DIAMONDS ARE FOREVER: THE VARIETY DA

PASCAL TESSON AND DENIS THÉRIEN
School of Computer Science, McGill University,
3480 rue University, Montréal, Québec,
H3A 2A7, Canada.
E-mail: {ptesso, denis}@ cs.mcgill.ca.

We survey different characterizations (algebraic, combinatorial, automata-theoretic, logical) of the variety **DA**, the class of monoids whose regular $\mathcal{D}$-classes form aperiodic semigroups. We study the interconnections of such characterizations and outline the importance of this variety in various computational complexity issues.

1 Introduction

The intuition that combinatorial properties of a language should relate to the algebraic structure of its syntactic congruence has been completely validated in the realm of rational languages, i.e. in the case where the syntactic congruence has finite index. Indeed the theory of finite monoids offers an elegant and powerful framework to understand rational languages, and in turn the language approach has proved to be very useful in the study of finite monoids.

A subset $L \subseteq A^*$ is *recognized* by a monoid M if there exists a morphism $\phi : A^* \to M$ and a subset F of M such that $L = \phi^{-1}(F)$. It is easy to show that for any language L, there is a "minimal" recognizer $M(L)$: it has the property that it *divides* (i.e. is a morphic image of a submonoid of) any other monoid M recognizing L. The famous theorem of Kleene states that L is a rational language if and only if $M(L)$ is finite.

The notion of pseudo-variety, which we will simply call variety in this paper, has become central in algebraic automata theory. A class $\mathbf{V}$ of finite monoids forms a variety if and only if it is closed under division and direct product. Certain varieties are real "jewels", in the sense that they admit several different characterizations. Unsurprisingly, these varieties usually turn out to be particularly important. This is certainly the case for the variety $\mathbf{A}$ of aperiodic (or group-free) monoids: a monoid M is group-free if no subset of it forms a group of cardinality greater than one. For example, the following equivalences can be shown:

- $M(L)$ is aperiodic;

- L is a star-free language [Sch65];
- L can be defined by a first-order sentence [MP71] (see Section 5);
- L can be defined by a formula in linear temporal logic [Kam68] (see Section 5);
- L can be recognized by a cascade product of set-reset automata [KR65].

Note that the situation with the variety **G** of finite groups is far from being as satisfactory, essentially because we still understand very little about simple non-Abelian groups.

In recent years, the variety **DA**, a proper subclass of aperiodic monoids, has emerged as a "jewel" in its own right and several beautiful characterizations are now known for it. This paper offers a survey of the most important ones.

For conciseness, we will often refer the reader to the bibliography for additional details. We will first review some basic results, then present congruence relations offering parametrizations of **DA** and algebraic decompositions and parametrizations of the variety. Finally, we will discuss applications of these characterizations to logic and descriptive complexity and to computational complexity related issues.

2 Basic Properties and Notation

All monoids considered in this paper are either free or finite. We denote by $eval_M$ the canonical morphism from M^* into M defined by

$$eval_M(m_1 \dots m_n) = m_1 \cdot \dots \cdot m_n$$

and denote by m^ω the unique positive power of m that is an idempotent. Let us recall the fundamental Green equivalences:

- $s\mathcal{R}t$ iff $sM = tM$;
- $s\mathcal{L}t$ iff $Ms = Mt$;
- $s\mathcal{D}t$ iff $s\mathcal{R} \vee \mathcal{L}t$;
- $s\mathcal{J}t$ iff $MsM = MtM$;
- $s\mathcal{H}t$ iff $s\mathcal{R}t$ and $s\mathcal{L}t$.

We also write $s \leq_{\mathcal{R}} t$ if $sM \subseteq tM$ and similarly for the other relations. We also denote by $\mathcal{R}_m$ the $\mathcal{R}$-class of m. For any of the above relations, we say that a given class is regular if and only if it contains an idempotent.

The following facts are elementary:

Lemma 1 *a) In a finite monoid, $\mathcal{D}$ and $\mathcal{J}$ coincide.*
b) If s and t belong to the same $\mathcal{J}$-class, then their product st belongs to $\mathcal{R}_s \cap \mathcal{L}_t$ iff $\mathcal{L}_s \cap \mathcal{R}_t$ contains an idempotent.

The variety **DA** is defined as the class of finite monoids in which every regular $\mathcal{D}$-class forms an aperiodic semigroup. By the lemma above, it follows that M is in **DA** iff $MeM = MfM$ and $e = e^2$ imply $f = f^2$.

The following alternative characterizations are particularly useful.

Theorem 2 *The following are equivalent:*
a) M is in **DA***;*
b) $e = e^2$ and $MeM \subseteq MsM$ imply $ese = e$;
c) for all x, y, z in M, $(xyz)^\omega y (xyz)^\omega = (xyz)^\omega$.

Proof: a) $\Rightarrow$ b) Let $e = usv$: since e is idempotent, we have $e = (usv)^\omega$, which is $\mathcal{J}$-related to $f = (svu)^\omega$. Because M is in **DA**, we have $e\mathcal{J}ef$, so that $e\mathcal{R}es$ and $es = eses$. We conclude that $ese = e$.
b) $\Leftrightarrow$ c) Obvious.
b) $\Rightarrow$ a) If $e = e^2\mathcal{J}f$, then $efe = e$. Thus ef is an idempotent, and $effef = ef$: it must be that f^2 is also in the same $\mathcal{J}$-class, hence f is an idempotent. ∎

If M is not in **DA**, then it contains a regular $\mathcal{J}$-class that has a non-idempotent. Conveniently, this can be used to show that M either contains a group or is divided by one of the aperiodics BA_2 or U, the transformation monoids associated with the automata of Figures 1 and 2 respectively.

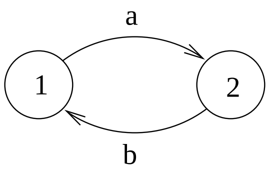

Fig. 1: BA_2 automaton

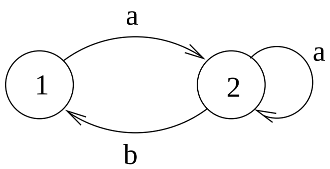

Fig. 2: U automaton

Throughout this paper, varieties of monoids will be denoted in **boldface**. We'll pay a particular attention to **DA** of course, but will also mention among others the variety of aperiodics **A**, commutative aperiodics $\mathbf{A_{com}}$, idempotent monoids or *bands* $\mathbf{A_1}$, $\mathcal{R}$-trivial monoids **R**, $\mathcal{R}$-trivial bands $\mathbf{R_1}$, $\mathcal{L}$-trivial monoids **L**, $\mathcal{L}$-trivial bands $\mathbf{L_1}$ and solvable groups $\mathbf{G_{sol}}$. Suppose that for every finite alphabet A we have a congruence relation $\sim_A$ over strings in A^*, we will denote by $\mathbf{V}_\sim$ the class of monoids

$$\mathbf{V}_\sim = \{M = A^*/\gamma : \text{ for some } A \text{ with } \sim_A \subseteq \gamma\}.$$

3 Congruences

In this section, we describe various families of congruences, each of which induces a parametrization of the variety **DA**. Let w be a string in A^*. We denote by $\alpha(w)$ the set of letters of A occurring in w and for any $a \in \alpha(w)$, we say that $w = w_0 a w_1$ is the a-left decomposition of w (resp. a-right decomposition of w) if $a \notin \alpha(w_0)$ (resp. $a \notin \alpha(w_1)$).

3.1 A Characterization of **DA** *via Congruences*

We define a first congruence $\sim_{n,k}$ on A^* where $n = |A|$ by induction on $n+k$. First, we have $x \sim_{n,0} y$ for all $x, y \in A^*$. Next, we let $x \sim_{n,k} y$ when and only when:

1. $\alpha(x) = \alpha(y)$;
2. 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 $x_0 \sim_{n-1,k} y_0$ and $x_1 \sim_{n,k-1} y_1$;
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-right decompositions of x and y then $x_0 \sim_{n,k-1} y_0$ and $x_1 \sim_{n-1,k} y_1$.

This equivalence relation is well-defined since $|\alpha(x_0)| < |\alpha(x)|$ in (2) and $|\alpha(x_1)| < |\alpha(x)|$ in (3). It is easy to check that $\sim_{n,k}$ is in fact a congruence of finite index.

