose $x_i = e$ where $e >_{\mathcal{J}} m$ is idempotent. Then Equation (1) gives us $m = e\bar{x}_i = ee\bar{x}_i = em$, and similarly (2) gives $m = me$. We cannot also have $\bar{x}_i >_{\mathcal{J}} m$, since then $\bar{x}_i$ would also be idempotent, which this leads to the contradiction $m^2 = e\bar{x}_i\bar{x}_ie = e\bar{x}_ie = em = m$. Thus $\bar{x}_i \, \mathcal{J} \, m$ whence $\bar{x}_i \mathcal{H} m$ and from aperiodicity $\bar{x}_i = m$.

Similarly, if $\bar{x}_i >_{\mathcal{J}} m$ then $x_i = m$. So if we set X_i to TRUE when $x_i = m$ and FALSE when $\bar{x}_i = m$, Eqs. (1) and (2) insure that our mapping between Boolean variables and variables in M is consistent, in the sense that for all i, exactly one of x_i, $\bar{x}_i$ is m and the other is an idempotent in a higher $\mathcal{J}$-class.

Finally, suppose that all 3 variables in Equation (3) have idempotent values. By a previous argument, these values fix m and so $m^2 = x_1\bar{x}_2x_3m = x_1\bar{x}_2m = x_1m = m$, a contradiction. We get a similar contradiction if two or more of the variables are set to m. Therefore (3) insures that exactly one literal in each clause is true, and the 1-3SAT formula is satisfiable. □

In fact, this is a special case of the more general hardness result:

Lemma 5.31 *If M is not a union of groups, then* T-EQN$^*_M$ *is NP-complete.*

Proof. Let m be a $\mathcal{J}$-maximal element satisfying $m^2 <_{\mathcal{J}} m$. Any monoid element u with $u >_{\mathcal{J}} m$ is $\mathcal{H}$-related to u^ω and $u^{\omega+1} = u$.

We use the same reduction from 1-3SAT as in our previous proof: for each literal in the formula X_i and its complement $\overline{X}_i$, we add equations $x_i\bar{x}_i = m$

and $\bar{x}_i x_i = m$ and for each clause we add the equation e.g.

$$x_1 \bar{x}_2 x_3 = m.$$

One can easily check that given a satisfying assignment to the formula we can obtain a satisfying assignment to the system of equations.

For the converse, Eqs. (1) and (2) show that if $x_i \mathcal{J} m$ then in fact $x_i \mathcal{H} m$, just as in Lemma 5.30. Since the $\mathcal{H}$-class of m contains no idempotent, the product of any two elements of $\mathcal{H}_m$ lies strictly $\mathcal{J}$-below m. Eqs. (1) and (2) thus force at least one of x_i, $\bar{x}_i$ to be strictly $\mathcal{J}$-above m.

Suppose both x_i and $\bar{x}_i$ are strictly $\mathcal{J}$-above m. Then we have $m = x_i \bar{x}_i = x_i \bar{x}_i^{\omega+1} = m\bar{x}_i^{\omega}$ and $m = m x_i^q$. Moreover, x_i and $\bar{x}_i$ commute by Eqs. (1) and (2), so we get $m = m x_i^{\omega} \bar{x}_i^{\omega} = m(x_i \bar{x}_i)^{\omega} = m^{\omega+1} <_{\mathcal{J}} m$, a contradiction. Therefore, at least one of $x_i, \bar{x}_i$ must be $\mathcal{H}$-equivalent to m while the other fixes the $\mathcal{H}$-class of m, so if we identify true literals with variables taking a value $\mathcal{H}$-equivalent to m, we obtain a consistent truth assignment, and, repeating the argument of the previous proof, exactly one literal in each clause corresponding to Equation (3) must be true. □

We will see later that these hardness results do not hold in the case of semigroups.

Lemma 5.32 *If M is aperiodic and idempotent but is not commutative, then* EQN^*_M *is NP-complete.*

Proof. Let a, b in M be such that $ab \neq ba$. We can choose a, b such that a is a $\mathcal{J}$-maximal element which is not *central* in M (i.e. which does not commute with every element) and b is a $\mathcal{J}$-maximal element which does not commute with a. We now obtain a reduction from 3SAT. For each Boolean variable X_i in the formula, we create variables $x_i, \bar{x}_i, y_i, \bar{y}_i$ and equations

$$\begin{array}{llll} (1) & x_i\bar{x}_i = a & (2) & \bar{x}_i x_i = a \\ (3) & y_i\bar{y}_i = b & (4) & \bar{y}_i y_i = b \\ (5) & x_i\bar{y}_i = \bar{y}_i x_i & (6) & \bar{x}_i y_i = y_i\bar{x}_i \end{array}$$

Also, for each 3SAT clause, e.g. $X_1 \vee \bar{X}_2 \vee X_3$, we add an equation

$$(7) \quad x_1\bar{x}_2x_3 = a.$$

Given a satisfying assignment to the formula, we can construct a solution to the above system by setting $x_i = a$, $\bar{x}_i = 1$, $y_i = b$, and $\bar{y}_i = 1$ whenever X_i is TRUE, and $x_i = 1$, $\bar{x}_i = a$, $y_i = 1$, $\bar{y}_i = b$ whenever X_i is FALSE.

Conversely, suppose the system of equations is satisfiable. Equation (1) shows that both x_i and $\bar{x}_i$ lie $\mathcal{J}$-above a. Since a and b don't commute, a cannot be the product of two elements commuting with b. However, any element strictly $\mathcal{J}$-above a is central so at least one of x_i, $\bar{x}_i$ must be $\mathcal{J}$-equivalent to a. Moreover, Eqs. (1) and (2) insure that x_i, $\bar{x}_i$ are both $\mathcal{L}$-above and $\mathcal{R}$-above a, so if $x_i \,\mathcal{J}\, a$ (say) we must also have $x \,\mathcal{H}\, a$ and thus $x = a$ by aperiodicity. Thus at least one of x_i, $\bar{x}_i$ must be a. Similarly at least one of y_i, $\bar{y}_i$ must be b, since any elements strictly $\mathcal{J}$-above b commute with a.

If $x_i = a$, then $\bar{y}_i$ commutes with a by Eq. (5). Thus $\bar{y}_i$ must be strictly $\mathcal{J}$-above b. If $y_i = b$, then $\bar{x}_i$ commutes with b by Eq. (6), so $\bar{x}_i$ is strictly $\mathcal{J}$-above a. We can thus obtain a consistent truth assignment to the literals by setting X_i to TRUE if and only $x_i = a$ and $y_i = b$ and $\overline{X}_i$ to TRUE if and only $\bar{x}_i = a$ and $\bar{y}_i = b$.

Since every element strictly $\mathcal{J}$-above a is central but a is not, a cannot be a product of elements $\mathcal{J}$-above it. Therefore, if $x_1\bar{x}_2x_3 = a$ then one of $x_1, \bar{x}_2, x_3$ must be a, so the corresponding 3SAT clause is satisfied. □

Corollary 5.33 *If S is a band but is not a normal band (i.e. it does not satisfy $abca = acba$) then EQN^*_S is NP-complete.*

Proof. We claim that S is normal if and only if it is locally $\mathbf{J_1}$. Indeed, if S is normal then for any a, b, c we have $abaca = abacaa = acabaa = acaba$ and so S is locally $\mathbf{J_1}$.

Also, every band in $\mathbf{LJ_1}$ is regular because we have

$$abca = abababcacaca = (aba)(aba)(abca)(aca)(aca) = abacabcabaca = abaca.$$

Thus if S is a band in $\mathbf{LJ_1}$, we have

$$abca = abaca = acaba = acba$$

which proves our claim.

Every monoid eSe is inducible so if is S is a band which is not normal, then it must have an inducible submonoid for which EQN^*_S is NP-complete. $\square$

The following matches the upper bound of Lemma 5.23:

Lemma 5.34 *Let S be a band outside* $\mathbf{RB}$*, then* T-EQN^*_S *is NP-complete.*

Proof. Since S is not a regular band, there are $A, B, C \in S$ be such that $ABACA \neq ABCA$ and we choose A, B, C such that $ABCA$ is $\mathcal{J}$-maximal. We can assume without loss of generality that $ABACA$ is not $\mathcal{R}$-related to $ABCA$ for otherwise $ABACA$ is not $\mathcal{L}$-related to $ABCA$ and we can proceed dually.

Setting $a = ABA$, $b = AB$, $c = CABA$, we obtain $ab = b$, $ba = a$, $ca = c$. Also, $ac = abaca \neq abca = bc$ since $abaca = ABACABA$ is $\mathcal{R}$-related to $ABACA$ and $abca = ABCABA$ is $\mathcal{R}$-related to $ABCA$.

Let $a, b, c \in S$ as above be elements such that $ab = b$, $ba = a$, $ca = c$, $ac \neq bc$ and c is $\mathcal{J}$-maximal with the properties $abaca \neq abca$ and $c <_{\mathcal{J}} a, b$. We claim that for all $s \in S$ satisfying $a \geq_{\mathcal{J}} s >_{\mathcal{J}} c$ we have in fact $as = bs$. Indeed, the $\mathcal{J}$-maximality of c imposes $abasa = absa$. Since $ab = b$ and $ba = b$ this shows that $asa = bsa$ and so $as = asas = bsas = bs$.

