Minimal Non-Deterministic *xor* Automata

Jean Vuillemin, Nicolas Gamma
Ecole Normale Supérieure,
45 rue d'Ulm, 75230 Paris, France.

September 14, 2009

Keywords: Minimal Xor Automata, Mirror Reversal, Minimal Non-Deterministic Automata, Regular Languages

Abstract

A word $w \in \mathbf{B}^*$ in a regular language $L \subseteq \mathbf{B}^*$ is ***or*** accepted by a non-deterministic finite automaton $\cup$-NFA *iff* (if and only if) there is a path along w from an initial state to a final state. Word w is ***xor*** accepted by the same non-deterministic automaton $\oplus$-NFA *iff* the number of such pathes is odd. In the special case of deterministic (and more generally unambiguous) automata, accepting pathes are unique and both (***or*** /***xor*** acceptance) notions coincide.

While minimal size $\cup$-NFA exist to ***or*** accept regular language L, they are not unique [17] and no polynomial time algorithm is known to compute their size [10], or to decide equivalence between such classical NFA.

We show that the situation is exponentially better with (non-classical) $\oplus$-NFA . The dimension of language L is characterized (in equivalent ways) by $d = \mathbf{dim}\ L$: an integer which is finite *iff* L is regular. It relates to the states $s = |\mathbf{mda}\ L|$ in the *minimal deterministic automaton* for L by

$$\log_2(s) \leq d \leq s. \tag{1}$$

The dimension d is the minimal number of states in *any* $\oplus$-NFA for L. All minimal ($d = \mathbf{dim}\ L$ states) $\oplus$-NFA for L are *similar* to the *minimal deterministic automaton* $X = \mathbf{mxa}\ L$ (the unique such automaton in diagonal form). Automaton **mxa** can equivalently be derived from the **mda** through replacing the $s - d$ linearly dependent states by non-deterministic transitions. Each automaton X and $M = \mathbf{mda}\ L$ is a *strong normal form* ***SNF***: $L = L' \Leftrightarrow \mathbf{mxa}\ L = \mathbf{mxa}\ L' \Leftrightarrow \mathbf{mda}\ L = \mathbf{mda}\ L'$.

The dimension of a regular language L is invariant $\mathbf{dim}\ L = \mathbf{dim}\ \rho L$ by *mirror* (word reversal ρ). The minimal mirror $M = \mathbf{mxa}\ \rho L$ $\oplus$-NFA derives from the minimal $X = \mathbf{mxa}\ L$ in linear time. By contrast, the **mda** is sometimes exponentially related to its mirror.

Through matrix representation, we construct the minimal **mxa** L in time $O(n^3)$ and memory $O(n^2)$, from any representation of L by finite automata (or regular expression) with n states (or symbols). The complexity comparison favors **mxa** over **mda** in all regular operations (sum, concatenation, star, mirror, negation), with an exponential gain when $d = O(\log s)$.

The minimal **mxa** L is directly translated into efficient software & hardware recognizers for language L. In the special case of a finite language L, the synthesized memory-less (combinational) Boolean circuit is competitive

1 Introduction

Regular languages are at the core of computer science. In theory, Kleene's theorem [13] characterizes regular languages by equivalent representations as finite *Regular Expressions* RE, *Deterministic Finite Automaton* DFA and *Non-deterministic Finite Automaton* NFA. In practice, applications such as text/web searching, speech recognition/synthesis [14] and language understanding [9] all hinge on efficient manipulation of very large automata.

A DFA can be efficiently [**?**] reduced to its equivalent *Minimal Deterministic Automaton*. The MDA(L) is a *Strong Normal Form*: it is unique and characteristic of the regular language: $L = L' \Leftrightarrow MDA(L) = MDA(L')$.

The functions $\textsc{n} \in \mathbf{B}^* \mapsto \mathbf{N}^+$ and $\mathbf{w} \in \mathbf{N}^+ \mapsto \mathbf{B}^*$ are recursively defined by:

$$\begin{array}{rclrcl} w \in \mathbf{B}^* & & & n \in \mathbf{N}^+ & = & \{n \in \mathbf{N} : n > 0\} \\ \textsc{n}(\epsilon) & = & 1 & \mathbf{w}(1) & = & \epsilon \\ \textsc{n}(0 \cdot w) & = & 2\textsc{n}(w) & \mathbf{w}(2n) & = & 0 \cdot \mathbf{w}(n) \\ \textsc{n}(1 \cdot w) & = & 1 + 2\textsc{n}(w) & \mathbf{w}(1+2n) & = & 1 \cdot \mathbf{w}(n) \end{array} \quad (2)$$

They are respective inverses: $n = \textsc{n}\mathbf{w}(n) \in \mathbf{N}^+$ and $w = \mathbf{w}\textsc{n}(w) \in \mathbf{B}^*$.

$n = \textsc{n}(w)$	1	2	3	4	5	6	7	8	9	$\cdots$
$w = \mathbf{w}(n)$	ϵ	0	1	00	10	01	11	000	100	$\cdots$

Figure 1: One-to-one correspondence $\mathbf{B}^* \rightleftharpoons \mathbf{N}^+$.