Theorem 3 ([TW98]) *Let $M = A^*/\gamma$, with $|A| = n$. Then $M \in$* **DA** *iff $\sim_{n,k} \subseteq \gamma$ for some k.*

Proof: For one direction, it suffices to show that $A^*/\sim_{n,k}$ is in **DA**. This will follow if we show that $u \sim_{n,k} v$ whenever u has the form $u = (xyz)^k y (xyz)^k$ and v has the form $v = (xyz)^k$. If $k = 0$, there is nothing to argue. If $k > 0$, we clearly have $\alpha(u) = \alpha(v)$. Let a be a letter in $\alpha(u)$, and let $u = u_0 a u_1$ and $v = v_0 a v_1$ be the a-left decompositions of u and v respectively. Then $u_0 = v_0$: also $u_1 = w(xyz)^{k-1} y (xyz)^k$ and $v_1 = w(xyz)^{k-1}$. Using induction, we see that $(xyz)^{k-1} \sim_{n,k-1} (xyz)^k$ and that $(xyz)^{k-1} yxyz^{k-1} \sim_{n,k-1} (xyz)^{k-1}$, so that $u_1 \sim_{n,k-1} v_1$. By left-right symmetry, we deduce that $u \sim_{n,k} v$.

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 $u = u_0 a_1 u_1 \dots a_t u_t$, with $a_i \in A$ and $u_i \in A^*$ such that:

1. $[u_0 a_1 \dots a_s u_s] \not\mathcal{R} [u_0 a_1 \dots a_s u_s a_{s+1}]$ for any $s < t$;
2. $[u_0 a_1 \dots a_s] \mathcal{R} [u_0 a_1 \dots a_s u_s]$ for any $s \leq t$;
3. $[u_0] = 1_M$.

It is not hard to see that in the $\mathcal{R}$-decomposition of u, one must have $a_i \notin \alpha(u_{i-1})$. In particular, $u = u_0\, a_1\, u_1 a_2 \dots a_t u_t$ 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 subalphabet with $|B| = m$ and suppose u, v are strings in B^*. We claim that if $u \sim_{m,mk} v$ then $[u] = [v]$ 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 $u \sim_{m,mk} v$, 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. We have $t \leq k$. Write w_i for $u_i a_{i+1} \dots a_t u_t$. By our earlier remark, $w_i = u_i a_{i+1} w_{i+1}$ 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 $u_i \sim_{m-1,mk-i} v_i$ for $i < t$ (Note that this actually implies $u_i \sim_{m-1,(m-1)k} v_i$). 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 $[u]\mathcal{R}[v]$. The same argument using $\mathcal{L}$-decompositions establishes $[u]\mathcal{L}[v]$ so, by aperiodicity, we have in fact $[u] = [v]$. ∎

Corresponding to the definition of $\sim$, we can recursively define $*_{n,k}$ languages in the following way. First, we let A^* be a $*_{|A|,0}$ language for any A. Next, suppose A_0, A_1 are finite alphabets whose difference is a single letter $a \in A_1 - A_0$ and let $L_0 \subseteq A_0^*$, $L_1 \subseteq A_1^*$ be $*_{|A_0|,k}$- and $*_{|A_1|,k}$-languages respectively, then $L_0 a L_1$ and $L_1 a L_0$ are both $*_{|A_1|,k+1}$ languages. From Theorem 3, we can immediately conclude:

Corollary 4 *Let $L \subseteq A^*$. We have $M(L) \in$ **DA** if and only if L is a (possibly empty) disjoint union of $*_{n,k}$ languages.*

For any languages $L_0, \dots, L_t \subseteq A^*$ and letters $a_1, \dots, a_t \in A$, we say that the language $K = L_0 a_1 L_1 \dots a_t L_t$ is an *unambiguous concatenation* of the L_i's if for any $w \in K$ there exists a unique factorization $w = w_0 a_1 w_1 \dots a_t w_t$ with $w_i \in L_i$. For such languages, we refer to the a_i's as *bookmarks*. The results of this subsection provide an alternative proof to a Theorem due to Schützenberger [Sch76]:

Theorem 5 *Let $L \subseteq A^*$. We have $M(L) \in \mathbf{DA}$ if and only if L is the disjoint union of unambiguous concatenations $K = A_0^* a_1 A_1^* \ldots a_t A_t^*$, with $a_i \in A$ and $A_i \subseteq A$.*

Proof: The $*_{n,k}$ languages are examples of such unambiguous concatenations so the "only if" part of the statement can be obtained from Corollary 4.

For the converse, we argue by induction on t that an unambiguous concatenation $K = A_0^* a_1 A_1^* \ldots a_t A_t^*$ is a union of $*_{|A|,t}$ languages. The case $t = 0$ is trivial. For $t \geq 1$, observe that by the unambiguity restriction, it can not be the case that all a_i's belong to the intersection $A_0 \cap A_t$ (or else, the string $(a_1 \ldots a_t)^2$ would admit two different factorizations). Suppose without loss of generality that $a_i \notin A_0$ and consider for any $w \in K$ the a_i-left-decomposition of $w = u a_i v$. Because of unambiguity, exactly one of the following must occur: $u \in K_0 = A_0^* a_1 \ldots a_{i-1} A_{i-1}^*$ while $v \in K_1 = A_i^* a_{i+1} \ldots a_t A_t^*$ or, for some $j < i$ we have $a_i \in A_j$ and $u \in L_j = A_0^* a_1 \ldots a_j A_j^*$ while $v \in L'_j = A_j^* a_{j+1} \ldots a_t A_t^*$. In either case, the languages K_0 and L_j are unambiguous concatenations with at most $t-1$ bookmarks over $(A - \{a_i\})^*$ which, by induction, are disjoint unions of $*_{|A|-1,t-1}$ languages while the languages K_1 and L'_j are unambiguous concatenations with at most $t-1$ bookmarks over A^* and are thus unions of $*_{|A|,t-1}$ languages. ∎

3.2 An Alternative Parametrization of **DA**

Let $|w|_a$ denote the number of occurences of the letter a in the word w and let $\alpha_t(w)$ be the vector of dimension $|A|$ giving, for each $a \in A$, the value $|w|_a$ up to the threshold t. We now define a second family of congruences $\approx_{k,t}$ on A^* for any $k, t \in \mathbb{N}$. First, for any t, we let $x \approx_{0,t} y$ for all $x, y \in A^*$. Then recursively, we define $x \approx_{k,t} y$ if and only if

1. $x \approx_{k-1,t} y$;
2. $\alpha_t(x) = \alpha_t(y)$;
3. For all $x = x_0 a x_1$ and $y = y_0 a y_1$ with $|x_0|_a = |y_0|_a \leq t$, we have $x_0 \approx_{k-1,t} y_0$ and $x_1 \approx_{k-1,t} y_1$;
4. For all $x = x_0 a x_1$ and $y = y_0 a y_1$ with $|x_1|_a = |y_1|_a \leq t$, we have $x_0 \approx_{k-1,t} y_0$ and $x_1 \approx_{k-1,t} y_1$.

One can check that for all k, t, $\approx_{k,t}$ is a well defined finite index congruence. This congruence is quite close to $\sim$ defined in the previous subsection: where in $\sim$ 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 $\approx_{k,1}$ is *not* equal to $\sim_{|A|,k}$ because the recursion on the prefixes and suffixes is slightly different in the two definitions. For example, one can verify that $ab \approx_{1,1} ba$ while $ab \not\approx_{2,1} ba$.