We can now obtain the following reduction from 3SAT to T-EQN$^*_T$ where T is the (inducible) semigroup of elements lying $\mathcal{J}$-below a. For each Boolean literal X_i in the formula, we introduce the variables $x_i, \bar{x}_i, y_i$ and construct the equations

$$\begin{array}{llll} (1) & cx_i = c & (2) & c\bar{x}_i = c \\ (3) & y_i a = a & (4) & ax_i a\bar{x}_i = ac \\ (5) & bx_i b\bar{x}_i = bc & (6) & y_i x_i ac = ac \\ (7) & y_i\bar{x}_i bc = bc & & \end{array}$$

Moreover for any q which is $\mathcal{R}$-related to a we add the equations

$$(8) \quad qx_i qc = qc \qquad\qquad (9) \quad q\bar{x}_i qc = qc$$

Note that in any solution to these equations we know from Eqs. (1,2) that both x_i and $\bar{x}_i$ lie $\mathcal{J}$-above c. Suppose that both lie *strictly* $\mathcal{J}$-above c then by our previous remarks $ax_i = bx_i$ and $a\bar{x}_i = b\bar{x}_i$. But then $ax_i a\bar{x}_i = bx_i b\bar{x}_i$ and this contradicts Eqs. (4,5).

Suppose on the other hand that both x_i and $\bar{x}_i$ are $\mathcal{J}$-related to c: by Eqs. (1,2) we get $x_i \,\mathcal{L}\, \bar{x}_i \,\mathcal{L}\, c$. We thus have $x_i = x_i ac \quad \bar{x}_i bc = \bar{x}_i$ and in fact $x_i yc = x_i$ for any $y \geq_{\mathcal{J}} c$. Since Eq. 3 imposes $y_i \,\mathcal{R}\, a$ we deduce from Eqs. (8,9) that

$$y_i x_i ac = y_i x_i = y_i x_i y_i c = y_i c = y_i \bar{x}_i y_i c = y_i \bar{x}_i bc.$$

This, however contradicts Eqs. (6,7). Hence, exactly one of $x_i, \bar{x}_i$ is $\mathcal{J}$-related to c and the other lies strictly $\mathcal{J}$-above c.

We complete our reduction by introducing, for each of clause of the 3SAT formula, e.g. $X_1 \vee \bar{X}_2 \vee X_3$, the pair of the equations:

$$(10) \quad ax_1 a\bar{x}_2 ax_3 = ac \qquad\qquad (11) \quad bx_1 b\bar{x}_2 bx_3 = bc$$

One can now verify that if the 3SAT instance is satisfiable, then we can satisfy the system obtained through our reduction by letting $x_i = c$, $\bar{x}_i = a$, $y_i = a$ whenever X_i is TRUE, and $x_i = a$, $\bar{x}_i = c$, $y_i = b$ whenever X_i is FALSE.

Conversely, suppose the system of the equations is satisfiable. Since exactly one of $x_i, \bar{x}_i$ is $\mathcal{J}$-related to c, we get a consistent truth assignment to the literals by setting X_i (resp. $\overline{X}_i$) to TRUE if and only if $x_i \mathcal{J} c$ (resp. $\bar{x}_i \mathcal{J} c$). This assignment satisfies every clause of the original formula for if the variables occurring in Eq. (11) all lie strictly $\mathcal{J}$-above c we have $ax_1 = bx_1 \quad a\bar{x}_2 = b\bar{x}_2$ and $ax_3 = bx_3$ so that

$$ax_1 a\bar{x}_2 ax_3 = bx_1 b\bar{x}_2 bx_3$$

in violation of Eqs. (11,12). □

Lemma 5.35 *If S is a union of groups but is not orthodox then* T-EQN$^*_S$ *is NP-complete.*

Proof. We can assume without loss of generality that S is a completely simple unorthodox semigroup. Otherwise, we know that it contains such a subsemigroup S'. For any $s \in S'$, the expression $x_1 s x_2$ induces the subsemigroup of elements $\mathcal{J}$-below S'. Furthermore, if t lies in S' we can use target-equations such as $(txt)^\omega = t^\omega$ to restrict variables to values lying $\mathcal{J}$-above S' and so NP-completeness for T-EQN$^*_{S'}$ implies NP-completeness for T-EQN$^*_S$.

We consider the Rees matrix representation of the completely simple semigroup S: suppose S has α $\mathcal{R}$-classes and β $\mathcal{L}$-classes. There exists a group G and a matrix $R \in G^{\alpha \times \beta}$ such that elements of S can be represented as triples (i, g, j) with $g \in G$, $1 \leq i \leq \alpha$, $1 \leq j \leq \beta$ and multiplication given by

$$(i_1, g_1, j_1) \cdot (i_2, g_2, j_2) = (i_1, g_1 \cdot R_{j_1, i_2} \cdot g_2, j_2)$$

We can assume that the first row and first column of R contain only the identity of the group e.

We can recursively reorder the rows and columns in the following way: suppose row k is such that $R_{k,i} = e$ for every $i \leq t$. We choose the row $(k+1)$ as

$$\begin{pmatrix}
e & e & \dots & \dots & \dots & \dots & \dots & \dots & e \\
e & e & \dots & \dots & \dots & \dots & \dots & \dots & e \\
e & e & \dots & \dots & e & R_{a,b} & * & * & * \\
e & e & \dots & \dots & e & * & * & * & * \\
e & \dots & e & * & * & R_{c,b} & ? & ? & ? \\
e & e & * & ? & ? & ? & ? & ? & ? \\
e & * & ? & ? & ? & ? & ? & ? & ?
\end{pmatrix}$$

Figure 5.2.3: Rees matrix of S after reordering: all entries above the dotted line are e. The $*$'s represent entries which cannot be e.

the one with the most number of e's among $R_{k+1,i}$ with $i \leq t$ and reorder the columns such that all these entries appear first in the row.

Because we assumed that the class is not orthodox, there is some non-identity entry in R so after reordering, we can, as shown in Figure 5.2.3, find indices a, b, c with $R_{a,b} \neq e$ and $a < c$ and such that

- $R_{i,j} = e$ for all $1 \leq j < b$ if and only if $i < c$;
- for all $a \leq i < c$ and all $j \geq b$, we have $R_{i,j} \neq e$.

We can now obtain a reduction from 1-3SAT in the following way: for each Boolean variable X_i we create variables $x_i, \bar{x}_i$ and force them to be idempotent. We begin by adding a number of equations to constrain the x_i and $\bar{x}_i$ in a helpful way. We first impose the equations

$$(1) \quad x_i\bar{x}_i = (1, e, 1) \qquad\qquad (2) \quad x_i(b, e, a)\bar{x}_i = (1, R_{a,b}, 1)$$

In any solution to the system, we must have $x_i = (1, e, k_i)$ and $\bar{x}_i = (t_i, e, 1)$ since the first row and column of R consist of all e's and the variables are constrained to take on only idempotent values. Equation 1 thus further forces $R_{k_i,t_i} = e$ and from Eq. 2 we have $R_{k_i,b} \cdot R_{a,t_i} = R_{a,b}$. Similarly we require that $R_{k_i,j} = e$ for all $1 \leq j < b$ by using equations of the form:

$$(3) \quad x_i(j, e, 1) = (1, e, 1)$$

We thus have insured that $k_i < c$ and in fact that either $k_i < a$ or $t_i < b$ for otherwise $R_{k_i,t_i} \neq e$.

For a clause $X_1 \vee \bar{X}_2 \vee X_3$ we wish to add the requirement $R_{k_1,b} \cdot R_{a,t_2} \cdot R_{k_3,b} = R_{a,b}$. This can be encoded as an equation such as:

$$(4) \quad x_1(b,e,a)\bar{x}_2(1,e,1)x_3(b,e,1) = (1, R_{a,b}, 1)$$

If the 1-3SAT is satisfiable, then the system can be satisfied by setting $x_i = (1,e,a)$ and $\bar{x}_i = (1,e,1)$ whenever X_i is TRUE and $x_i = (1,e,1)$ and $\bar{x}_i = (b,e,1)$ whenever X_i is FALSE.

For the converse, suppose first that $R_{a,b}$ does not have order 2. Note that if $k_i < a$ then $R_{k_i,b} = e$ and so $R_{a,t_i} = R_{a,b}$ whereas if $t_i < b$ then $R_{a,t_i} = e$ so $R_{k_i,b} = R_{a,b}$. Hence, we get a well-defined truth assignment by setting X_i to TRUE if $R_{k_i,b} = R_{a,b}$ and $R_{a,t_i} = e$ and setting X_i to FALSE if $R_{k_i,b} = e$ and $R_{a,t_i} = R_{a,b}$. For any clause in the 1-3SAT instance, say $X_1 \vee \bar{X}_2 \vee X_3$, there is an equation of type (4) imposing $R_{k_1,b} \cdot R_{a,t_2} \cdot R_{k_3,b} = R_{a,b}$ and since $R_{a,b}$ does not have order 2, exactly one of $R_{k_1,b}, R_{a,t_2}, R_{k_3,b}$ is $R_{a,b}$ and the other two are e so exactly one literal per clause is TRUE.