	DFA	$\cup$-NFA	$\oplus$-NFA
$A \cup B$	*quad*	*lin*	*quad*
$A \oplus B$	*quad*	*quad*	*lin*
$A \cap B$	*quad*	*quad*	*quad*
$A \cdot B$	*quad*	*lin*	*lin*
A^*	*quad?*	*lin*	*lin*
ρA	*exp*	*lin*	*lin*

With such SNF [2], one maps different states of the MDA to different computer memory addresses, and equivalent states to a single shared memory address. Testing for state equivalence is reduced to testing equality of memory addresses, in constant time. The MDA is a natural implementation choice for applications which require a fast equivalence test between regular languages.

The number of states $s = |MDAL|$ is a key complexity measure for regular language L: integer $m = l2(s)$ gives the minimal memory (in bits) necessary (and sufficient) for any *Digital Synchronous Circuit* DSC to recognize L by hardware, and the least memory m required by any equivalent software recognition.

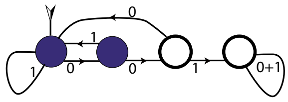
2 preliminaries

Very few **minimal** *non-deterministic finite automata* NFA are known, for good reasons [10]. Some were found through exhaustive search, up to 5 states [5, 18]; others through mathematical arguments [16]. Yet minimizing NFAs has remained computationally intractable [11] for over half a century [12].

We show that the situation is quite different when we consider the ***xor*** variant $\oplus$-NFA of the classical ***or*** acceptance by $\cup$-NFA (def. 2). In general, NFA a (resp. regular expression RE e) accepts word $w \in \mathbf{B}^*$ with multiplicity $a_+(w) \in \mathbf{N}$ (resp. $e_+(w) \in \mathbf{N}$). The multiplicity of w in automaton a is the number $a_+(w) = |\{w : i \overset{w}{\mapsto} f\}|$ of successful pathes (from initial i to final state f) along word w in the graph of a. The multiplicity of w in regular expression

e is the number $e_+(w)$ of ways to parse w according to e.

[18] $a_+ \in \mathbf{B}^* \mapsto \mathbf{N}$

Ex. 1 *Our regular work language* $\mathbf{R}$ *is (equally) defined by: the (un-ambiguous* $\mathbf{R} = \lambda_\cup E4 = \lambda_\oplus E4$*) regular expression* $E4 = (1 + 0 \cdot (1 + 0 \cdot 0))^* \cdot (\epsilon + 0)$*; the (equivalent un-ambiguous* $\mathbf{R} = \lambda_\cup A4 = \lambda_\oplus A4$*) automaton* $A4 = \mathbf{mda}\ \mathbf{R}$*, as presented by the matrices* $A4 = ([1000], \begin{bmatrix} 0100 \\ 0010 \\ 1000 \\ 0001 \end{bmatrix}, \begin{bmatrix} 1000 \\ 1000 \\ 0001 \\ 0001 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix})$ *or by the graph* 

.

3 Words & Languages

In a first reading, our theory is just presented for the binary alphabet $\mathbf{B} = \{0, 1\}$. Languages are sets of words $L \subseteq \mathbf{B}^*$. Words are binary $\mathbf{B}^* = \{\epsilon, 0, 1, 00, 10, 01 \cdots\}$. The empty word ϵ has length $0 = |\epsilon|$. The empty set of word is noted $\emptyset = \{\}$.

3.1 Binary Word Number

In fig. 1, word $w \in \mathbf{B}^*$ is numbered by $n = \textsc{n}(w) \in \mathbf{N}^+$, and conversely $w = \mathrm{w}(n)$. One-to-one correspondence (2) lets us freely import/export word/integer operations.

Length of integer $n \in \mathbf{N}^+$ is the length of word $\mathrm{w}n \in \mathbf{B}^*$: $|n| = |\mathrm{w}n| = \lfloor \log_2 n \rfloor$.

Order maps integer order $<$ to the (suffix) lexicographic order $\prec$ over words:

$$\textsc{n}w < \textsc{n}w' \Leftrightarrow w \prec w' \Leftrightarrow \begin{cases} |w| < |w'|, \\ |w| = |w'| \,\&\, (w = u \cdot 0 \cdot v,\ w' = u' \cdot 1 \cdot v) \text{ for } u, u', v \in \mathbf{B}^*. \end{cases}$$

Catenate positive integers $n, m \in \mathbf{N}^+$ by: $n \cdot m = \textsc{n}(\mathrm{w}n \cdot \mathrm{w}m)$.

Mirror $\rho n \in \mathbf{N}^+$ integer $n \in \mathbf{N}^+$ through word mirror: $\rho n = \textsc{n}\rho\mathrm{w}(n)$. Word mirror is defined by $\epsilon = \rho(\epsilon)$, $b = \rho b$ for $b \in \mathbf{B}$, and by $\rho(u \cdot v) = \rho v \cdot \rho u$ for $u, v \in \mathbf{B}^*$.

3.2 Arbitrary Alphabet

In a second reading, all results generalize to an alphabet $\Sigma = \{0, \cdots, k-1\}$ of arbitrary size $k > 0$. Words Σ^* are numbered in *suffix lexicographic order* by

$$\textsc{n}(w_1 \cdots w_n) = 1 + \sum_{j<n} (1 + w_{j+1}) k^j = k^n + \sum_{j<n} w_{j+1} k^j. \quad (3)$$

The language $O \subset \{0\}^*$ over the one-letter alphabet $\{0\}$ is equivalently defined by: $O = \{0^n : (n \bmod 7) \in \{0,1,2,4\}\}$; the graph of **mda** (O) is drawn above to the left, that of **mxa** $(O) = ([100], \begin{bmatrix} 010 \\ 001 \\ 101 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix})$ to the right.