Not so surprisingly, given the similarity to $\sim$, we can obtain from $\approx_{k,t}$ an alternative parametrization of **DA**:

Theorem 6 *Let $M = A^*/\gamma$, with $|A| = n$. Then $M \in$ **DA** iff $\approx_{k,t} \subseteq \gamma$ for some k, t.*

Proof: To show that $A^*/\approx_{k,t}$ is in **DA**, we show by induction on k that for any $x, y, z \in A^*$ $u = (xyz)^{kt}y(xyz)^{kt} \approx_{k,t} v = (xyz)^{kt}$. The base case is clear. For $k \geq 1$, it is clear that $\alpha_t(u) = \alpha_t(v)$. Also, let $u = u_0 a u_1$ with $|u_0|_a \leq t$, then clearly this occurrence of the letter a appears in the first $(xyz)^t$ block of u so that in fact we have $u = u_0 a w (xyz)^{(k-1)t} y (xyz)^{kt}$ and $v = u_0 a w (xyz)^{(k-1)t}$. By induction we get $(xyz)^{(k-1)t} y (xyz)^{(k-1)t} \approx_{(k-1),t} (xyz)^{(k-1)t}$ and $(xyz)^{(k-1)t} = xyz^{kt}$ so, since $\approx$ is a congruence, we have $w(xyz)^{(k-1)t} y (xyz)^{kt} \approx_{(k-1),t} w(xyz)^{(k-1)t}$. By left-right symmetry, we can conclude $u \approx_{k,t} v$.

The converse follows from Theorem 3 since, from the definitions of $\approx$ and $\sim$, we have $\approx_{nk,1} \subseteq \sim_{n,k}$. ∎

4 Algebraic decompositions for DA

The variety **DA** admits several interesting algebraic decompositions. We present three different ones here. For two of them, we require the formalism of categories as described in [Thé91].

Let $C = (V, E)$ be a finite directed graph and let C^* be the free category generated by C, i.e. the set of objects of C^* is V and for any $i, j \in V$, $Hom(i, j)$ is the set of paths of finite length from vertex i to vertex j (including, when $i = j$, the path 1_i of length 0). Two paths x, y in C^* are *coterminal* if they belong to the same $Hom(i, j)$ and are *consecutive* if $x \in Hom(i, j)$ and $y \in Hom(j, k)$, for some $i, j, k \in V$. A graph congruence on C^* is an equivalence relation γ on the paths such that

1) $x \,\gamma\, y$ implies x, y are coterminal;

2) $x_1 \gamma y_1$ and $x_2 \gamma y_2$ and x_1, x_2 consecutive implies $(x_1 x_2) \,\gamma\, (y_1 y_2)$.

We are interested in the following case. Let γ be a finite-index congruence on the free monoid A^*. We form the graph $C_\gamma = (V, E)$ where $V = A^*/\gamma \times A^*/\gamma$ and for every $(u, a, v) \in A^* \times A^* \times A^*$ there is an edge $e_{u,a,v}$ from vertex $([u]_\gamma, [av]_\gamma)$ to vertex $([ua]_\gamma, [v]_\gamma)$: of course if $u \,\gamma\, u'$ and $v \,\gamma\, v'$ we do not distinghuish between $e_{u,a,v}$ and $e_{u',a,v'}$ and sometimes use the notation $e_{L,a,K}$ where $L = [u]_\gamma$ and $K = [v]_\gamma$. If $x = x_1 x_2 \ldots x_n \in A^*$ has length n, we will denote by $P_{u,v}(x)$ the path of length n in C^* that goes from vertex $([u]_\gamma, [xv]_\gamma)$ to vertex $([ux]_\gamma, [v]_\gamma)$ where the i^{th} edge is $e_{ux_1 \ldots x_{i-1}, x_i, x_{i+1} \ldots x_n v}$. Note that every path in C^* is of that form.

Let β be a finite index graph congruence on C^*. We form a new congruence $\beta \Box \gamma$ (the *block product* of β and γ) of finite index on A^* by defining $x \, \beta \Box \gamma \, y$ if and only if
1) $x \, \gamma \, y$;
2) for all $u, v \in A^*$, $P_{u,v}(x) \, \beta \, P_{u,v}(y)$.

We define an edge in a graph as *falling* if it connects two distinct strongly connected components. We will also say that two paths p, q are l_1-congruent if they are coterminal and traverse the same set of falling edges. For a finite-index congruence γ on A^* we say that L is a γ-language if it is the union of γ-classes.

Lemma 7 *Let L_0, L_1 be γ-languages and suppose $L_0 a L_1$ is an unambiguous concatenation. Then $L_0 a L_1$ is a $l_1 \Box \gamma$ language. Conversely, any $l_1 \Box \gamma$ language is the disjoint union of unambiguous concatenations $L_0 a_1 L_1 \ldots a_k L_k$ with $a_i \in A$ and where the L_i's are γ-languages.*

Proof: Without loss of generality, we can assume that L_0, L_1 are γ-classes. Clearly, $x \in L_0 a L_1$ if and only if $P_{\epsilon,\epsilon}(x)$ goes through the edge e_{L_0, a, L_1}. But this is a falling edge since the concatenation is unambiguous. Hence the $l_1 \Box \gamma$-class of x determines its membership in $L_0 a L_1$. Similarly, the argument for the converse is based on the observation that falling edges in the graph induced by γ correspond to unambiguous concatenation. ∎

Define on A^* the congruence $x \gamma y$ if and only if $\alpha(x) = \alpha(y)$. It is well known that $M = A^*/\beta$ belongs to the variety $\mathbf{J_1}$ of semilattices if and only if $\beta \supseteq \gamma$. Define now $\delta_1 = \gamma$ and, for every $k > 1$, $\delta_k = l_1 \Box \delta_{k-1}$.

Theorem 8 *$M = A^*/\beta$ belongs to* **DA** *if and only if $\beta \supseteq \delta_k$ for some k.*

Proof: This follows from Lemma 7 and the theorem of Schützenberger cited (and re-proved) in Section 3. ∎

This result is equivalent to saying that **DA** is the smallest variety containing $\mathbf{J_1}$ and closed under block-product with locally trivial categories [PST88].

Consider the congruence $l_1 \Box \gamma$ and suppose $x \, \gamma \, y \gamma \, z \, \gamma \, x^2$. We can observe that $P_{u,v}(xyz) \, l_1 \, P_{u,v}(xz)$. Indeed the two paths are certainly coterminal. Moreover for any $x = x_0 a x_1$, the edge $e_{ux_0, a, x_1 yzv}$ is identical to the edge $e_{ux_0, a, x_1 zv}$ and similarly for any edge coming from a factorization $z = z_0 a z_1$. Finally, for any $y = y_0 a y_1$, the edge $e_{uxy_0, a, y_1 zv}$ is not falling and can thus not be taken into account by the l_1 congruence. In the terminology of [PST88], this means that for any $M = A^*/\beta$ for $\beta \supseteq l_1 \Box \gamma$, there is an **LI**-relational morphism (see e.g. [PST88, Pin86] for the formal definition) from M to A^*/γ.

Since the composition of **LI**-morphisms is also an **LI**-morphism, we can conclude that for any $M \in \mathbf{DA}$, there is an **LI**-relational morphism from M to a semilattice. Conversely, it is not too hard to show that any such monoid will belong to **DA**. This operation, known as the Malcev product, is denoted by Ⓜ . We thus have:

Theorem 9 $\mathbf{DA} = \mathbf{LI}$ Ⓜ $\mathbf{J_1}$.

The last decomposition we present is also using the notion of block product (and is due to [ST02]). Let γ be again the free $\mathbf{J_1}$-congruence on A^*, and construct the graph $C_\gamma = (V, E)$. We define morphisms from E^* to A^* as follows:

$$\phi_{1,a}(e_{u,b,v}) = \begin{cases} b & \text{if } a \notin \alpha(ub), \\ \epsilon & otherwise; \end{cases}$$

$$\psi_{1,a}(e_{u,b,v}) = \begin{cases} b & \text{if } a \in \alpha(u), \\ \epsilon & otherwise; \end{cases}$$

$$\phi_{2,a}(e_{u,b,v}) = \begin{cases} b & \text{if } a \notin \alpha(bv), \\ \epsilon & otherwise; \end{cases}$$

$$\psi_{2,a}(e_{u,b,v}) = \begin{cases} b & \text{if } a \in \alpha(v), \\ \epsilon & otherwise. \end{cases}$$

We next set up a congruence on A^* by letting $x \sim_{n,k} y$ if and only if

1. $x \, \gamma \, y$;
2. For all $a \in A$, for $i = 1, 2$:
   $\phi_{i,a}(P_{\epsilon,\epsilon}(x)) \sim_{n-1,k} \phi_{i,a}(P_{\epsilon,\epsilon}(y))$ and
   $\psi_{i,a}(P_{\epsilon,\epsilon}(x)) \sim_{n,k-1} \psi_{i,a}(P_{\epsilon,\epsilon}(y))$.