If $R_{a,b}$ does have order 2, our last argument breaks down because equations of type (4) might be satisfied even if all three of $R_{k_1,b}, R_{a,t_2}, R_{k_3,b}$ are equal to $R_{a,b}$. By appealing once again to Schaefer's Theorem, we can assume that each clause in the 1-3SAT instance contains at least one negated and one unnegated literal, say X_1 and $\overline{X}_2$. For a clause $X_1 \vee \bar{X}_2 \vee X_3$, Eq. 4 imposes $X_1 \oplus \bar{X}_2 \oplus X_3$. If we can further guarantee that one of X_1 or $\overline{X}_2$ is FALSE, we will be able to conclude that exactly one of the three literals in the clause is TRUE. We do so by adding the constraint $R_{k_1,t_2} = e$ using equation

$$(5) \quad x_1\bar{x}_2 = (1,e,1)$$

which cannot be satisfied if $k_1 < a$ and $t_2 < b$. Thus, one of X_1 and $\overline{X}_2$ is assigned the value FALSE. □

This can be used to prove the NP-completeness of T-EQN$^*_S$ for any S that contains a completely simple unorthodox subsemigroup. We can also prove:

Corollary 5.36 *Let S be a completely simple semigroup (i.e. a single $\mathcal{J}$-class). Then* EQN$^*_S$ *is in P if S is orthodox and has only commutative subgroups but* T-EQN$^*_S$ *is NP-complete otherwise.*

Proof. The hardness result follows directly from the previous lemma. The upper bound stems from the observation that a completely simple orthodox semigroup is the direct product of an Abelian group and an idempotent semigroup satisfying $xyz = xz$ for all $x, y, z \in S$. Systems over such bands are obviously solvable in polynomial time. $\square$

Lemma 5.37 *If S is an orthodox union of groups such that $\mathcal{H}$ is not a congruence on S, then* T-EQN$^*_S$ *is NP-complete.*

Proof. Suppose we have $a, b, c \in S$ such that $a \,\mathcal{H}\, b$ but $ac \not\mathcal{H}\, bc$ (the dual argument can be used if we have $ca \not\mathcal{H}\, cb$). In fact, it is easy to see that there exists a and an idempotent e lying $\mathcal{J}$-below a such that $ae \not\mathcal{H}\, a^\omega e$.

We choose a and e as $\mathcal{J}$-maximal such that $ae \not\mathcal{H}\, a^\omega e$. So for any $x \in S$ with $a >_\mathcal{J} x >_\mathcal{J} e$, and any $y \,\mathcal{H}\, z \,\mathcal{J}\, x$, we have both $ye \,\mathcal{H}\, ze$ and $a^\omega y \,\mathcal{H}\, az$. In particular, we cannot have $xe = ae$ for otherwise, since $a^\omega x \,\mathcal{H}\, a^{\omega-1} x \,\mathcal{J}\, x$ we have

$$a^\omega e = a^{\omega-1} xe \,\mathcal{H}\, a^\omega xe = ae$$

a contradiction. Similarly, we cannot have $xae \,\mathcal{H}\, ae$.

We can further assume without loss of generality that $ea^\omega = e$ for otherwise the idempotent $f = (ea^\omega)^\omega$ has the property that

$$af \,\mathcal{R}\, ae \not\mathcal{R}\, a^\omega e \,\mathcal{R}\, a^\omega f$$

and $fa^\omega = f$.

We build a reduction from 3SAT as follows. For each Boolean variable X_i, we introduce variables $x_i, \bar{x}_i, v_i$ such that v_i is $\mathcal{H}$-related to a and add the equations:

$$\begin{array}{llll} (1) & x_i e = ae & (2) & (\bar{x}_i ae)^\omega = (ae)^\omega \\ (3) & v_i x_i ae \geq_{\mathcal{R}} ae & (4) & v_i a \bar{x}_i e \geq_{\mathcal{R}} ae \\ (5) & (v_i x_i \bar{x}_i)^\omega = (ae)^\omega & & \end{array}$$

Moreover, for each 3SAT clause, e.g. $X_1 \vee \overline{X}_2 \vee X_3$ we introduce a variable w_j that is $\mathcal{H}$-related to a and the equation

$$(6) \quad w_j x_1 \bar{x}_2 x_3 = ae.$$

Given an assignment to the Boolean literals satisfying the 3SAT formula, one can verify that this system has a solution by setting $x_i = ae$, $\bar{x}_i = a^\omega$ and $v_i = a^\omega$ whenever X_i is TRUE and $x_i = a$, $\bar{x}_i = ae$ and $v_i = a^{\omega-1}$ whenever X_i is FALSE.

Conversely, suppose that there exists a solution to the constructed system. Equations (1,2) show that x_i and $\bar{x}_i$ are $\mathcal{R}$-above ae while Eq. (5) forces at least one of them to lie $\mathcal{J}$-below a. If we suppose on the other hand that they are both $\mathcal{J}$-related to e then from Eqs. (2,3) we have $v_i x_i \,\mathcal{R}\, ae \,\mathcal{R}\, v_i a \bar{x}_i$ and thus $v_i a^2 e \,\mathcal{R}\, v_i ae$ which is a contradiction since $v_i \,\mathcal{H}\, a$. By the remarks made above, neither x_i nor $\bar{x}_i$ must however be $\mathcal{J}$-related to one of e or a. If we set X_i (resp. $\overline{X}_i$) to TRUE if and only if $x_i \mathcal{J} e$ (resp. $\bar{x}_i \mathcal{J} e$), our assignment is consistent. Furthermore Eq. (6) guarantees that every clause in the 3SAT instance contains at least one TRUE literal for otherwise the corresponding product $w_j x_1 \bar{x}_2 x_3$ will be $\mathcal{J}$-related to a. □

5.2.4 Dichotomy Theorems for EQN^*_M and T-EQN^*_M over Monoids

In the case of monoids, it is possible to combine the upper bounds and NP-completeness results to obtain complete dichotomies for the complexity of EQN^*_M

and T-EQN$^*_M$. This involves the structural results about the varieties $\mathbf{J_1} \vee \mathbf{Ab}$ and $\mathbf{RB} \vee \mathbf{Ab}$.

Theorem 5.38 *For any monoid M, we have* EQN$^*_M$ *lying in P if M belongs to $\mathbf{J_1} \vee \mathbf{Ab}$ and* EQN$^*_M$ *is NP-complete otherwise.*

Proof. The upper bound is Lemma 5.27. On the other hand, EQN$^*_M$ is NP-complete if M either contains a non-Abelian subgroup (Lemma 5.29) or is not a union of groups (Lemma 5.31). If it is a union of groups but is not orthodox, we can appeal to Lemma 5.35. Finally, if M is an orthodox union of groups, but $E(M)$ fails to be $\mathcal{J}$-trivial, then NP-completeness follows from Lemma 5.32 because $E(M)$ is an inducible submonoid. Otherwise, M is an orthodox union of Abelian groups with $E(M)$ commutative and must thus belong to $\mathbf{J_1} \vee \mathbf{Ab}$. □

Similarly, our characterization of $\mathbf{RB} \vee \mathbf{Ab}$ allows us to prove

Theorem 5.39 *For any monoid M, we have* T-EQN$^*_M$ *lying in P if M belongs to $\mathbf{RB} \vee \mathbf{Ab}$ and* T-EQN$^*_M$ *is NP-complete otherwise.*

Proof. The upper bound is Lemma 5.28. As we argued previously, T-EQN$^*_M$ is NP-complete unless it is an orthodox union of Abelian groups. If the latter holds, however, we still have NP-completeness if $E(M)$ does not form a regular band (Lemma 5.34). Finally, by Lemma 5.37, we have NP-completeness unless M is an orthodox union of Abelian groups such that $E(M)$ is a regular band and $\mathcal{H}$ forms a congruence. By Lemma 5.26, this mean that we can show NP-completeness of T-EQN$^*_M$ for any M not belonging to $\mathbf{RB} \vee \mathbf{Ab}$. □

5.2.5 Results and Questions in the Semigroup Case

Do similar dichotomies hold in the case of semigroups? While this is very tempting to conjecture, one is faced with an obstacle illustrated in the following example.

Example 5.39. Consider the semigroup K with three generators[4] r, g, b, elements $\{r, g, b, rr, rg, rb, gr, gg, gb, br, bg, bb, 0\}$ and such that $xyz = 0$ for any $x, y, z \in K$. Since the product of any three elements of K is 0, any equation over K can be assumed to have at most two variables or constants on either side and any equation of the form $x_1 x_2 = ab$ with $a, b \in \{r, g, b\}$ is equivalent to the two equations $x_1 = a$ and $x_2 = b$.