Figure 2: Single letter alphabet, an example.

Eilenberg [6] calls (3) the *Russian correspondence* and (1) is the special case $k = 2$.

Before proceeding to the second reading, replace throughout: $\mathbf{B}$ by Σ, $+2$ by $+k$, $\div 2$ by $\div k$, and (1) by (3); once properly done, all claimed results hold for $k > 0$.

3.3 Language Table

Definition 1 *The* table *of a language* $L \subseteq \mathbf{B}^*$ *is the characteristic infinite bit-vector* $\tau L = [\textsc{n} \mapsto L^1_{\mathbf{N}}]$ *defined, for* $\textsc{n} \in \mathbf{N}^+$ *by* $L^1_{\mathbf{N}} = \begin{cases} 1 \Leftrightarrow \textsc{wn} \in L \\ 0 \Leftrightarrow \textsc{wn} \notin L \end{cases}$.

For example, the table of $\mathbf{R}$ (ex. 1) is $\tau\mathbf{R} = [111011110110111110001\cdots]$. Since $w \in L \Leftrightarrow L^1_{\mathbf{N}w} = 1$ (def. 1), table $\tau L = [\textsc{n} \mapsto L^1_{\mathbf{N}}]$ is a (first infinite) ***SNF***:

$$L = L' \Leftrightarrow \tau L = \tau L'.$$

The mirror table $\tau\rho L = [\textsc{n} \mapsto L^{\mathbf{N}}_1 = L^1_{\rho\mathbf{N}}]$ is a second infinite SNF, the truth-matrix a third.

3.4 Truth Matrix

Definition 2 *The* truth-matrix *of language* $L \subseteq \mathbf{B}^*$ *is* $\mu L = [L^r_c : r, c \in \mathbf{N}^+]$, *where*

$$L^r_c = \begin{cases} 1 \Leftrightarrow \mathbf{w}(\rho r \cdot c) \in L \\ 0 \Leftrightarrow \mathbf{w}(\rho r \cdot c) \notin L \end{cases} \quad \textit{for } r, c \in \mathbf{N}^+.$$

The first row $L^1_{1\cdots}$ of μL is table τL. The first column is the mirror $\tau\rho L = [L^{1\cdots}_1]^{\mathbf{tr}}$. Each entry in μL is equal to a single bit in the first row/column, by the rules

$$L^{b+2r}_c = L^r_{b+2c} \text{ for } b \in \mathbf{B}, \text{ or their equivalent } L^r_c = L^1_{\rho r \cdot c} = L^{\rho c \cdot r}_1. \tag{4}$$

It follows that $(\rho L)^r_c = L^c_r$ and the truth-matrix of the mirror is the transposed matrix:

$$\mu\rho L = (\mu L)^{\mathbf{tr}}. \tag{5}$$

$$
\mu\mathbf{R} = \begin{bmatrix} 1110111101\cdots \\ 1011101100\cdots \\ 1110111101\cdots \\ 0101010001\cdots \\ 1110111101\cdots \\ 1011101100\cdots \\ 1110111101\cdots \\ 1110111101\cdots \\ 0000000000\cdots \\ \cdots \end{bmatrix}. \tag{6}
$$

Figure 3: The truth-matrix of language $\mathbf{R}$ (ex. 1).

Row $L^2_{1\cdots} = \tau(0^- \cdot L)$ of μL is the table of the 0-suffix language $0^- \cdot L = \{w : 0 \cdot w \in L\}$. Column $L^{1\cdots}_2 = L \cdot 0^-$ tabulates the 0-prefix $L \cdot 0^- = \{w : w \cdot 0 \in L\}$. Def. 2 implies that:

Proposition 1 *Row $r = \text{N}u$ of matrix μL (def. 2) is the table $\tau(u^- \cdot L)$ of the $u = \text{W}n$* suffix *language $u^- \cdot L = \{w \in \mathbf{B}^* : \rho u \cdot w \in L\}$. Column $c = \text{N}w$ tabulates $L \cdot w^- = \rho w^- \rho L$.* $\diamond$

4 Regular Languages

Proposition 2 *Let $r = |\mathcal{R}(L)|$ count the (different) rows $\mathcal{R}(L) = \{L^r_{1\cdots} : r > 0\}$ in the truth-matrix μL of $L \subseteq \mathbf{B}^*$, and $c = |\mathcal{C}(L)|$ the columns $\mathcal{C}(L) = \{L^{1\cdots}_c : c > 0\}$. Then:*

- *L is regular* iff *$r < \infty$, and $r = |\mathbf{mda}\ L|$ is the number of states in* **mda** *L.*
- *L is regular* iff *$c < \infty$, and the mirror ρL has $c = |\mathbf{mda}\ \rho L|$ states.*

Proof: Automata theory ([6], chap. III) shows that the suffixes (prop. 1) of a language L are *finite* $\text{suffix}(L) = \{L^1 \cdots L^r\}$ *iff* L is regular. The suffixes of a regular L correspond one-to-one [6] to the $r = |\mathbf{mda}\ L|$ states and to the r (different) rows of matrix μL. By mirror (5), the states of **mda** ρL are the *prefixes* of L. $\diamond$

For example, the number of (different) rows in matrix $\mu\mathbf{R}$ (ex. 1, fig 3) is equal to $4 = |\mathbf{mda}\ \mathbf{R}|$. Language $M_n = \mathbf{B}^* 1 \mathbf{B}^n$ (sec. 5.4) has $|\mathbf{mda}\ M_n| = 2^{n+1}$ exponentially bigger than its mirror $|\mathbf{mda}\ \rho M_n| = n + 3$.