It should be clear that this $\sim_{n,k}$ is identical to the one defined in Section 3. Let $\mathbf{V}$ be the smallest variety containing $\mathbf{J_1}$ and satisfying $\mathbf{V} \Box \mathbf{J_1} = \mathbf{V}$. The above observation establishes that $\mathbf{DA} \subseteq \mathbf{V}$. Since it is easy to show that $\mathbf{DA}$ does satisfy $\mathbf{DA} \Box \mathbf{J_1} = \mathbf{DA}$, we have in fact $\mathbf{V} = \mathbf{DA}$.

5 Logical Descriptions of DA

In this section we will use logical formulas to define properties of strings over a finite alphabet A. First, we consider first-order formulas where variables are integers denoting the positions in a string and predicates are either the binary predicate $<$ or, for each $a \in A$, the unary predicate Q_a, where $Q_a x$ means "position x in the string holds an a". A sentence ϕ in this logic naturally defines a language in A^* consisting of strings over which ϕ holds (see [Str94] for a detailed introduction). As an example, the sentence $\exists i \exists j (Q_a i \wedge Q_b i \wedge i < j)$ defines the regular language $A^* a A^* b A^*$.

We denote by $\mathbf{FO}[<]$ the class of languages that can be defined by a first-order sentence as above. It is well known that a language belongs to $\mathbf{FO}[<]$ if and only if its syntactic monoid is finite aperiodic [MP71, Sch65].

If ϕ is a sentence not containing i as a variable, we denote $\phi_{<i}$ (resp. $\phi_{>i}$) the logical formula obtained from ϕ by requiring that every bound variable in ϕ be strictly less than i (resp. strictly greater than i).

5.1 $\mathbf{\Sigma_2} \cap \mathbf{\Pi_2}[<]$

The most studied subclasses of $\mathbf{FO}[<]$ are the ones defined by bounded quantifier alternation. We will naturally denote by $\mathbf{\Sigma_k}[<]$ (resp. $\mathbf{\Pi_k}[<]$) the class of languages definable by first-order formulas in prenex form having k alternating blocks of existential and universal quantifiers starting with a block of existential quantifiers (resp. a block of universal quantifiers). The following is a corollary of a powerful theorem due to Pin and Weil [PW97]:

Theorem 10 *A language L belongs to both $\mathbf{\Sigma_2}[<]$ and $\mathbf{\Pi_2}[<]$ iff $M(L)$ belongs to $\mathbf{DA}$.*

Proof: The "if" direction of the statement can be obtained from the results of Section 3 without resorting to the result of Pin and Weil. Indeed we noted that if $M(L)$ is in $\mathbf{DA}$, then there exists n, k such that L can be expressed as a disjoint union of $*_{n,k}$ languages. It is clear that $*_{n,0}$ languages are in $\mathbf{\Sigma_2} \cap \mathbf{\Pi_2}[<]$. Now for $n \geq 1$, suppose we want to define the language $L = L_0 a L_1$ where L_0, L_1 are $*_{n,k-1}$ and $*_{n-1,k}$ languages respectively with L_1 a subset of $(A - \{a\})^*$. Then L can be defined by the Σ_2 sentence

$$\exists i (Q_a i \wedge \phi_{<i} \wedge \psi_{>i})$$

where ϕ and ψ, obtained from the induction, are the Σ_2 sentences defining L_0 and L_1 respectively. Similarly, we can express membership in L by saying that there is a position holding an a and that any position holding an a is either not holding the last a, or defines a suffix belonging to L_1 and a prefix belonging to L_0, i.e. L is defined by the Π_2 sentence:

$$\forall i ((\neg Q_a i \vee (\exists j (Q_a j \wedge i < j)) \vee (Q_a i \wedge \rho_{<i} \wedge \chi_{>i})))$$

where ρ, χ are now the $\Pi_2[<]$ sentences for L_0 and L_1.

For the "only if" part, one first needs a result of Arfi [Arf91] which shows that a language is in $\mathbf{\Sigma_2}[<]$ if and only if it is expressible as the disjoint union of languages of the form $A_0^* a_1 A_1^* \ldots a_k A_k^*$ (without any sort of unambiguity restriction). If L is in $\mathbf{\Sigma_2} \cap \mathbf{\Pi_2}[<]$, both L and its complement can be expressed in that way. Let n be the length of the largest expression needed to obtain L or its complement. Now for any $x, y, z, u, v \in A^*$, if $u(xyz)^{n+1}v$ belongs to L then it belongs to some $A_0^* a_1 A_1^* \dots a_k A_k^* \subseteq L$ (with $k \leq n$) and it can be factorized as $w_0 a_1 \dots a_k w_k$. By the pigeon hole principle, some w_j must contain (xyz) entirely so $\alpha(xyz) \subseteq A_j$. Consequently, $u(xyz)^{n+1}y(xyz)^{n+1}v$ also belongs to $A_0^* a_1 A_1^* \dots a_k A_k^* \subseteq L$. Similarly, using the representation of L's complement, if $u(xyz)^{n+1}v$ is not in L, then $u(xyz)^{n+1}y(xyz)^{n+1}v$ is not. So $M(L)$ must satisfy $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$ and thus belongs to **DA**. ∎

5.2 $\mathbf{FO_2}[<]$ *and Unary Temporal Logic*

Another meaningful measure of the complexity of these first-order sentences is given by the number of variables used. We will denote by $\mathbf{FO_k}[<]$ the class of sentences using only k different variables (possibly re-used of course). For example, to describe the language of strings in A^* having at least three a's, one would naturally use the three variable sentence

$$\exists i \exists j \exists k ((i < j < k) \wedge Q_a i \wedge Q_a j \wedge Q_a k)$$

which is equivalent to the two-variable sentence:

$$\exists i \exists j ((Q_a i \wedge Q_a j \wedge (i < j) \wedge \exists i ((j < i) \wedge Q_a i)))$$

It is an easy exercise to show that a language L is definable in $\mathbf{FO_1}[<]$ if and only if its syntactic monoid is idempotent and commutative. On the other hand three variables suffice to express *any* first-order definable language [Kam68, IK89].

Etessami, Vardi and Wilke showed in [EVW97] that $\mathbf{FO_2}$ is exactly the class of languages expressible in a fragment of temporal logic called *unary temporal logic* (or **UTL**). Formally, a **UTL** formula is built using ordinary Boolean connectives, elements of the alphabet A and the two unary temporal operators eventually in the future, denoted $\diamondplus$ and "eventually in the past": $\diamondminus$. Given a word $w = w_1 w_2 \dots w_n$ and a **UTL** formula ϕ, we define what it means for ϕ to be true over w at i (and denote $(w, i) \models \phi$) using the following semantics (the semantics of Boolean connectives is clear):

- $(w, i) \models a$ if w_i is a.
- $(w, i) \models \diamondplus\phi$ if there exists $j > i$ such that $(w, j) \models \phi$.
- $(w, i) \models \diamondminus\phi$ if there exists $j < i$ such that $(w, j) \models \phi$.
- We also say that a formula $\diamondplus\phi$ (resp. $\diamondminus\phi$) is true over w if $(w, 0) \models \diamondplus\phi$ (resp. $(w, n+1) \models \diamondminus\phi$.

For a unary temporal logic formula ϕ, we write $L(\phi)$ the set of words over which ϕ is true and denote by **UTL** the class of languages which are $L(\phi)$ for some unary temporal logic formula ϕ.

Once again, the natural classes $\mathbf{FO_2}$ and **UTL** turn out to have nice algebraic characterizations (obtained in [TW98]) and one can in fact show:

Theorem 11 ([EVW97, TW98]) *For a language L, the following are equivalent:*

1. *$M(L)$ is in* **DA***;*
2. *L is in* **UTL***;*
3. *L is in* $\mathbf{FO_2}$.

Proof [sketch]:

1) $\leftrightarrow$ 3) is one of the main results of [TW98].

2) $\leftrightarrow$ 3) is due to [EVW97].