To solve a system $\mathcal{E}$ of equations over K with variables $x_1, \ldots, x_n$, we can proceed as follows. If there are no equations of the form $x_i = c$ or $x_i x_j = ab$, then we know that the all-0 assignment satisfies $\mathcal{E}$. Otherwise, the values of x_i and x_j are now forced and we can replace their occurrences in $\mathcal{E}$ with the appropriate constants. The new system thus obtained has strictly fewer variables and we can repeat this strategy until we either obtain a satisfying assignment for $\mathcal{E}$ or obtain an obviously unsatisfiable system. This algorithm clearly solves EQN^*_K in polynomial time.

In contrast, consider the semigroup T with generators r, g, b, elements $\{r, g, b, E, N, 0\}$ with 0 being the sole idempotent and such that the square of any generator is E, the product of any two distinct generators is N and any other product is 0. One can verify that T is a morphic image of K.

Yet, we claim that T-EQN^*_T is NP-complete. We use a reduction from 3-Colorability: for every node v_i in the graph G we create a variable x_i and add the equation $x_i^2 = E$. Furthermore, for every edge (v_i, v_j), we add equation $x_i x_j = N$. If the original graph can be colored using colors Red, Green and Blue then the system can be satisfied by setting $x_i = r$ (resp. g, b) if and only if v_i is colored Red (resp. Green, Blue). Conversely, in any satisfying assignment to the system, each x_i is assigned one of r, g or b and no pair x_i, x_j with $(v_i, v_j) \in G$ is assigned the same generator so the graph is

[4]In semigroup jargon, K is the free nilpotent semigroup of threshold 3 over three generators.

3-colorable.

In light of this example, the class of semigroups S for which EQN_S^* (or T-EQN$_S^*$) lies in P does not form a variety (unless $P = NP$). We therefore choose, as a first step, to restrict our attention to the complexity of solving systems over *regular* semigroups where, empirically, such phenomena do not seem to occur. Note also that K is an example of a semigroup which is not a union of groups but for which EQN$_K^*$ is tractable, in sharp contrast of Lemma 5.31 in the case of monoids.

Lemma 5.40 *Let S be a regular semigroup consisting of*

- *a $\mathcal{J}$-maximal $\mathcal{J}$-class B with at least two $\mathcal{R}$-classes and with exactly one idempotent per $\mathcal{L}$- and $\mathcal{R}$-class;*
- *$\mathcal{J}$-classes below B are all subgroups.*

Then T-EQN$_S^*$ *is NP-complete.*

Proof. We know that B is a square $\mathcal{J}$-class containing exactly one idempotent per $\mathcal{R}$-class and per $\mathcal{L}$-class. Therefore the product of any two distinct idempotents of B does not lie in B.

Let H be a $\mathcal{J}$-maximal $\mathcal{H}$-class such that $xy \in H$ for some idempotents $x, y \in B$. Let $E_H = \{x : x^2 = x, x \in B, xy \in H \text{ for some } y \in B\}$ and let e_H be the unique idempotent of H. Note that if z is an idempotent of B which is not in E_H then neither xz nor zx lie in H for any idempotent $x \in B$. Also, for any distinct $x, y \in E_H$, we have both xy and yx lying in H. Indeed, we must have $e_H x$ and $y e_H$ lying in H so xy must lie $\mathcal{J}$-below B but $\mathcal{J}$-above H so xy lies in H.

Suppose first that $|E_H| = 2$, i.e. $E_H = \{a, b\}$. We build a reduction from Monotone nae3sat as follows: for each variable X_i in the formula, we create the variables $x_i, \bar{x}_i$, force them to be idempotents in B and add the equation

$$(1) \quad (x_i\bar{x}_i)^\omega = e_H.$$

Moreover, for any clause $X_1 \vee X_2 \vee X_3$ we add the two equations

$$(2) \quad (x_1x_2x_3)^\omega = e_H \qquad\qquad (3) \quad (\bar{x}_1\bar{x}_2\bar{x}_3)^\omega = e_H$$

One can easily verify that if we are given an assignment to the X_i satisfying the MONOTONE NAE3SAT instance, we can obtain a solution to the system by letting $x_i = a$, $\bar{x} = b$ if X_i is TRUE and $x_i = b$, $\bar{x} = a$ if X_i is FALSE otherwise.

Conversely, consider any solution to this system of equations. Equation 1 insures that x_i and $\bar{x}_i$ take on distinct values in E_H. In other words, we are guaranteed that $\{x_i, \bar{x}_i\} = \{a, b\}$. If we set X_i to TRUE if and only if $x_i = a$ we get from Eq. (2) that not all three literals in a clause are TRUE (for we would then have $x_1x_2x_3 = a$) and similarly from Eq. (3), not all literals are FALSE.

Suppose now that $|E_H| = k \geq 3$. Using similar ideas, we can now obtain a reduction from k-COLORABILITY which is NP-complete for $k \geq 3$. For each vertex v_i in the graph, we create the variable x_i, force it to be an idempotent in B and for all edges e.g. (v_1, v_2) in the graph, add the equation

$$(4) \quad (x_1x_2)^\omega = e_H$$

Given a valid k-coloring of the graph, we obtain a solution to the system by identifying the k different colors with the k idempotents of E_H. Conversely, given a solution to the system, we color vertex v_i with the value x_i. We can assume that no vertex in the graph is isolated so that every variable x_i is involved in some equation of the form $(x_ix_j)^\omega = e_H$ and therefore lies in E_H. For two adjacent v_i, v_j the corresponding values x_i, x_j must be distinct for otherwise we get $x_ix_j = x_i \neq e_H$ in violation of Eq. (4). We therefore have a valid k-coloring of the graph. $\square$

Row-block 1	*	...	a*	b*	-	-	-	-
	*	*	*	*	-	-	-	-
Row-block 2	*	...	c*	d-				
	*	...	*	-				
	*	*	-					
	*	-						

Figure 5.1: Idempotents in B: the $*$'s (resp. $-$'s) mark $\mathcal{H}$-classes which contain (resp. do not contain) an idempotent.

Whereas this will allow us to handle the case where the regular semigroup S is inverse but not a union of groups, the next lemma will cover the case where S is not a union of groups and not inverse.

Lemma 5.41 *If S is a regular 0-simple semigroup but is not a union of groups, then* T-EQN$^*_S$ *is NP-complete.*

Proof. We can assume from the previous Lemma that S is not inverse. So the non-minimal $\mathcal{J}$-class, B has at least one $\mathcal{R}$-class (or $\mathcal{L}$-class) containing two idempotents. Our reduction will naturally exploit the location of idempotents in B. We represent this in Figure 5.1 as follows: the s $\mathcal{R}$-classes and t $\mathcal{L}$-classes can be represented in an s by t table where each cell is labeled with a $*$ if the corresponding $\mathcal{H}$-class contains an idempotent and labeled with $-$ otherwise.

We further reorder the rows and column of this table such that the first row contains a maximal number of $*$ (say k of them) and the first k cells in this row are labeled with $*$'s. The first n rows are then those equal to row 1, if any and we say that these form row-block 1. Next, we choose the $(n+1)^{\text{th}}$ row such that it has a maximal number $m < k$ of $*$'s occurring in its first k cells and reorder the columns such that these m cells occur first. Row-block 2 consists of all rows with $*$'s in their first m cells.

We now reduce from Monotone nae 3sat. For each literal X_i we create variables $x_i, \bar{x}_i$ and force them to be idempotent. We also impose for all $1 \leq$

$j \leq m$ the conditions

$$(1) \quad a_j x_i \in B \qquad\qquad (2) \quad a_j \bar{x}_i \in B$$

where a_j is the idempotent in the $\mathcal{H}$-class corresponding to the j^{th} cell of row 1. As we already observed, the constraint $z \in B$ can be imposed by the target-equation $u_1 z u_2 = b$ where b is any element of B. Note that these restrictions ensure that in any solution to the system, both variables x_i and $\bar{x}_i$ belong to one of the two row-blocks. Indeed, if the p^{th} cell (with $p < m$) in the row of x_i is labeled with $-$, we have $a_p x_i = 0$ since $\mathcal{R}_{x_i} \cap \mathcal{L}_{a_p}$ does not contain an idempotent.

Similarly, we use equations to impose

$$(3) \quad x_i a_1 \in J \qquad\qquad (4) \quad \bar{x}_i a_1 \in J.$$

This forces $x_i, \bar{x}_i$ to lie in $\mathcal{H}$-classes of B sitting in the first k columns.

Because $k \geq 2$, we can find a, b in row-block 1 and c, d in row-block 2 (also shown in Figure 5.1) such that a, b, c are idempotent with $ad = b$, $ba = c$, $ac = ba = a$ and $cd = db = cb = d$ but H_d contains no idempotent. Notice that for any element u of the first k cells in row-blocks 1 or 2, we have $(aua)^\omega = a$ because both $\mathcal{R}_a \cap \mathcal{L}_u$ and $\mathcal{L}_a \cap \mathcal{R}_u$ contain an idempotent. Similarly, the product of any number of idempotents lying in the first k cells of a same row-block also belong to this row-block.