Note 1 *The truth-matrix becomes the Hankel matrix defined in [7] if we forget mirror ρ in (4). Rows of the Hankel matrix [7] thus permute those of our truth-matrix.*

4.1 Automata as Matrices

Non-empty NDAs are represented by graphs, or their adjacency matrices [6].

Definition 3 *Non-deterministic automaton $A = (n, I, T(), F)$ is presented by:*

- $n \in \mathbf{N}^+$ *is the number of states* $n = |A|$.
- $I \in \mathbf{N}[1, n]$ *is the row representing (the multiplicity of) initial states.*
- $T(b) \in \mathbf{N}[n, n]$ *are the transition matrices for* $b \in \mathbf{B}$.
- $F \in \mathbf{N}[n, 1]$ *is the column representing (the multiplicity of) final states.*

Transition $T(w) \in \mathbf{N}[n, n]$ *extends* $T()$ *from letters to words* $w \in \mathbf{B}^*$ *by:*

$$\begin{array}{rcl} T(\epsilon) & = & \mathbf{Id}_n = [\delta^r_c : 1 \leq r, c \leq n] \text{ is the } n \times n \text{ identity matrix;} \\ T(u \cdot v) & = & T(u) \cdot T(v) \text{ is the integer matrix product, for } u, v \in \mathbf{B}^*. \end{array} \tag{7}$$

Definition 4 *The multiplicity* $A(w) \in \mathbf{N}$ *of word* $w \in \mathbf{B}^*$ *in NDA* $A = (n, I, T(), F)$ *is defined by the matrix expression* $A(w) = \mathbf{trace}\,(I \cdot T(w) \cdot F)$.

Multiplicity $A(w) \in \mathbf{N}$ *counts [15, 6] the number of paths labeled by* $w \in \mathbf{B}^*$ *which connect initial to final states in the graph of* A.

Definition 5 *NDA* A ***or*** */* ***xor*** *accepts two (in general different) languages:*

$$\lambda_\cup A = \{w \in \mathbf{B}^* : A(w) > 0\} \text{ and}$$

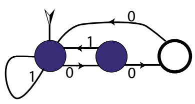
$$\lambda_\oplus A = \{w \in \mathbf{B}^* : 1 = A(w) \pmod 2\}.$$

Deterministic *automata are unambiguous [6]:* $\forall w \in \mathbf{B}^* : D(w) \in \{0, 1\}$. *For unambiguous automata, both acceptances coincide* $\lambda_\cup D = \lambda_\oplus D$. *It follows [6] that* ***xor*** *accepted languages coincide with Kleene's [13] classical (****or*** *accepted)* regular languages.

The ***or*** */****xor*** *languages accepted by* A *are different* iff *some word* $w \in \mathbf{B}^*$ *has an even non-zero multiplicity:* $A(w) = 2n$ *for* $n > 0$. *For example in* $C = \mathbf{mxa}\ \mathbf{R}$ *(ex. 4), word* 00 *has multiplicity* $2 = C(00)$ *and* $\mathbf{R} = \lambda_\oplus C \neq \lambda_\cup C = \mathbf{B}^*$.

Note 2 *The* ***or*** *acceptance (def. 5) is the classical one ([6], ch. VI). Linear automata are introduced by Schützenberger [15] and their dimension by Fliess [7] over any field* K. *The* ***xor*** *acceptance (def. 5) specializes [7] to the binary field* $K = \mathbf{F}_2$.

Ex. 2 (Minimal NDA for* R*) *The* **mda R** *(ex. 1) is* accessible, *but not co-accessible [6]. It is* trimmed *[6] to the equivalent un-ambiguous NDA* $\mathbf{R} = \lambda_\oplus A_3 = \lambda_\cup A_3$, *presented by matrices* $A3 = ([100], \begin{bmatrix} 010 \\ 001 \\ 100 \end{bmatrix}, \begin{bmatrix} 100 \\ 100 \\ 000 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix})$

or graph 

. *A computer enumeration of all languages recognized by NDAs with 2 states or less indicates that* **R** *is not one of them. So,* $A3$ *is a* minimal *NDA for* **R**.

4.2 Minimal Deterministic Automaton

Definition 6 *The* minimal access path *to row n of the truth-matrix μL of L is*

$$\boldsymbol{p}_L(n) = \begin{cases} n \Leftrightarrow \forall m < n: \ L^m_{1\cdots} \neq L^n_{1\cdots}, \\ m' \Leftrightarrow \exists m < n: \ L^m_{1\cdots} = L^n_{1\cdots} \text{ and } m' = \boldsymbol{p}_L(m). \end{cases}$$

By prop. 2, the minimal access paths $\boldsymbol{map}_L = \{p_1 \cdots p_s\} = \{\boldsymbol{p}_L(n) : \ n > 0\}$ of a regular language L canonically number the $s = |\mathbf{R}(L)|$ (different) rows of μL in increasing order $1 = p_1 < p_2 < \cdots < p_s$. The set is closed by suffix: $\forall k > 1, \exists i < k : \ p_i = p_k \div 2$.