1) $\leftrightarrow$ 2): we give a direct proof of this last equivalence. For one direction it is sufficient to show by induction that $*_{n,k}$ languages are in **UTL**. It is clear that all $*_{n,0}$ languages are in **UTL**. Next, let $A_1 = A_0 \cup \{a\}$ (with $a \notin A_0$) and let $L_0 \subseteq A_0^*$, $L_1 \subseteq A_1^*$ be $*_{|A_0|,k}$ and $*_{|A_1|,k}$ languages respectively. By induction, we have **UTL** formulas ϕ, ψ defining L_0 and L_1 respectively. It is easy to verify that the language $L_0 a L_1$ will be defined by the formula

$$\Diamond^{+} a \wedge \phi' \wedge \psi'' \wedge (\neg \Diamond^{-} a)$$

Here, ϕ' is obtained from ϕ, by replacing $\Diamond^{+}$ operators by $\Diamond^{-}$ and vice-versa and by replacing each subformula $\Diamond^{+}\rho$ by $\Diamond^{+}(\rho \wedge \neg \Diamond^{-} a)$ and similarly for subformulas $\Diamond^{-}\rho$. Similarly, ψ'' is obtained from ψ by replacing each subformula $\Diamond^{+}\rho$ by $\Diamond^{+}(\rho \wedge \Diamond^{-} a)$.

For the converse, let u, v, x, y, z be strings in A^* and define $s_n = u(xyz)^n y (xyz)^n v$ and $t_n = u(xyz)^{2n} v$. We want to prove by induction on n that if ϕ is a temporal logic formula of operator depth at most $d \leq n$, then $(s_n, i) \models \phi$ if and only if $(t_n, f_d(i)) \models \phi$ where $f_d(i)$ maps a position in the $u(xyz)^d$ prefix or $(xyz)^d v$ suffix of s_n to the same position in t_n. Positions in the middle $(xyz)^{n-d} y (xyz)^{n-d}$ are mapped to the corresponding position in the d^{th} copy of (xyz) from the left in t_n if they lie within $(xyz)^{n-d}$ and mapped to the d^{th} copy of (xyz) in t_n *from the right* otherwise.

The claim is trivially true for $n = 0, 1$. Next, suppose $(s_{n+1}, i) \models \Diamond^{+}\phi$ where ϕ has operator depth $d \leq n$. By definition, there exists $j > i$ such that $(s_{n+1}, j) \models \phi$ and so, by induction we have $(t_{n+1}, f_d(j)) \models \phi$. Clearly, $f_d(j) > f_{d+1}(i)$ so we do have $(t_{n+1}, f_{d+1}(i)) \models \Diamond^{+}\phi$. Conversely, suppose

$(t_{n+1}, f_{d+1}(i)) \models \diamondplus\phi$. There is $k > f_{d+1}(i)$ with $(t_{n+1}, k) \models \phi$. For any j such that $f_d(j) = k$, we have by induction $(s_{n+1}, j) \models \phi$ and one can choose such a j to be greater than i so $(s_{n+1}, i) \models \diamondplus\phi$. This proves our claim and shows that the syntactic monoid of $L(\phi)$ satisfies $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$, i.e. is in $\mathbf{DA}$. ∎

It is possible to extend further the expressive power of $\mathbf{FO}$ by allowing the use of so-called modular quantifiers (see e.g. [Str94] for relevant definitions). One denotes by $\mathbf{(FO + MOD)_k}[<]$ the class of languages definable by such generalized first-order sentences using at most k variables. These were studied by Straubing and Thérien in [ST01] who showed $\mathbf{(FO + MOD)_3}[<] = \mathbf{FO} + \mathbf{MOD}[<]$ and, most notably, that $\mathbf{(FO + MOD)_2}[<] = \mathbf{DA} * \mathbf{G_{sol}}$, where $\mathbf{DA} * \mathbf{G_{sol}}$ is the class of languages whose syntactic monoid divides the wreath product of a monoid in $\mathbf{DA}$ and a solvable group. Another generalization of Theorem 11 has recently yielded algebraic characterizations and decidability results for the levels of the so-called Until/Since hierarchy in temporal logic [TW02].

To conclude this section, we would like to point out that the above results imply the existence of polynomial space algorithms for deciding whether a regular language L (input, say, as a DFA) is in $\mathbf{\Sigma_2} \cap \mathbf{\Pi_2}[<]$, $\mathbf{FO_2}[<]$ or $\mathbf{UTL}$.

6 Monoids as Machines and Complexity Problems

The notion of recognition of a language by a finite monoid can be extended in various ways to provide a meaningful algebraic point of view on computation and complexity classes. In these contexts, the monoid is viewed as a machine using some relatively simple mechanism to process input. Quite naturally, the power of these machines is related to the algebraic properties of the underlying monoid. This approach has been particularly successful in providing algebraic characterizations of classes of languages recognized by small depth boolean circuits via the notion of programs over monoids (see Subsection 6.2 for definitions and basic results) and of subclasses of PSPACE via the leaf-language formalism [HLS+93].

This, in turn, has given rise to questions concerning the computational complexity of certain problems whose hardness is parametrized by an underlying monoid. Typically, these are studied first in the restricted cases of groups and aperiodic monoids. In the latter case, $\mathbf{DA}$ almost always turns up at the center of these investigations and we will survey such examples in this section.

6.1 2-way Partially Ordered DFA's

A finite automaton (FA) is said to be partially ordered (or p.o.) if all its strongly connected components are trivial, i.e. if there is a partial order $\prec$ on the states of the automaton such that whenever there exists a transition from state s to state s', we have $s \prec s'$. It is a classical result that a language is recognized by a p.o. deterministic FA if and only if its syntactic monoid is $\mathcal{R}$ trivial. It is also quite simple to show, by using the result of Arfi already cited in the proof of Theorem 10, that a language can be recognized by a p.o. non-deterministic FA if and only if it belongs to $\Sigma_2[<]$.

Two-way finite automata (2-DFA's) are known to have the same computational power as DFA's and have thus received limited attention since their introduction [HU79]. They can, however provide more concise descriptions of certain languages and can be studied as a way to compute translations on strings [MSTV00].

Formally, a 2-DFA is a tuple[a] $T = (S, A, \delta, l_0, F)$ where S, the set of states, is the disjoint union of two sets L (the states "reached" from the left) and R (the one "reached" from the right), A is the input alphabet, $l_0 \in L$ is the initial state, $F \subseteq S$ is the set of accepting states, $\delta : (S \times A) \cup (L \times \{\lhd\}) \cup (R \times \{\rhd\}) \rightarrow S$ is a total transition function and $\rhd, \lhd \notin A$ are the left- and right markers respectively. If T is in state s reading the symbol a, it enters state $\delta(s,a)$ and moves its head to the right if $\delta(s,a) \in L$ and to the left if $\delta(s,a) \in R$. On input $w = w_1 w_2 \dots w_n$, the machine is initially in state l_0 scanning the symbol w_1 within the string $\rhd w_1 w_2 \dots w_n \lhd$. For technical reasons, we require that a 2-DFA halts for any given input by "falling off" its input tape, i.e. by encountering a transition $\delta(r, \rhd) \in R$ or a transition $\delta(l, \lhd) \in L$.

Recently, it was proved that a language can be recognized by a partially ordered 2-DFA if and only if its syntactic monoid is in **DA**. In order to establish this result, [STV01] introduce the notion of *turtle programs*, of which we give here a brief description. A turtle program P over an alphabet A is a sequence of turtle instructions, i.e. pairs (d, t) where $d \in \{\rightarrow, \leftarrow\}$ is a direction and $t \in A$ is a target symbol. On an input $w = w_1 w_2 \dots w_n$, a turtle program whose first instruction is a pair $(\rightarrow, a)$ (resp. $(\leftarrow, a)$) moves a read-only head (the turtle) from w_1 right (resp. from w_n left) until the head encounters a position w_i holding an a. If no such position exists, we say that the instruction has failed. Each subsequent instruction similarly moves the turtle right or left from its current position until it encounters its target symbol or fails by reaching either end of the input. We define as $L(P)$, the language recognized by the turtle program P, as the set of words w such that

[a] We present here the definition given in [MSTV00].

the sequence of all instructions of the program succeed on input w.