We introduce the equation

$$(5) \quad (a x_i \bar{x}_i x_i a)^\omega = 0.$$

By our last remarks, this is not satisfied if both $x_i, \bar{x}_i$ are taken in the same row-block for $(a x_i \bar{x}_i x_i a)^\omega$ then equals a. It is satisfied however if $\{x_i, \bar{x}_i\} = \{b, c\}$. Finally, for every clause $X_1 \vee X_2 \vee X_3$ we add equations

$$(6) \quad (a x_1 x_2 x_3 a)^\omega = 0 \qquad\qquad (7) \quad (a \bar{x}_1 \bar{x}_2 \bar{x}_3 a)^\omega = 0.$$

Now, given an assignment to the X_i satisfying the MONOTONE NAE3SAT instance we can set $x_i = b$ and $\bar{x}_i = c$ when X_i is TRUE and $x_i = c$ and $\bar{x}_i = b$ when X_i is FALSE and easily verify that this constitutes a solution to the system constructed. Suppose conversely that we are given a solution to the system. All the $x_i, \bar{x}_i$ are idempotents in the first k cells of row-blocks 1 and 2 and if we set the literal X_i (resp. $\overline{X}_i$) to TRUE whenever x_i (resp. $\bar{x}_i$) belongs to row-block 1 and to FALSE when x_i (resp. $\bar{x}_i$) lies in row-block 2 we obtain a consistent truth assignment because of Eq. (5). Because Eqs. (6) and (7) are also satisfied it must be that for each clause $X_1 \vee X_2 \vee X_3$ the variables x_1, x_2, x_3 do not all lie in the same row-block. Similarly $\bar{x}_1, \bar{x}_2, \bar{x}_3$ do not all lie in the same row-block and so our assignment to the literals satisfies the MONOTONE NAE3SAT instance. □

The reductions presented in the last two lemmas can be slightly generalized to obtain:

Lemma 5.42 *If S is regular, then* T-EQN$^*_S$ *is NP-complete unless S is an orthodox union of Abelian groups.*

Proof. If S is a union of groups but is not orthodox then T-EQN$^*_S$ is NP-complete from Lemma 5.31. If S is regular but is not a union of groups, there exists some $\mathcal{J}$-minimal $\mathcal{J}$-class B that is not a union of groups and it suffices to show the NP-completeness of T-EQN$^*_T$ where T is the subsemigroup of elements lying $\mathcal{J}$-below this class. We distinguish two cases.

If B contains exactly one idempotent per $\mathcal{R}$- and $\mathcal{L}$-class and all $\mathcal{J}$-classes strictly below B are subgroups, we can use Lemma 5.40. Otherwise, the reduction in that proof must be slightly refined: we can still find a $\mathcal{J}$-maximal $\mathcal{J}$-class H such that there are two distinct idempotents in B with $xy \in H$. We can define E_H as before and still obtain that for any $x, y \in E_H$ both xy and yx lie in H. If e_H is an idempotent in H, however, we cannot assume that

$(xy)^\omega = e_H$ in this case. Still, it is easy to show that two idempotents x, y in B, belong to E_H and are distinct if and only if, for all $z \in E_H$, we have $(zxyz)^\omega = ze_Hz$. We can thus salvage our reduction, by replacing Eqs. (1,2,3) by $|E_H|$ different equations which will impose these constraints.

If B contains at least two idempotents in, say, some $\mathcal{R}$-class, we can reuse the reduction of Lemma 5.41. Now, we cannot anymore use the equation $(ax_i\bar{x}_ix_ia)^\omega = 0$. Note however, that we can replace it by

$$(ax_i\bar{x}_ia)^\omega(a\bar{x}_ix_ia)^\omega = d^\omega.$$

Indeed, this is still unsatisfied if both variables are taken from the same row block but is satisfied when $\{x_i, \bar{x}_i\} = \{b, c\}$. Similarly, we can replace the equations

$$(6)\quad (ax_1x_2x_3a)^\omega = 0 \qquad\qquad (7)\quad (a\bar{x}_1\bar{x}_2\bar{x}_3a)^\omega = 0.$$

by the equations

$$(6)\quad (ax_1x_2a)^\omega(ax_2x_1a)^\omega(ax_1x_3a)^\omega(ax_3x_1a)^\omega(ax_2x_3a)^\omega(ax_3x_2a)^\omega = d^\omega$$

and

$$(7)\quad (a\bar{x}_1\bar{x}_2a)^\omega(a\bar{x}_2\bar{x}_1a)^\omega(a\bar{x}_1\bar{x}_3a)^\omega(a\bar{x}_3\bar{x}_1a)^\omega(a\bar{x}_2\bar{x}_3a)^\omega(a\bar{x}_3\bar{x}_2a)^\omega = d^\omega.$$

□

Theorem 5.43 *For any regular semigroup S, we have* T-EQN$^*_S$ *lying in P if S belongs to* $\mathbf{RB} \vee \mathbf{Ab}$ *and* T-EQN$^*_S$ *is NP-complete otherwise.*

Proof. The proof is almost identical to the one of Theorem 5.39 for we have established that the NP-completeness of T-EQN$^*_S$ when S is regular but not a union of groups. □

We are so far unable to provide an equivalent dichotomy theorem in the case of EQN_S^* for regular semigroups. We define a *strong normal band of Abelian groups* to be a strong regular band of Abelian groups in which the idempotents form a normal band. We have established:

Lemma 5.44 *If S is regular but is not a strong normal band of Abelian groups then EQN_S^* is NP-complete.*

Proof. This is a slight refinement of half of Lemma 5.43: if S is not a strong regular band of Abelian groups then T-EQN_S^* is NP-complete. If it is, then $E(S)$ is not a normal band and we get NP-completeness from Corollary 5.33. □

In fact we conjecture:

Conjecture 5.45 *Let S be a regular semigroup. Then EQN_S^* is tractable if S is a strong normal band of Abelian groups and EQN_S^* is NP-complete otherwise.*

It seems that the only piece missing to complete this puzzle is a polynomial time algorithm to solve EQN_S^* for normal bands S because we can reasonably expect to extend such an algorithm to any strong normal band of Abelian group. It is quite easy to obtain an algorithm in the very special case where S is a free normal band on k generators and this has already been pointed out, in a different context, in [Klí03a]. As of yet, we are unable to extend these solutions to all normal bands.

Of course, many open questions remain concerning the complexity of EQN_S^* and T-EQN_S^* for non-regular S. The fact that answers to such questions will not be given by varieties should not be a pretext to dismiss such inquiries and should on the contrary be taken as added motivation for the problem. It is absolutely reasonable to assume that a simple necessary and sufficient condition for tractability can be formulated in these cases and our results thus far can already establish the NP-completeness of both problems in a large class of non-regular semigroups.

5.3 Conclusion

At first glance, we would expect questions concerning the complexity of solving single equations and systems of equations over a finite monoid or semigroup to be closely related. It is of course unreasonable to argue that they are not, yet the lessons learned from the two main sections of this chapter are very different. The case of single equations is closely related to questions about the power of finite monoids as language recognizers. It outlines once again the importance of cornering the computational power of CC^0 circuits and programs over solvable groups, and it further establishes the importance of **DA** and **DO** in complexity issues related to finite monoids.

Considering semigroups as machines on the other hand is a useless point of view in the case of systems of equations. Yet, many beautiful connections with previous algebraic approaches to constraint satisfaction problems have been uncovered. The fact that we can prove dichotomies for EQN^*_M, T-EQN^*_M and T-EQN^*_S for regular S indicates that these problems are "well-behaved" restrictions of CSP. We have mentioned some of the very general results from universal algebra identifying sufficient conditions on Γ for the tractability of CSP(Γ). It would be interesting to see if all our upper bounds can be obtained using these techniques and, if so, to understand whether the full gamut of them is needed in our context. For instance, there seems to be some similarity between the algorithms presented for EQN^*_S and T-EQN^*_S for, respectively, strong semilattices and strong regular bands of Abelian groups and the notion of para-primal algebras introduced of Dalmau [Dal00].

There might also be natural ways other than equations to define sets of relations Γ on S^k in terms of the algebraic structure of S. A similar direction in which to extend this research is to investigate the complexity of the satisfiability of inequations over a fixed ordered monoid. This is a very natural extension of our problem and it is quite possible that its structure will be as nice and rich of meaning.

Chapter 6

Conclusion

We have studied various computational complexity issues involving finite semigroups and monoids. We have first focused on the algebraic point of view on NC^1 emanating from the seminal work of D. Barrington and D. Thérien [Bar89, BT88]. We showed that some monoids are so weak that they cannot gain any advantage in computational power if polynomial length restrictions for the "program over monoid" formalism are relaxed and have given some evidence that any monoid failing to have this polynomial length property is rich enough to recognize arbitrary languages via programs of exponential length. We have also shown that programs over certain varieties of monoids are no more powerful than morphisms over that same variety.

Next, we have established a rich algebraic point of view on communication complexity. This has allowed us to algebraically characterize, up to a constant, the communication complexity of every regular language in some of the best-known two-party models, thus making a fundamental contribution to our understanding about the power of these models, their interrelations and the key role played by regular languages in the development of this theory. We have also proved a number of similar classifications in the multiparty "input on the forehead" model which has led us to isolate regular languages which quite certainly constitute key objects of study for further research on this model of communication complexity. Our results also suggest the possible existence of an algebraic characterization of languages having bounded multiparty communication complexity for some bounded number of players. Our communication complexity results can also be used to obtain new insights on the computational limitations of polynomial length programs over certain varieties of monoids, thus opening new paths to an eventual resolution of questions about the program model.