Definition 7 (MDA) *Let L be a regular language, with (def. 6) $\boldsymbol{map}_L = \{p_1 \cdots p_s\}$. The* **mda** *$L = (s, I, T(), F)$ is defined, for $1 \leq r, c \leq s$ by:*

$$I_c = \delta^1_c, \ T(b)^r_c = \delta^{r'}_c \text{ where } p_{r'} = \boldsymbol{p}_L(b + 2p_r) \text{ for } b \in \mathbf{B}, \ F^r = L^{p_r}_1.$$

*$D =$ **mda** L is a* finite *($s = |\boldsymbol{map}_L|$ states)* ***SNF*** *for regular language $L = \lambda_\cup D = \lambda_\oplus D$.*

4.3 Accepted Language

Definition 8 *The* row-access-matrix *$R = \boldsymbol{ram}(A)$ of NDXA $A = (n, I, T(), F)$ is the matrix $R \in \mathbf{F}_2[\infty, n]$ whose r-th row is $R^r = I \cdot T(\mathrm{w}\rho r)$, for $r \in \mathbf{N}^+$. The c-th column of the* column-access-matrix *$C = \boldsymbol{cam}(A) = \boldsymbol{cam}(\rho A)^{\mathbf{tr}}$ of A is $C_c = T(\mathrm{w}c) \cdot F$.*

By defs. 5 and 8, the product $P = \boldsymbol{ram}(A) \cdot \boldsymbol{cam}(A)$ is the truth-matrix:

$$L^r_c = P^r_c = \mathbf{trace}\,(I \cdot T(\rho r) \cdot T(c) \cdot F) \pmod 2. \tag{8}$$

Proposition 3 *NDXA A accepts the regular language L* iff *the truth-matrix is the product of the access matrices $\mu L = \boldsymbol{ram}(A) \cdot \boldsymbol{cam}(A)$.* ◇

4.4 Atomic NDA Operations

The minimization procedure (alg. 1) relies on three atomic operations.

Definition 9 (Atomic Operations) *We map NDA $A = (n, I, T(), F)$ to the:*

Transposed *$A^{\mathbf{tr}} = (n, F^{\mathbf{tr}}, T'(), I^{\mathbf{tr}})$ where $T'(b) = T(b)^{\mathbf{tr}}$ for $b \in \mathbf{B}$.*

Similar *$P \cdot A \cdot P^- = (n, I \cdot P, T'(), P^- \cdot F)$ where $b \in \mathbf{B}$, $T'(b) = P^- \cdot T(b) \cdot P$ and $P, P^- \in \mathbf{N}[n, n]$ form an inverse $P \cdot P^- = \mathbf{Id}_n$ matrix pair.*

Reduced *$A/ = (n-1, I/, T()/, F/)$ where $M/$ represents the removal of row & column $n-1$ from matrix M.*

By (5), the transposed NDXA (def. 9) accepts the mirror: $A^{\mathbf{tr}}(w) = A(\rho w)$ for $w \in \mathbf{B}^*$. Similarity $B = P \cdot A \cdot P^-$ preserves multiplicity: $B(w) = A(w)$ for $w \in \mathbf{B}^*$. Reduction preserves acceptance in alg. 1, as columns n of I and $T(b)$ are null for $b \in \mathbf{B}$.

5 Linearly Independent Automata

By prop. 2, the truth-matrix μL of a regular language L has finitely many $r = |\mathbf{mda}\ L| = |\mathcal{R}(L)|$ different rows. The vector space $\mathbf{F}_2 \langle \mathcal{R}(L) \rangle$ generated by linear combinations (with coefficients in $\mathbf{F}_2$) of rows has a finite dimension: $d = \mathbf{dim}\ \mathbf{F}_2 \langle \mathcal{R}(L) \rangle \leq r$. The dual *space generated by columns has the same dimension: $d = \mathbf{dim}\ \mathbf{F}_2 \langle \mathcal{C}(L) \rangle \leq |\mathbf{mda}\ \rho L|$.*

Definition 10 *The dimension $d = \mathbf{dim}\ (L)$ of language $L \subseteq \mathbf{B}^*$ is the size of the largest $d \times d$ sub-matrix of μL which has an odd determinant.*

Proposition 4 *The dimension of a language L is finite* iff *L is regular. The dimension $d = \mathbf{dim}\ (L)$ of a regular language L is equal to:*

(i) The rank of matrix μL over $\mathbf{F}_2$.

(ii) The dimension of the vector space $\mathbf{F}_2 \langle \mathcal{R}(L) \rangle$ generated by the rows of μL.

(iii) The dimension of the vector space $\mathbf{F}_2 \langle \mathcal{C}(L) \rangle$ generated by the columns.

(iv) The dimension of the mirror language $d = \mathbf{dim}\ (\rho L)$.

Proof: Standard linear algebra shows that (i), (ii), (iii) are equivalent. (i) is equivalent to def. 10, and to (iv) by (5). $\diamond$

5.1 Regular Kernel

A refined version of def. 10 searches the truth-matrix μL of a regular language L for its (unique) upper & left-most sub-matrix with full rank over $\mathbf{F}_2$.