Theorem 12 ([STV01]) *Let K be a language. The following are equivalent:*
a) $M(K)$ is in **DA***;*
b) K is the Boolean combination of turtle languages, i.e. languages recognized by a turtle program;
c) K can be recognized by a p.o. 2-DFA.

Proof [sketch]: a) $\rightarrow$ b) follows from the $\sim_{n,k}$ characterization of **DA**.

For b) $\rightarrow$ c), note first that a turtle language can trivially be recognized by a p.o. 2-DFA and that the class of languages recognizable by p.o. 2-DFA's is closed under complementation. It is a simple exercise to show that this class is also closed under union, which completes the argument.

c) $\rightarrow$ a) is the hardest implication. Let T be 2-DFA over the alphabet A, with states $S = L\dot{\cup}R$ partially ordered by $\prec$ and recognizing a language K. Without loss of generality, we will assume that T always halts by exiting on the right. We will show how to obtain from T a p.o. NFA N for K. Together with Theorem 10, this will yield our result since K's complement is also recognized by a p.o. 2-DFA. States of our NFA N are odd length sequences of T states $(s_0, s_1, \ldots, s_{2k})$ with $s_i \in L$ for even i and $s_i \in R$ for odd i and such that $s_i \prec s_j$ for $i < j$. This last restriction shows that N has a bounded number of states. We think of these "crossing sequences" (following the terminology of [HU79]) as the sequence of states in which T might possibly cross the border between two input tape cells and add a transition in N from state $p = (l_0, r_1, \ldots, l_i)$ to state $q = (l'_0, r'_1, \ldots, l'_j)$ while reading a if and only if they are locally consistent, i.e. if there is some word $w = uav$ such that execution of T on w yields crossing sequences p and q on the left and right borders of the cell containing the a. Similarly, N's initial states are any sequences $p = (l_0, r_1, \ldots, l_k)$ such that l_0 is T's initial state and such that $\delta(r_i, \triangleright) = l_{i+1}$ and the final states are sequences $p = (l_0, r_1, \ldots, l_k)$ such that $\delta(l_i, \triangleleft) = r_{i+1}$ for $i < k$ and $\delta(l_k, \triangleleft) = l$ for some accepting state l of T. One can easily verify that N recognizes K.

We define a relation $\sqsubset$ on the states of N as follows: let $p = (l_0, r_1, \ldots, l_t) \sqsubset q = (l'_0, r'_1, \ldots, l'_u)$ if and only if there exist indices $p_1 < \ldots < p_k = t$ and $0 = q_1 < \ldots < q_k$ such that for all $1 \leq i \leq k$ we have $l_{p_i} \prec l'_{q_i} \prec r'_{q_{i+1}-1} \prec r_{p_i+1}$. In other words $p \sqsubset q$ if and only if the two chains $l_0 \prec r_1 \prec \ldots \prec l_t$ and $l'_0 \prec r'_1 \prec \ldots \prec l'_u$ can be interleaved to form a new chain starting in l_0, ending in l'_u and such that each *new* pair in the chain is of the form $l_i \prec l'_j$ or $r'_i \prec r_j$. It can be shown that $\sqsubset$ is a partial-order and that N has a transition from p to q only if $p \sqsubset q$. ∎

Because we know that languages recognized by **DA** lie in $\mathbf{\Sigma_2} \cap \mathbf{\Pi_2}[<]$, we can conclude that $K \subseteq A^*$ is recognized by a p.o. 2-DFA if and only if both K and its complement $\overline{K}$ can be recognized by p.o. NFA's. This somewhat innocent looking automata-theoretic fact has, to the best of our knowledge, no proof simpler than the one implicitly presented in this survey.

6.2 Equation and Program Satisfiability

We briefly mentioned earlier the notion of programs over monoids. Formally, given an input alphabet A, an n-input program of length s over a monoid M is a sequence of instructions

$$(i_1, f_1), (i_2, f_2), \ldots, (i_s, f_s)$$

with $i_j \in \{1, \ldots, n\}$ and $f_j : A \to M$, together with an accepting subset $F \subseteq M$. The program accepts the input $x_1 x_2 \ldots x_n$ iff the sequence $f_1(x_{i_1}) \ldots f_l(x_{i_s})$ evaluates to an element of F.

We say that $L \subseteq A^*$ is P-recognized by M if there is a sequence $(\varphi_0, \varphi_1, \ldots)$ where φ_n is an n-input program over M, of length $l(n)$ with $l(n)$ polynomial in n, such that the program φ_n accepts $L \cap A^n$. This formalism gives rise to algebraic characterizations of many well-known boolean circuit classes (see [BT88] and [MPT91] for a thorough survey). In particular, a language L belongs to the class NC^1 (resp. AC^0, CC^0, ACC^0) if and only if it is recognized by a program over a finite monoid (resp. aperiodic monoid, solvable monoid, solvable group).

For any fixed M we define the two following similar problems:

- EQUATION SATISFIABILITY over M (EQNSAT_M): we are given an equation over a monoid M of the form

  $$c_0 X_{i_1} c_1 \ldots c_{n-1} X_{i_n} c_n = m$$

  where the c_i's and m are constants in M and the X_i's are variables which might be repeated. The problem is to determine whether one can assign values in M to the variables so that the equation is satisfied.

- PROGRAM SATISFIABILITY over M ($\mathrm{PROGSAT}_M$): we are given an n-input program ϕ of length l over M and a target set $F \subseteq M$. The problem is to determine whether there exists $x \in \{0,1\}^*$ such that $\phi(x) \in F$.

Both problems trivially lie in NP. Also, it is a simple exercise to show that EQNSAT_M reduces to $\mathrm{PROGSAT}_M$ for any M.

The hardness of these problems is of course highly dependent of the underlying monoid. In the case where M is aperiodic, the following dichotomy results involving **DA** can be obtained [BMM+00]:

Theorem 13 a) EQNSAT_M *and* $\mathrm{PROGSAT}_M$ *lie in P for any* M *in* $\mathbf{DA}$.

b) *If* M *is not in* $\mathbf{DA}$ *then* $\mathrm{PROGSAT}_M$ *is NP-complete.*

c) *If* M *is not in* $\mathbf{DA}$*, there exists a monoid* N *in the variety* $\langle M \rangle$ *generated by* M *such that* EQNSAT_M *is NP-complete.*

Proof [sketch]: For a), it is sufficient to establish the upper bound for $\mathrm{PROGSAT}_M$. Recall from Section 3 that the set $M_F = \{w : eval_M(w) \in F\}$ is the disjoint union of unambiguous concatenations $M_0^* a_1 M_1^* \dots a_k M_k^*$. It is thus sufficient to determine whether there is some $x \in \{0,1\}^*$ with $\phi(x) \in M_0^* a_1 M_1^* \dots a_k M_k^*$.

This can only happen if there are k instructions $i_1, \dots, i_k$ in the program that can, on some input, output the bookmarks $a_1, \dots, a_k$. So, it suffices to consider the at most $\binom{l}{k}$ k-tuples of instructions possibly responsible for the presence of the bookmarks and, for each of them, check in linear time whether there is an x such that $\phi(x)$ is of the form $M_0^* a_1 M_1^* \dots a_k M_k^*$.

For b) and c), it is sufficient to show that $\mathrm{PROGSAT}_U$ and EQNSAT_{BA_2} are both NP-complete. These lower bounds follow from fairly simple reductions from 3-SAT and 1-3SAT respectively. ∎

6.3 Communication Complexity

Over the past twenty years, communication complexity has become an essential and versatile tool in theoretical computer science (see [NK97] for a complete introduction). The communication complexity of monoids has been studied [RTT98, Tes99] to understand the power (and limitations) of communication complexity based lower bounds on small depth circuits (e.g. [Gro92, Gro94, Gro98, HG90] among many others) and to provide an algebraic point of view on some of these circuit and communication complexity bounds.