Finally, we have studied how the algebraic properties of a semigroup S impact the complexity of solving equations over S. We provided the first non-trivial upper bounds for checking the satisfiability of an equation over S_3 or, more generally, over any group which is too weak to efficiently compute AND via programs and established a number of upper bounds and hardness results for equation and program satisfiability over monoids which are not groups. We found stark dichotomies in the complexity of solving systems of equations over a fixed monoid and found the problem, and its main variant, to be either tractable or NP-complete depending on whether the monoid belonged to a specified variety. We also established a number of interesting partial results in the semigroup cases and argued that our dichotomy results were to be expected in light of the conjectures arising from the algebraic study of constraint satisfaction problems.

Our results highlight the importance of certain classes of monoids and semigroups in such contexts. Indeed, monoids in the variety $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$, for instance, are exactly the monoids in $\mathbf{DS}$ having the polynomial length property, are exactly the monoids with $O(1)$ communication complexity for some bounded number of players and are the only ones for which we know how to solve P-SAT in polynomial time. Other varieties such as solvable groups, $\mathcal{J}$-trivial monoids and specific monoids like U and B_2 have played key roles in different contexts both in this work and other similar investigations.

Traditionally, the analysis of problems whose complexity is parametrized by an underlying finite semigroup has been done separately for the group case and the aperiodic case but often left open in the general case. In many cases, the limit between tractable and intractable cases involve varieties $\mathbf{Ab}$, $\mathbf{G_{nil}}$ and $\mathbf{G_{sol}}$ in the group case and $\mathbf{DA}$ in the aperiodic case. Such a phenomenon occurs, for example, in the communication complexity and equation satisfiability settings but also in the context of membership problems [BMT92], learning expressions over monoids [GTT01] among others. Our work suggests that considering varieties of the form $\mathbf{DO} \cap \overline{\mathbf{H}}$ should be the first attempt at combining results for groups and aperiodics to resolve the general case.

We have of course left open a number of open questions concerning the main topics of this thesis and have discussed them in the relevant chapters but we want to recall here that many of these questions are deeply intertwined. For instance, our questions on the exact complexity of P-SAT$_G$ for non-nilpotent solvable groups can only be resolved if we are able to understand whether $\mathbf{G_{sol}}$ forms a program-variety or not.

If L is a language with neutral letter and bounded two-party communication complexity then, by Szegedy's Theorem, L can be recognized by a program over a commutative monoid. Since $\mathbf{Com}$ has the Crane Beach property, we must thus have that L is regular with $M(L)$ commutative. Similarly, if $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ has the Crane Beach property, then our proposed generalization of Szegedy's Theorem holds for languages with a neutral letter. On the other hand, if it does not have the Crane Beach property, then a counterexample of a non-regular language with neutral letter which can be recognized by a family of programs over some monoid in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ would certainly be an interesting candidate to disprove the communication complexity conjecture.

We believe that a most important avenue for research is to further understand the connections between the various contexts which we have analyzed with a semigroup algebra perspective. One has to believe that under favorable circumstances, the algebra of semigroups and monoids can constitute a bridge linking in meaningful ways issues in communication complexity, circuit complexity, logic, algebraic automata theory, to name but a few areas.

In this thesis, we have been dealing exclusively with languages of finite words. In some applications of finite automata, such as model checking, the focus is on so-called ω-languages of infinite words. There exists a well-developed algebraic theory of ω-regular languages quite similar to classical algebraic automata theory [PP03]. It would be most interesting to understand the impact of our results and the intuitions we have developed on this theory and its many applications.

Bibliography

[Ajt83] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[All97] E. Allender. Circuit complexity before the dawn of the new millennium. Technical Report 97-49, DIMACS, 1997.

[Amb96] A. Ambainis. Upper bounds on multiparty communication complexity of shifts. In *Proc. 13th Symp. on Theoretical Aspects of Comp. Sci.*, pages 631–642. 1996.

[Bar89] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. Preliminary version appeared in STOC'86.

[BDFP86] A. Borodin, D. Dolev, F. E. Fich, and W. Paul. Bounds for width two branching programs. *SIAM Journal on Computing*, 15(2):549–560, 1986.

[Bei93] R. Beigel. The polynomial method in circuit complexity. In *8th Annual Conf. on Structure in Complexity Theory (SCTC '93)*, pages 82–95. 1993.

[BFS86] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proc. 27th IEEE FOCS*, pages 337–347. 1986.

[BIL+01] D. A. M. Barrington, N. Immerman, C. Lautemann, N. Schweikardt, and D. Thérien. The Crane Beach conjecture. In *Proc. 16th Symp. on Logic in Comp. Sci. (LICS-01)*, pages 187–196. 2001.

[BJV02] A. Bulatov, P. Jeavons, and M. Volkov. Finite semigroups imposing tractable constraints. In G. Gomez, P. Silva, and J-E.Pin, editors, *Semigroups, Algorithms, Automata and Languages*. WSP, 2002.

[BK97] L. Babai and P. G. Kimmel. Randomized simultaneous messages. In *Proc. 12^{th} Conf. on Comp. Complexity (CCC '97)*, pages 239–247. 1997.

[BKJ00] A. Bulatov, A. Krokhin, and P. Jeavons. Constraint satisfaction problems and finite algebras. In *Proceedings 27th International Colloquium on Automata, Languages and Programming—ICALP'00*, volume 1853 of *Lecture Notes in Computer Science*, pages 272–282. 2000.

[BKLM01] D. Barrington, P. Kadau, K. Lange, and P. McKenzie. On the complexity of some problems on groups input as multiplication tables. *Journal of Computer and System Sciences*, 63:186–200, 2001.

[BLS87] L. Babai, E. Luks, and A. Seress. Permutation groups in NC. In *Proc. 19th Symp. on Theory of Computing*, pages 409–420. 1987.

[BMM$^+$00] D. M. Barrington, P. McKenzie, C. Moore, P. Tesson, and D. Thérien. Equation satisfiability and program satisfiability for finite monoids. In *Proc. Math. Foundations of Comp. Sci (MFCS'00)*, pages 172–181. 2000.

[BMPT97] M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien. Finite monoids: From word to circuit evaluation. *SIAM Journal on Computing*, 26(1):138–152, 1997.

[BMT92] M. Beaudry, P. McKenzie, and D. Thérien. The membership problem in aperiodic transformation monoids. *Journal of the ACM*, 39(3):599–616, 1992.

[BMT99] A. Baziramwabo, P. McKenzie, and D. Thérien. Modular temporal logic. In *Proc. 15th Conf. on Logic in Comp. Sci. (LICS'99)*. 1999.

[BNS92] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, October 1992.

[BPT01] P. Bouyer, A. Petit, and D. Thérien. An algebraic characterization of data and timed languages, 2001. Submitted to Information and Computation.

[BS94] D. A. M. Barrington and H. Straubing. Complex polynomials and circuit lower bounds for modular counting. *Computational Complexity*, 4(4):325–338, 1994.

[BS95] D. A. M. Barrington and H. Straubing. Superlinear lower bounds for bounded-width branching programs. *Journal of Computer and System Sciences*, 50(3):374–381, 1995.

[BS99] Barrington and Straubing. Lower bounds for modular counting by circuits with modular gates. *Computational Complexity*, 8(3):258–272, 1999.

[BSSV00] P. Beame, M. Saks, X. Sun, and E. Vee. Super-linear time-space tradeoff lower bounds for randomized computation. In *41st Symp. on Foundations of Comp. Sci. (FOCS'00)*. 2000.

[BST90] D. A. M. Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, 1990.

[BT88] D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35(4):941–952, October 1988.

[BT94] R. Beigel and J. Tarui. On ACC. *Computational Complexity*, 4(4):350–366, 1994.

[BTT02] D. A. M. Barrington, D. Thérien, and T. Tsukiji. On the constant-degree hypothesis, 2002. Document in preparation.

[Bul02] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. of 43rd Foundations of Comp. Sci. (FOCS'02)*, pages 649–658. 2002.

[BV02] P. Beame and E. Vee. Time-space tradeoffs multiparty communication complexity and nearest neighbor problems. In *34th Symp. on Theory of Computing (STOC'02)*, pages 688–697. 2002.

[Cau96] H. Caussinus. A note on a theorem of Barrington, Straubing and Thérien. *Information Processing Letters*, 58(1):31–33, 1996.

[CFL83] A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *Proc. 15^{th} ACM STOC*, pages 94–99. 1983.

[CG85] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *Proc. 26th IEEE FOCS*, pages 429–442. 1985.

[Chu90] F. Chung. Quasi-random classes of hypergraphs. *Random Structures and Algorithms*, 1(4):363–382, 1990.

[CT93] F. Chung and P. Tetali. Communication complexity and quasi-randomness. *SIAM J. Discrete Math.*, 6(1):110–123, 1993.