Definition 11 *Let L be a regular language of dimension $d > 0$ and truth-matrix μL. The* base rows *are defined by $\mathbf{B}_r(L) = \{L^{r_1}_{1\cdots} \cdots L^{r_d}_{1\cdots}\}$, where $r_1 = 1$ and r_j is the least integer such that row $L^{r_j}_{1\cdots}$ is linearly independent from the previous: $L^{r_j}_{1\cdots} \notin \mathbf{F}_2 \left\langle L^{r_1}_{1\cdots} \cdots L^{r_{j-1}}_{1\cdots} \right\rangle$.* Base columns *$\mathbf{B}_c(L) = \{L^{1\cdots}_{c_1} \cdots L^{1\cdots}_{c_d}\}$ are mirrors $\mathbf{B}_c(L) = \mathbf{B}_r(\rho L$.*

*Base rows (def. 11) form a subset $\mathbf{B}_r(L) \subseteq$ **map**$_L$ of minimal access paths (def. 6).*

Definition 12 *The* kernel *of a regular language L is the $d \times d$ sub-matrix which projects μL onto base rows & columns (def. 11) $\mathbf{B}_r(L)$ & $\mathbf{B}_c(L)$:*

$$\ker L \quad = \quad \begin{bmatrix} L^1_1 \cdots L^1_{c_d} \\ \cdots L^{r_i}_{c_j} \cdots \\ L^{r_d}_1 \cdots L^{r_d}_{c_d} \end{bmatrix}.$$

For example, $\ker \mathbf{R} = \begin{bmatrix} 11 \\ 10 \end{bmatrix}$ *(ex. 6) is the upper-most 2×2 sub-matrix of (6).*

It follows from defs. 12 and 11 that all sub-matrices $M = [L^{h_i}_{v_j} : 1 \leq i, j \leq k]$ of μL, either up ($\exists i : h_i < r_i$) or left ($\exists i : v_i < c_i$) of $\ker L$ *have an even determinant.*

Proposition 5 *Let L be a regular language with kernel matrix*

$$K = \ker L = [L^{r_i}_{c_j} : \ 1 \le i, j \le d].$$

1. *The determinant of K is odd:* $\det K \in 2\mathbf{N} + 1$.
2. *Matrix K has an inverse $K^- \in \mathbf{F}_2[d,d]$ such that* $\mathbf{Id}_d = K \cdot K^- \pmod 2$.
3. *The kernel of the mirror language is the transposed kernel matrix.* $\diamond$

Definition 13 *Let NDXA $A = (d, I, T(), F)$ accept $L = \lambda_\oplus A$. The row-access-base matrix $R = \boldsymbol{rab}(A) \in \mathbf{F}_2[d,n]$ is extracted from $\boldsymbol{ram}(A)$ at base rows (def. 11) $\mathbf{B}_r(L) = [r_1 \cdots r_d]$: $R^k_{1 \cdots n} = I \cdot T(\mathbf{w}\rho r_k)$ for $k \in [1 \cdots d]$. We define $\boldsymbol{cab}(A) = \boldsymbol{rab}(A^{\mathbf{tr}})^{\mathbf{tr}}$ by mirror.*

It follows from (8) and defs. 13, 12 that $\ker L = \boldsymbol{rab}(A) \cdot \boldsymbol{cab}(A)$.

5.2 Minimal *xor* Automata

Definition 14 *NDXA A is* minimal *if the number of states $n = |A| =$ $\mathbf{dim}$ L is equal to the dimension of the accepted language $L = \lambda_\oplus A$.*

Minimal automata are trim [6], and characterized as follows:

Theorem 1 *Let A be a NDXA with $n = |A|$ states, and $d =$ $\mathbf{dim}$ L be the dimension of the accepted language $L = \lambda_\oplus A$. Then:*

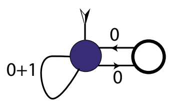
1. $n \ge d$.
2. *Automaton A is* minimal *$(n = d)$* iff $n = \mathbf{rank}\ \boldsymbol{ram}(A) = \mathbf{rank}\ \boldsymbol{cam}(A)$.
3. *A* minimal *NDXA A is* similar *$A' = P \cdot A \cdot P^-$ to any other* minimal *A', for some inverse $\mathbf{Id}_d = P \cdot P^- \pmod 2$ matrix pair.*

Proof: Prop. (3) shows that $\mu L = \boldsymbol{ram}(A) \cdot \boldsymbol{cam}(A)$. It follows that $d =$ $\mathbf{rank}\ \mu L \le \mathbf{rank}\ \boldsymbol{ram}(A) \le n$ and equality $n = d$ holds iff $n = \mathbf{rank}\ \boldsymbol{ram}(A) = \mathbf{rank}\ \boldsymbol{cam}(A)$.

Matrix $R = \boldsymbol{rab}(A) \in \mathbf{F}_2[d,n]$ (def. 13) has rank d. If $n = d$, it has an inverse. NDXA A is similar to $C = R^- \cdot A \cdot R$ and in turn to $A' = R' \cdot C \cdot R'^-$, for $R' = \boldsymbol{rab}(A')$. $\diamond$

Note 3 *Th. 1 is a special case ($K = \mathbf{F}_2$) of Fliess' theorem, which is proved in [7] for linear automata over arbitrary rings K.*

Ex. 3 *The minimal NDXA $A_2 = P \cdot C \cdot P^-$ is similar to $C =$ $\mathbf{mxa}$ L (ex. 4) through $P = \begin{bmatrix} 10 \\ 11 \end{bmatrix} = P^- \pmod 2$. $A_2 = (2, [10], \begin{bmatrix} 11 \\ 10 \end{bmatrix}, \begin{bmatrix} 10 \\ 00 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix})$*

is presented by matrices, or by graph 

. We note the ambiguity: $\mathbf{R} = \lambda_\oplus A_2 \neq \lambda_\cup A_2 = \mathbf{B}^*$.