We consider the following communication game involving k parties: the players $P_1, \dots, P_k$ want to collaborate to evaluate, in a given monoid M, a string of n elements $x_1, \dots, x_n$. Each player P_j, has access to all the x_i except the ones such that $i \equiv j \pmod{k}$. One can imagine, for instance that P_j has these x_i's written on his forehead, available to all players but himself. The players are allowed to communicate by broadcasting bits according to some fixed protocol. The k-party communication complexity of M, denoted $C^{(k)}(M)$, is the least number of bits that they need to communicate in the worst case such that all players know the value $eval_M(x_1, \dots, x_n)$. (Note that $C^{(k)}(M)$ is in fact a function from $\mathbb{N}$ to $\mathbb{N}$ as the players may have a specific protocol for each input length.)

To argue for the relevance of this definition, we note that the communication complexity of a regular language[b] L is $\Theta(C^{(k)}(M(L)))$ and that for any $f : \mathbb{N} \to \mathbb{N}$, the class of monoids M such that $C^{(k)}(M) = O(f)$ forms a variety (see [RTT98]).

For groups, the question is completely solved: a group G has k-party communication complexity $O(1)$ if and only if it is nilpotent of class $(k-1)$ and has complexity $\Theta(n)$ otherwise. If we turn to aperiodics, **DA** appears as the key variety.

Theorem 14 ([RTT98, PRTT02]) *Let M be an aperiodic monoid.*

1. *In the 2-party case, $C^{(2)}(M) = O(\log n)$ if and only if M belongs to* **DA** *and $C^{(2)}(M) = \Theta(n)$ otherwise.*
2. *If $M \in \mathbf{V}_{\approx_{\mathbf{k},\mathbf{t}}}$ then $C^{(k+1)}(M) = O(1)$.*
3. *In fact, there exists a k such that $C^{(k)}(M) = O(1)$ only if M belongs to* **DA**.

Proof [sketch]: Both of the upper bounds for **DA** are easy to derive from the parametrization of **DA** induced by the congruence $\approx$ defined in Section 3. It is sufficient to establish the upper bound for $A^*/\approx_{k,t}$ for any alphabet A and any k, t. Let us first describe the constant cost $(k+1)$-party protocol for $A^*/\approx_{k,t}$ which we will build by induction on k. The case $k = 0$ is obvious since $\approx_{0,t}$ is trivial. Suppose $k \geq 1$ and take a set of representatives $[u_i]$ of $A^*/\approx_{k,t}$. For each u_i, the players will check whether $w \approx_{k,t} u_i$. First, it is clear that by exchanging $O(t) = O(1)$ bits, the players can insure that $\alpha_t(w) = \alpha_t(u_i)$. If $k = 1$, we are done because there is no recursive condition to check. If $k \geq 2$, 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 need to identify the location of the j^{th} a (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 $O(k \log t) = O(1)$ 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. We are assuming that there are at least 3 players, so Player l can indeed be identified. 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.

[b] This can be defined in a way similar to that of a monoid, with the x_i's being words in A^*

In the end, all players know which player ignores the j^{th} a and all except that player know the location of that a. Hence, there are k players available to check whether $v_0 \approx_{k-1,t} w_0$ and $v_1 \approx_{k-1,t} w_1$. By induction this is doable with $O(1)$ communication. Left-right symmetry completes the protocol.

For the 2-party $O(\log n)$ upper bound, it suffices to note that by exchanging the positions of the first t a's that they see in w, *both* players can correctly identify the location of the j^{th} a of w for any $j \leq t$ and apply the induction step. ∎

Naturally, we can ask whether the k-player protocol just described is optimal. We conjecture in fact:

Conjecture 15 *If M is aperiodic, we have $M \in \mathbf{V}_{\approx_{\mathbf{k},\mathbf{t}}}$ for some t if and only if $C^{(k+1)}(M) = O(1)$.*

For $k = 0$, this is obviously true since only the trivial monoid has 1-party communication complexity $O(1)$ (since there is no access to the input). For $k = 1$, one can easily prove that $\bigcup_t \mathbf{V}_{\approx_{\mathbf{1},\mathbf{t}}} = \mathbf{A_{com}}$ and it is shown in [RTT98] that this variety also corresponds exactly to the aperiodics having constant 2-party communication complexity.

Only partial results are known when $k \geq 2$ [PRTT02]. We should stress that this conjecture would imply that the regular language $A^*a_1A^* \ldots A^*a_kA^*$ can not be recognized by k players in constant communication. Of course, any set of $(k-1)$ input positions is seen by at least one player in the k-party model and the former remark would strengthen the intuition according to which this is the only real building block of the k-party model's computational power. Already, the existing results on the communication complexity of groups can be seen as stating that a group has high k-party complexity if and only if its computing power is not limited to "counting" subwords of length $(k-1)$ (see [RTT98] for a formal presentation).

6.4 Learning

We have seen in the communication complexity setting how an algebraic approach can help in outlining some fundamentally hard aspects of a computational problem. In [GTT01], an algebraic point of view on computational learning theory is developed.

The model considered is Angluin's query-based model of exact-learning. The setup is as follows. We fix a monoid M and consider functions $f : M^n \to M$, representable by an expression over M, i.e. $f(X_1, \ldots, X_n) = c_0X_{i_1}c_1 \ldots c_{s-1}X_{i_s}c_s$. The goal of a *Learner* (i.e. of a learning algorithm) is to identify such an unknown *target function* f by interacting, according to some protocol, with a *Teacher* who knows f. We assume that the Learner knows n and s that he is allowed to make two types of queries: Evaluation and Equivalence. For an *Evaluation* query, the Learner provides $w \in M^n$ and the Teacher simply returns $f(w)$. For an *Equivalence* query, the Learner provides an hypothesis h (in our case, we consider that h is also given as an expression over M) and the Teacher either returns YES if $h(w) = f(w)$ for all $w \in M^n$ or returns a counterexample $(w, f(w))$ where $w \in M^n$ and $f(w) \neq h(w)$.

The two measures of complexity we are concerned with are the number of queries made by the Learner and the computation time he requires. We say that expressions over a monoid M are learnable with $q_1(n,s)$ Evaluation queries, $q_2(n,s)$ Equivalence queries and $t(n,s)$ time if there exists a learning algorithm that can produce, within these bounds, an M-expression E such that $E(w) = f(w)$ for all $w \in M^n$.

Theorem 16 *Expressions over M in* **DA** *are learnable from a polynomial number of Evaluation queries and unbounded computation time.*

Proof: As in the PROGSAT$_M$ algorithm, we will use the fact that when M is in **DA**, the subsets of M^* defined as $M_m = \{w : eval_M(w) = m\}$ can be expressed as the disjoint union of unambiguous concatenations $M_0^* a_1 M_1^* \dots a_k M_k^*$. Suppose we have an expression $E(X_1, \dots, X_n)$ over some M in **DA** such that for each m in M, the unambiguous concatenations needed to express M_m have at most k bookmarks.

The Learner asks the Teacher to evaluate $E(w)$ for the $\binom{|M|^{2k}}{n2k}$ words of M^n having at most $2k$ non-identity components. By exhaustive search, one can construct an expression $F(X_1, \dots, X_n)$ agreeing with E on all these inputs.

Consider now any $x \in M^n$. We claim that $E(x) = F(x)$. Indeed, construct $z \in M^n$ by replacing in x each letter not occurring as a bookmark in either $E(x)$ or $F(x)$ (seen as words in M^*). By the definition of F, we have $E(z) = F(z)$, but also $E(z) = E(x)$ and $F(x) = F(z)$ since the bookmarks of $E(x)$ and $F(x)$ still appear in $E(z)$ and $F(z)$. Thus, E and F are equivalent expressions. ∎

In the special cases where M is aperiodic and commutative, $\mathcal{R}$-trivial and idempotent or $\mathcal{L}$-trivial and idempotent, it is possible to actually learn the expressions in polynomial time, but it is not known for a general $M \in$ **DA** whether an expression F as above can be obtained more efficiently than by exhaustive search.

In contrast to Theorem 16, it is also established in [GTT01] that expressions over BA_2 and U, the minimal aperiodics outside **DA**, are not learnable with a subexponential number of Evaluation queries.

If one considers Equivalence queries alone, then it can be shown that expressions over a commutative idempotent monoid can be learned using a linear number of Equivalence queries alone, while expressions over any non-idempotent M can not be learned with a sub-exponential number of queries.