[CvDNT99] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. *Lecture Notes in Computer Science*, 1509:61–74, 1999.

[Dal00] V. Dalmau. *Computational Complexity of Problems over Generalized Formula.* Ph.D. thesis, Universita Politécnica de Catalunya, 2000.

[DF89] D. Dolev and T. Feder. Multiparty communication complexity. In *Proc. 30^th IEEE FOCS*, pages 428–433. 30 October–1 November 1989.

[DF97] R. G. Downey and M. R. Fellows. *Parametrized Complexity*. Monographs in Computer Science. Springer, 1997.

[DKMW92] C. Damm, M. Krause, C. Meinel, and S. Waack. On relations between counting communication complexity classes. In *9th Annual Symp. on Theor. Asp. of Comp. Sci. (STACS'92)*. 1992. Full version to appear in Journal of Computer and Systems Sciences. Currently available e.g. from www.num.math.uni-goettingen.de/damm/.

[Eil76] S. Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, 1976.

[FKPS85] R. Fagin, M. M. Klawe, N. J. Pippenger, and L. Stockmeyer. Bounded-depth, polynomial-size circuits for symmetric functions. *Theoretical Computer Science*, 36(2-3):239–250, 1985.

[FSS84] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. Preliminary version appeared in FOCS'81.

[FV99] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. on Computing*, 28(1):57–104, 1999.

[Gas02a] W. Gasarch. Notes available from the author's web site., 2002.

[Gas02b] W. Gasarch. Complexity theory column 36: the P=?NP poll. *SIGACT News*, 33(2):34–47, 2002. Column edited by L. Hemaspaandra.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979.

[GLM96] J. Goldsmith, M. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT News*, 27, 1996.

[GP89] J. Gerhard and M. Petrich. Varieties of bands revisited. *Proc. of London Math. Soc.*, 58(3):323–350, 1989.

[GR99] M. Goldmann and A. Russell. The complexity of solving equations over finite groups. In *Proc. 14th Conf. on Computational Complexity*, pages 80–86. 1999.

[Gra68] R. Graham. On finite 0-simple semigroups and graph theory. *Math. Systems Theory*, 2:325–339, 1968.

[Gro92] V. Grolmusz. Separating the communication complexities of MOD m and MOD p circuits. In *Proc. 33rd IEEE FOCS*, pages 278–287. 1992.

[Gro93] V. Grolmusz. On multi-party communication complexity of random functions. Technical Report MPII-1993-162, Max Planck Institut für Informatik, December 1993.

[Gro94a] V. Grolmusz. The BNS lower bound for multi-party protocols in nearly optimal. *Information and Computation*, 112(1):51–54, 1994.

[Gro94b] V. Grolmusz. A weight-size trade-off for circuits and MOD m gates. In *Proc. 26^{th} ACM STOC*, pages 68–74. 1994.

[Gro97] V. Grolmusz. On the power of circuits with gates of low L_1 norms. *Theoretical Computer Science*, 188(1–2):117–128, 30 November 1997.

[Gro98] V. Grolmusz. A degree-decreasing lemma for MOD_p-MOD_m circuits. *Lecture Notes in Computer Science (ICALP'98)*, 1443:215–222, 1998.

[GRS80] R. L. Graham, B. L. Rotschild, and J. H. Spencer. *Ramsey Theorey*. Series in Discrete Mathematics. Wiley Interscience, 1980.

[GRT00] M. Goldmann, A. Russell, and D. Thérien. An ergodic theorem for read-once non-uniform deterministic finite automata. *Information Processing Letters*, 73(1–2):23–28, 2000.

[GT00] V. Grolmusz and G. Tardos. Lower bounds for (MODp − MODm) circuits. *SIAM Journal on Computing*, 29(4):1209–1222, 2000.

[GT03] R. Gavaldà and D. Thérien. Algebraic characterizations of small classes of boolean functions. In *Proc. of Symp. on Theoretical Aspects of Comp. Sci.' 03*. 2003.

[GTT01] R. Gavaldà, P. Tesson, and D. Thérien. Learning expressions and programs over monoids, 2001. Submitted to Information and Computation. Extended abstract appears in Proc. STACS 2001. Available from www.lsi.upc.es/ gavalda/papers.html.

[Gut00] C. Gutiérrez. Satisfiability of equations in free groups is in PSPACE. In ACM, editor, *Proceedings of the thirty second annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 21–27. 2000.

[HAB02] W. Hesse, E. Allender, and D. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65, 2002.

[Hås87] J. Håstad. *Computational Limitations for Small Depth Circuits*. MIT Press, Cambridge, MA., 1987.

[HG90] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. In *Proc. 31st IEEE FOCS*, pages 610–618. 1990.

[HLS+93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. Wagner. On the power of polynomial time bit-reductions. In *Conf. on Structure in Complexity Theory*. 1993.

[How76] J. Howie. *An Introduction to Semigroup Theory*. Academic Press, 1976.

[HR90] B. Halstenberg and R. Reischuk. Relations between communication complexity classes. *J. of Computer and System Sciences*, 41:402–429, 1990.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[HWWY94] J. Håstad, I. Wegener, N. Wurm, and S.-Z. Yi. Optimal depth, very small size circuits for symmetric functions in AC^0. *Information and Computation*, 108(2):200–211, 1994.

[JCG97] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[Juk01] S. Jukna. *Extremal combinatorics with applications in computer science*. Texts in theoretical computer science. Springer, 2001.

[Klí02] O. Klíma. Unification modulo associativity and idempotency is NP-complete. In *MFCS: Symposium on Mathematical Foundations of Computer Science*. 2002.

[Klí03a] O. Klíma. Complexity of unification and matching problems in the varieties of idempotent semigroups, 2003. To appear in *Int. J. of Algebra and Computation*.

[Klí03b] O. Klíma. On the complexity of checking identities in finite monoids, 2003. Preprint.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KNR99] Kremer, Nisan, and Ron. On randomized one-round communication complexity. *Computational Complexity*, 8:21–49, 1999.

[Koz77] D. Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science*, pages 254–266. IEEE, 1977.

[KR65] K. Krohn and J. Rhodes. The algebraic theory of machines I. *Trans. Amer. Math. Soc.*, 116:450–464, 1965.

[KS92] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. on Discrete Mathematics*, 5(4):545–557, 1992.

[KS00] O. Klíma and J. Srba. Matching modulo associativity and idempotency is NP-complete. In *MFCS: Symposium on Mathematical Foundations of Computer Science*. 2000.

[KTT03] O. Klíma, P. Tesson, and D. Thérien. Dichotomies in the complexity of solving systems of equations over finite semigroups, 2003. Document in preparation.

[KW88] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proc. 20^{th} ACM STOC*, pages 539–550. 1988.

[Lok01] S. Lokam. Spectral methods for matrix rigidity with applications to size-depth trade-offs and communication complexity. *JCSS: Journal of Computer and System Sciences*, 63:449–473, 2001.

[Lov89] L. Lovász. Communication complexity: a survey. Technical Report CS-TR-204-89, Princeton University, 1989.

[LT01] C. Lautemann and D. Thérien, 2001. Private communication.

[MNSW98] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.

[MPT91] P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.

[MPT00] A. Maciel, P. Péladeau, and D. Thérien. Programs over semigroups of dot-depth one. *Theoretical Computer Science*, 245(1):135–148, 2000.

[MR65] W. Maurer and J. Rhodes. A property of finite simple non-Abelian groups. *Proc. Amer. Math. Soc*, 16:552–554, 1965.

[MTT01] C. Moore, P. Tesson, and D. Thérien. Satisfiability of systems of equations over finite monoids. In *MFCS'01*, pages 537–547. 2001.

[Nis94] N. Nisan. The communication complexity of treshold gates, 1994. (Available from http://www.cs.huji.ac.il/ñoam/papers.html).

[NS96] I. Newman and M. Szegedy. Public vs. private coin flips in one round communication games (extended abstract). In *Proc. of 28th ACM Symp. on Theory of Comp. (STOC '96)*, pages 561–570. 1996.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing, 1994.

[Pin86] J.-E. Pin. *Varieties of formal languages*. North Oxford Academic Publishers Ltd, London, 1986.

[Pin95] J.-É. Pin. $PG = BG$, a success story. In J. Fountain, editor, *NATO Advanced Study Institute Semigroups, Formal Languages and Groups*, pages 33–47. Kluwer academic publishers, 1995.

[Pin97] J.-E. Pin. Syntactic semigroups. In G. R. et A. Salomaa, editor, *Handbook of language theory*, volume 1, chapter 10, pages 679–746. Springer Verlag, 1997.

[PP03] D. Perrin and J. Pin. Infinite words. Book in preparation, 2003. Available from the author's web page.

[PST88] J.-É. Pin, H. Straubing, and D. Thérien. Locally trivial categories and unambiguous concatenation. *J. Pure Applied Algebra*, 52:297–311, 1988.

[PT88] P. Péladeau and D. Thérien. Sur les langages reconnus par des groupes nilpotents. *C.R. Acad. des Sci. Paris Sér. I Math.*, 306(2):93–95, 1988. English translation by A. Russell and S. Russell appears as TR01-040 of ECCC.