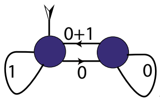
5.3 Canonical Form

Definition 15 *Let L be a regular language of* dimension $d = \mathbf{dim}\ L$ *and kernel* $K = \ker L$ *(def. 12). The canonical* $\mathbf{mxa}\ L = (d, I, T(), F)$ *is defined by:*

- $I = [10 \cdots 0] \in \mathbf{F}_2[d, 1]$
- $T(b) = [L_{c_j}^{b+2r_i} : 1 \le i, j \le d] \cdot K^-$ *for* $b \in \mathbf{B}$.
- $F = [K_1^1 \cdots K_d^1]^{\mathbf{tr}} \in \mathbf{F}_2[1, d]$.

Defs. 3 and 15 imply that $\mathbf{mxa}\ L$ *accepts* L*: for all* $w \in \mathbf{B}^*$

$$[L_{\mathbf{N}(w)}^1] = I \cdot T(w) \cdot K \cdot I^{\mathbf{tr}} = I \cdot T(w) \cdot F \pmod 2. \tag{9}$$

Ex. 4 *The canonical* $C = \mathbf{mxa}\ (\mathbf{R}) = (2, [10], \begin{bmatrix} 01 \\ 11 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix})$ *for* $\mathbf{R}$ *(ex. 1) has graph* 

. We note the ambiguity: $\mathbf{R} = \lambda_\oplus C \neq \lambda_\cup C = \mathbf{B}^*$.

Theorem 2 *Let L be regular and* $C = \mathbf{mxa}\ L$ *(def. 15) its canonical NDXA.*

(i) C *is a* finite *SNF for* L*:* $\mathbf{mxa}\ L = \mathbf{mxa}\ L' \Leftrightarrow L = L'$.

(ii) C *is the unique* minimal *automaton for* L *in diagonal form (def. 13):*

$$\boldsymbol{rab}(C) = \mathbf{Id}_d \text{ and } \boldsymbol{cab}(C) = \ker L. \tag{10}$$

Proof: Automaton $C = \mathbf{mxa}\ L$ *is* uniquely *constructed (def. 15) to accept language* $L = \lambda_\oplus C$*; thus* $\mathbf{mxa}\ L = \mathbf{mxa}\ L' \Leftrightarrow L = L'$*. By th. 1, all minimal automata for L are similar (def. 9), and* $C = \mathbf{mxa}\ L$ *is the unique one in diagonal form* $\boldsymbol{rab}(C) = \mathbf{Id}_d$*. It follows from* $\ker L = \boldsymbol{rab}(C) \cdot \boldsymbol{cab}(C)$ *that* $\boldsymbol{cab}(C) = \ker L$. ◇

5.4 Examples

Fig. 4 compares, over five classes of regular languages L, the sizes of four finite ***SNF*** *for L:* $\mathbf{mda}\ L$, $\mathbf{mda}\ \rho L$, $\mathbf{mxa}\ L$*, the 2-degree (note 4) of L, and the minimal memory* $mem(L) = |s|$ *(where* $s = |\mathbf{mda}\ L|$*) in minimal memory circuits [21] for L. In the corresponding circuit functions [19], operator* $\mathbf{z}$ *denotes the unit delay element,* $\mathbf{z}^n$ *the n bit shift-register, and* cm_n *the n bit counter modulo* 2^n.

Note 4 *For a binary alphabet, the third row* $L_{1\cdots}^3 = \tau(1^- L)$ *of the truth-matrix* μL *is identical to the* automatic sequence *associated to* $L \subseteq \mathbf{B}^*$ *by [1, 4]. Although* L^3 *is* not *a normal form for L, [4] shows that the* automatic sequence *of*

*language*L	*function*	**mda** L	**mda** ρL	**mxa** L	$\deg_2 L$	*memory*
$\mathbf{B}^*1$	x	3	2	2	0	0
$0^*1(\mathbf{B}^*0)^*$	$-x$	3	5	3	1	1
$\mathbf{B}^*1\mathbf{B}^n$	$\mathbf{z}^n x = 2^n x$	2^{n+1}	$n+3$	$n+2$	$n+2$	$n+1$
$\mathbf{B}^*1\mathbf{B}^n1\mathbf{B}^*$		$2^{n+1}+1$	$2^{n+1}+1$	$n+3$	$n+3$	$n+2$
$(0+(10^*)^{2^n})^*$	$cm_n(x)$	2^n	2^n	2^n	2^n-1	n

Figure 4: Sizes of canonical representations for some regular languages

a regular language is algebraic, i.e. root of some polynomial equation. It is further shown in [20, 21] that table $\tau L = L^1_{1\dots}$ *is algebraic* iff L *is regular, and that the characteristic* $P =$ **pol** (L) *is a finite* ***SNF****:* P *is the minimal polynomial which has table* $Y = \tau L$ *for root* $0 = P(Y) \pmod 2$.

Ex. 5 *The formal power series* $Y(z) = \sum_{n\in\mathbf{N}} \mathbf{R}_{n+1} z^n$ *for the truth table of language* $\mathbf{R}$ *is root* $0 = P(Y)$ *of the characteristic polynomial [20]* $P =$ **pol R***:*

$$P(Y) = 1 + Y + z(1+z)Y^2 + z^3Y^4.$$

The 2-degree of P *is* $2 = \log_2(4)$*. By [21], here it has maximal value* $2 =$ **dim R**.