6.5 Membership

Consider the problem MEMB of determining whether a function g from a set X to itself can be written as the composition of some functions $f_1, \ldots, f_t$, i.e. of deciding whether g is a member of the transformation monoid $\langle f_1, \ldots, f_t \rangle$ generated by the f_i's.

MEMB is known to be complete for polynomial space in general [Koz77]. A series of technically involved papers (culminating in [BLS87]) was necessary to show that when the f_i's are permutations on X, i.e. when the monoid $\langle f_1, \ldots, f_m \rangle$ is a group, MEMB lies in the class NC (polylogarithmic parallel time using a polynomial number of processors).

For any variety $\mathbf{V}$, we denote by $MEMB(\mathbf{V})$ the restriction of MEMB to instances where the monoid $\langle f_1, \ldots, f_t \rangle$ lies in $\mathbf{V}$. The proof of [Koz77] shows in fact that $MEMB(\mathbf{A})$ is already PSPACE-complete. The complexity of MEMB for subvarieties of $\mathbf{A}$ was investigated in [BMT92] who proved:

- $MEMB(\mathbf{R_1} \vee \mathbf{L_1})$ lies in P.
- $MEMB(\mathbf{A_1})$ lies in NP.
- $MEMB(\mathbf{R})$ is NP-complete.
- $MEMB(\mathbf{V})$ is PSPACE-complete for any aperiodic variety $\mathbf{V}$ not contained in $\mathbf{DA}$ and NP-hard for any aperiodic variety $\mathbf{V}$ not contained in $\mathbf{R_1} \vee \mathbf{L_1}$.

It is further conjectured that $MEMB(\mathbf{DA})$ in fact lies in NP. The missing upper bound would follow if one could show the following refinement of Schützenberger's theorem: a language recognizable by an n-state automaton whose transformation monoid lies in $\mathbf{DA}$ can be expressed as the disjoint union of concatenations $A_0^* a_1 \ldots a_k A_k^*$ where k is bounded by a polynomial in n.

7 Summary

We have shown that languages whose syntactic monoid lies in $\mathbf{DA}$ have useful combinatorial, logical, and automata-theoretic characterizations. For a language $L \subseteq A^*$, the following equivalences are known:

- $M(L)$ is in **DA**;
- L is the union of $\sim_{n,k}$ classes for $n = |A|$ and some k;
- L is the union of $\approx_{k,t}$ classes for some k, t;
- L is the disjoint union of unambiguous concatenations $A_0^* a_1 \ldots a_k A_k^*$ with $A_i \subseteq A$, $a_i \in A$;
- L is both $\mathbf{\Sigma_2}[<]$ and $\mathbf{\Pi_2}[<]$ definable;
- L is in $\mathbf{FO_2}[<]$, i.e. is definable by a two variable first-order formula;
- L is in **UTL**, i.e. definable by a unary temporal logic formula;
- L can be recognized by a partially-ordered 2-way DFA.

We have also shown how these characterizations can be used to answer questions motivated by computation and complexity theory: Checking the satisfiability of an equation or a program, evaluating a product using limited communication, learning expressions can always be done efficiently when the underlying monoid lies in **DA** whereas these problems are, typically, provably much harder when the monoid is aperiodic but not in **DA**.

Moreover, the variety **DA** has a number of useful algebraic characterizations. In particular, **DA** is the smallest variety $\mathbf{V}$ containing $\mathbf{J_1}$ satisfying $\mathbf{V} \Box \mathbf{J_1} = \mathbf{V}$ and the smallest variety $\mathbf{V}$ containing $\mathbf{J_1}$ satisfying $l_1 \Box \mathbf{V} = \mathbf{V}$.

Acknowledgements.

Research supported by grants from NSERC, FCAR and the von Humboldt Foundation. The authors would also like to acknowledge the financial support of Fundação Calouste Gulbenkian (FCG), Fundação para a Ciência e a Tecnologia (FCT), Faculdade de Ciências da Universidade de Lisboa (FCUL) and Reitoria da Universidade do Porto.

References

[Arf91] M. Arfi. Opérations polynomiales et hiérarchies de concaténation. *Theoretical Computer Science*, 91:71–84, 1991.

[BLS87] L. Babai, E. Luks, and A. Seress. Permutation groups in NC. In *Proc. ACM STOC*, pages 409–420, 1987.

[BMM+00] D. Mix Barrington, P. McKenzie, C. Moore, P. Tesson, and D. Thérien. Equation satisfiability and program satisfiability for finite monoids. In *Proc. MFCS'00*, pages 172–181, 2000.

[BMT92] M. Beaudry, P. McKenzie, and D. Thérien. The membership problem in aperiodic transformation monoids. *Journal of the ACM*, 39(3):599–616, 1992.

[BT88] D. A. Mix Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35(4):941–952, 1988.

[EVW97] K. Etessami, M. Vardi, and T. Wilke. First-order logic with two variables and unary temporal logic. In *Proc. IEEE LICS*, pages 228–235, 1997.

[Gro92] V. Grolmusz. Separating the communication complexities of MOD m and MOD p circuits. In *Proc. 33rd IEEE FOCS*, pages 278–287, 1992.

[Gro94] V. Grolmusz. A weight-size trade-off for circuits and MOD m gates. In *Proc. 26th ACM STOC*, pages 68–74, 1994.

[Gro98] V. Grolmusz. Circuits and multi-party protocols. *Computational Complexity*, 7(1):1–18, 1998.

[GTT01] R. Gavaldà, P. Tesson, and D. Thérien. Learning expressions and programs over monoids. Submitted to Information and Computation. Extended abstract appears in Proc. STACS 2001. Available from www.lsi.upc.es/ gavalda/papers.html, 2001.

[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*, pages 200–207, 1993.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[IK89] N. Immerman and D. Kozen, Definability with Bounded Number of Bound Variables. *Information and Computation*, 83(2):121–139, 1989.

[Kam68] J. A. W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, University of California, Berkeley, 1968.

[NK97] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[Koz77] D. Kozen. Lower bounds for natural proof systems. In *Proc. IEEE FOCS*, pages 254–266, 1977.

[KR65] K. Krohn and J.L. Rhodes, *Algebraic Theory of Machines, I Principles of Finite Semigroups and Machines, Transactions of the American Mathematical Society*, 116:450–464, 1965.

[MP71] R. McNaughton and S. Papert. *Counter-Free Automata.* MIT Press, Cambridge, Mass., 1971.

[MPT91] P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.

[MSTV00] P. McKenzie, T. Schwentick, D. Thérien, and H. Vollmer. The many faces of a translation. In *Proc. ICALP'00, LNCS 1853*, pages 890–901, 2000.

[Pin86] J.-E. Pin. *Varieties of formal languages.* North Oxford Academic Publishers Ltd, London, 1986.

[PRTT02] P. Pudlák, K. Rheinhardt, P. Tesson, and D. Thérien. On the multiparty communication complexity of regular languages. Draft available at www.cs.mcgill.ca/ ptesso, 2002.

[PST88] J-É. Pin, H. Straubing, and D. Thérien. Locally trivial categories and unambiguous concatenation. *J. Pure Applied Algebra*, 52:297–311, 1988.

[PW97] J. E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems*, 30:383–422, 1997.

[RTT98] J.-F. Raymond, P. Tesson, and D. Thérien. An algebraic approach to communication complexity. *Proc. ICALP'98, LNCS 1443*, pages 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.

[ST01] H. Straubing and D. Thérien. Regular languages defined by generalized first-order formulas with a bounded number of bound variables. In *Proc. STACS'01*, 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: Birkhäuser, 1994.

[STV01] T. Schwentick, D. Thérien, and H. Vollmer. Partially-ordered two-way automata: a new characterization of **DA**. In *Proc. DLT'01*, 37–56, 2001.

[Tes99] P. Tesson. An algebraic approach to communication complexity. Master's thesis, School of Computer Science, McGill University, 1999.

[Thé91] D. Thérien. Two-sided wreath product of categories. *J. Pure and Applied Algebra*, 74:307–315, 1991.

[TW98] D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation. In *Proc. ACM STOC'98*, pages 256–263, 1998.

[TW02] D. Thérien and T. Wilke. Nesting until and since in linear temporal logic. Submitted to STACS'02, 2002.