[Pud03] P. Pudlák. An application of Hindman's theorem to a problem on communication complexity, 2003. To appear in *Combinatorics, Probability and Computing*.

[PW96] J. E. Pin and P. Weil. Profinite semigroups, Mal'cev products, and identities. *Journal of Algebra*, 182:604–626, 1996.

[Raz87] A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *MATHNASUSSR: Mathematical Notes of the Academy of Sciences of the USSR*, 41, 1987.

[Raz92] A. Razborov. On the distributed complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 14 December 1992. Note.

[Raz00] R. Raz. The BNS-Chung criterion for multi-party communication complexity. *Computational Complexity*, 9(2):113–122, 2000.

[Rhe01] K. Rheinhardt. Upper bounds for the bicycle. Private Communication, 2001.

[RM97] R. Raz and P. McKenzie. Separation of the monotone NC hierarchy. In *Proc. 38^{th}th IEEE FOCS*. 1997.

[ROKY94] V. Roychowdhury, A. Orlitsky, and K.-Y.Siu. Lower bounds on threshold and related circuits via communication complexity. *IEEETIT: IEEE Transactions on Information Theory*, 40, 1994.

[RR97] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[RT89] J. Rhodes and B. Tilson. The kernel of monoid morphisms. *J. Pure and Applied Algebra*, 62:227–268, 1989.

[RTT98] J.-F. Raymond, P. Tesson, and D. Thérien. An algebraic approach to communication complexity. *Lecture Notes in Computer Science (ICALP'98)*, 1443:29–40, 1998.

[Sch65] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, April 1965.

[Sch76] M. P. Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13:47–75, 1976.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. 10^{th} ACM STOC*, pages 216–226. 1978.

[Sim75] I. Simon. Piecewise testable events. In *Proc. 2nd GI Conf.*, pages 214–222. 1975.

[Sip97] M. Sipser. *Introduction to the Theory of Computation*. PWS, 1997.

[Smo86] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19^{th} ACM STOC*, pages 77–82. 1986.

[ST] H. Straubing and D. Thérien. A note on MOD_p-MOD_m-circuits. Submitted to Theoretical Computer Science.

[ST01] H. Straubing and D. Thérien. Regular languages defined by generalized first-order formulas with a bounded number of bound variables. In *Proc. of 18th Symp. on Theoretical Aspects of Comp. Sci. conference*, pages 551–562. 2001.

[ST02] H. Straubing and D. Thérien. Weakly iterated block products of finite monoids. In *Proc. of the 5th Latin American Theoretical Informatics Conference (LATIN '02)*. 2002.

[Str94] H. Straubing. *Finite Automata, Formal Logic and Circuit Complexity*. Boston: Birkhauser, 1994.

[Str00] H. Straubing. When can one monoid simulate another? In *Algorithmic Problems in Groups and Semigroups*, pages 267–288. Birkhäuser, 2000.

[Str01] H. Straubing. Languages defined by modular quantifiers. *Information and Computation*, 166:112–132, 2001.

[Sze93] M. Szegedy. Functions with bounded symmetric communication complexity, programs over commutative monoids, and ACC. *Journal of Computer and System Sciences*, 47(3):405–423, 1993.

[Tes99] P. Tesson. *An algebraic approach to communication complexity*. Master's thesis, School of Computer Science, McGill University, 1999.

[Thé79] D. Thérien. Languages of nilpotent and solvable groups (extended abstract). In *Automata, Languages and Programming, 6th Colloquium*, volume 71 of *Lecture Notes in Computer Science*, pages 616–632. Springer-Verlag, 1979.

[Thé83] D. Thérien. Subword counting and nilpotent groups. In L. Cummings, editor, *Combinatorics on Words: Progress and Perspectives*, pages 195–208. Academic Press, 1983.

[Thé89] D. Thérien. Programs over aperiodic monoids. *Theoretical Computer Science*, 64(3):271–280, 29 May 1989.

[Thé94] D. Thérien. Circuits constructed with MOD_q gates cannot compute AND in sublinear size. *Computational Complexity*, 4:383–388, 1994.

[TS99] Ta-Shma. Classical versus quantum communication complexity. *SIGACT News*, 30, 1999.

[TT02a] P. Tesson and D. Thérien. The computing power of programs over finite monoids. *Journal of Automata, Languages and Combinatorics*, 7(2):247–258, 2002.

[TT02b] P. Tesson and D. Thérien. Diamonds are forever: the variety **DA**. In G. Gomez, P. Silva, and J-E.Pin, editors, *Semigroups, Algorithms, Automata and Languages*. WSP, 2002.

[TT03] P. Tesson and D. Thérien. Complete classifications for the communication complexity of regular languages. In *Proc. Symp. on Theoretical Aspects of Comp. Sci. 2003*. 2003.

[TW98] D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation. In *Proc. 30th ACM Symposium on the Theory of Computing*, pages 256–263. 1998.

[TW02a] D. Thérien and T. Wilke. Nesting until and since in linear temporal logic, 2002. Symp. on Theoretical Aspects of Comp. Sci. (STACS'02).

[TW02b] D. Thérien and T. Wilke. Temporal logic and semidirect products: An effective characterization of the until hierarchy. *SICOMP: SIAM Journal on Computing*, 31, 2002.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Texts in theoretical computer science. Springer, 1999.

[Weg00] I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[Wei92] P. Weil. Closure of varieties of languages under products with counter. *J. Comput. Syst. Sci.*, 45:316–339, 1992.

[Yao79] A. C. Yao. Some complexity questions related to distributive computing. In *Proc. 11th ACM STOC*, pages 209–213. 1979.

[Yao90] A. C.-C. Yao. ON ACC and threshold circuits. In *Proc. of 31st IEEE FOCS*, volume 2, pages 619–627. 1990.

[Yao93] A. C.-C. Yao. Quantum circuit complexity. In *Proc. of 34th IEEE FOCS*, pages 352–361. 1993.

Index

Index of Symbols and Notation

Problems and Languages

$CSP(\Gamma)$ Constraint Satisfiability Problem (with relations Γ).

$DISJ$ Disjointness.

EQ Equality.

(T)-EQN (Target)-Equation Satisfiability.

(T)-EQN* System of (Target)-Equations Satisfiability.

$GIP_{k,q}$ k-wise Generalized Inner Product $\pmod q$.

GT Greater Than.

$INDEX$ Index.

IP_q Inner Product $\pmod q$.

P-SAT Program Satisfiability.

Varieties

$\mathbf{A}$ Aperiodic semigroups.

$\mathbf{A_1}$ Bands.

$\mathbf{A_{com}}$ Commutative aperiodics.

$\mathbf{Ab}$ Abelian groups.

$\mathbf{Com}$ Commutative semigroups.

$\mathbf{DV}$ Regular $\mathcal{J}$-classes lie in $\mathbf{V}$.

$\mathbf{G_p}$ p-groups.

$\mathbf{G_{nil}}$, $\mathbf{G_{nil,k}}$ Nilpotent groups (of class k).

$\mathbf{G_{sol}}$ Solvable groups.

$\overline{\mathbf{H}}$	Semigroups with subgroups in $\mathbf{H}$.
$\mathbf{I}$	Trivial variety.
$\mathbf{J_1}$	Semilattices.
$\mathbf{L_1}$	$\mathcal{L}$-trivial bands.
$\mathbf{LV}$	Semigroups with local monoids in $\mathbf{V}$.
$\mathbf{NB}$	Normal bands.
$\mathbf{R_1}$	$\mathcal{R}$-trivial bands.
$\mathbf{RB}$	Regular bands.
$\mathbf{UG}$	Unions of groups.
$\mathbf{V} \square \mathbf{W}$	Block product.
$\mathbf{V} Ⓜ \mathbf{W}$	Mal'cev product.
$\mathbf{V} * \mathbf{W}$	Wreath product.
$\mathbf{W}$	Monoids with $R(M) = \Theta(\log\log n)$.

Other Symbols

AC^0	Bounded depth polynomial size AND,OR circuits.
ACC^0	Bounded depth polynomial size AND,OR,MOD$_m$ circuits.
CC^0	Bounded depth polynomial size MOD$_m$ circuits.
$D(L)$	Deterministic communication complexity.
$D^{\|\|}(L)$	Simultaneous communication complexity.
$M_pPol(\mathcal{L})$	Mod$_p$-counter closure.
$N^1(L)$	non-deterministic communication complexity.
$N^{Mod_p}(L)$	Mod$_p$-counting communication complexity.
NC^1	Log depth AND,OR circuits with bounded fan-in.
$\mathcal{P}(\mathbf{V})$	Languages recognized via polylength $\mathbf{V}$-programs.
$R(L)$	Probabilistic communication complexity.
$R^{\|\|}(L)$	Simultaneous probabilistic communication complexity.
$UPol(\mathcal{L})$	Unambiguous polynomial closure.

$\equiv_L$	Syntactic congruence.
$\leq_{\mathcal{J}}$	$\mathcal{J}$-preorder (similarly for $\mathcal{R}$, $\mathcal{L}$).
$\approx^G$	Congruence parametrizing subclasses of **DO**.