6 Minimization Algorithm

Our algorithm proceeds à la *Brzozowski [3]: a reverse traversal followed by a direct one* $C = \boldsymbol{EDR}(\boldsymbol{EDR}(A^{\mathbf{tr}}))^{\mathbf{tr}}$*. Unlike [3], our minimal output* C *is non-deterministic.*

Algorithm 1 (Minimization Algorithm) *The input is a* $n = |A|$ *states NDXA* A *which accepts the regular language* $L = \lambda_\oplus A$.

1. *Eliminate dependent columns of* A *& compute* $B = \boldsymbol{EDR}(A^{\mathbf{tr}})$ *by alg. 2.*
2. *Eliminate dependent columns of* B *& compute* $C = \boldsymbol{EDR}(B^{\mathbf{tr}})$ *by alg. 2.*

Output $C =$ **mxa** L*: the canonical automaton (def. 15).*

Ex. 6 *Alg. 1 reduces* $A =$ **mda R** *(ex. 1) to* $C =$ **mxa R** *(ex. 4):*

$$B = \boldsymbol{EDR}(A^{\mathbf{tr}}) = (2, [11], \begin{bmatrix} 01 \\ 11 \end{bmatrix}, \begin{bmatrix} 11 \\ 00 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}),$$

$$C = \boldsymbol{EDR}(B^{\mathbf{tr}}) = (2, [10], \begin{bmatrix} 01 \\ 11 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}).$$

Theorem 3 *The canonical* $C =$ **mxa** L *of a regular language* $L = \lambda_\oplus A$ *is computed, from any NDXA* A *with* $n = |A|$ *states, by alg. 1 in time* $O(n^3)$ *and memory* $O(n^2)$.

Proof: *By lemma 3,* $B = \boldsymbol{EDR}(A^{\mathbf{tr}})$ *is a mirror (*$\lambda_\oplus A = L$, $\lambda_\oplus B = \rho L$*) with* $r = |B| = \mathbf{rank}\ \boldsymbol{ram}(A^{\mathbf{tr}}) = \mathbf{rank}\ \boldsymbol{cam}(A) = \mathbf{rank}\ \boldsymbol{ram}(B) = \mathbf{rank}\ \boldsymbol{cam}(B^{\mathbf{tr}})$ *states.*

By lemma 3, $C = \boldsymbol{EDR}(B^{\mathbf{tr}})$ *is equivalent* $\lambda_\oplus C = L$*, with* $c = \mathbf{rank}\ \boldsymbol{ram}(B^{\mathbf{tr}}) = \mathbf{rank}\ \boldsymbol{cam}(B)$ *states. Since* $\boldsymbol{cam}(B^{\mathbf{tr}})$ *has full-rank* r *and* $\mu L = \boldsymbol{ram}(B^{\mathbf{tr}}) \cdot \boldsymbol{cam}(B^{\mathbf{tr}})$ *by prop. 3, the rank* $c = \mathbf{rank}\ \boldsymbol{ram}(B^{\mathbf{tr}}) = \mathbf{rank}\ \mu L$ *is equal to* $d = \mathbf{dim}\ L$ *(def. 10).*

By lemma 3, B *is computed in time* $O(rn^2)$ *and memory* $O(n^2)$*, and* C *in time* $O(dr^2)$*. Total time is* $O(n^3)$*, and memory* $O(n^2)$ *is shared between the two passes.* ◇

6.1 Eliminate Dependent Rows

Algorithm $\boldsymbol{EDR}$ *eliminates linearly dependent rows/states from NDXA* A*.*

Algorithm 2 (EDR) *The input is NDXA* $A = (n, I, T(), F)$*. The output is the empty automaton* $\emptyset$ *(of size* $0 = |\emptyset|$*) if* $I = [0^n]$ *is null. Otherwise, we first compute* $(r, P, B) = \boldsymbol{HNF}(1, 0, 0, [0], A)$ *by alg. 3. We reduce* $B = (n, I', T'(), F')$ *to its first* r *rows & columns and output* $C = (r, I'', T''(), F'')$*, where* $I'' = I'_{1\cdots r}$ *and likewise for* F'' *and* $T''(b)$*,* $b \in \mathbf{B}$*.*

Lemma 3 shows that $\boldsymbol{EDR}$ *reduces* A *to* $r = |C| = \mathbf{rank}\ \boldsymbol{ram}(A) \leq n = |A|$ *states.*

6.2 Hermite Normal Form

Procedure $\boldsymbol{HNF}$ *(alg. 3) is a variant over* $\mathbf{F}_2$ *of Hermite's [8] normal form .*

Algorithm 3 (HNF) *The input is three numbers* b, j, k*, integer vector* $P = [p_1 \cdots p_k]$ *and NDXA* $A = (n, I, T_0, T_1, F)$*, all satisfying (11). The output is recursively computed:*

$$\boldsymbol{HNF}(b,j,k,P,A) = \begin{cases} \textit{if } (j > k) \textit{ then } (k, P, A) \textit{ else} \\ \textit{if } (\exists m \geq k:\ 1 = J_m) \textit{ then } \boldsymbol{HNF}(b', j', k+1, Q, B) \\ \textit{else } \boldsymbol{HNF}(b', j', k, P, A), \end{cases}$$

where $J = \textit{if } j = 0 \textit{ then } I \textit{ else } T_b^j$*, and* $T_b^j = T_{1\cdots n}^j$ *is row* j *of* $T = T(b)$*;*

$(b', j') = \textit{if } b = 0 \textit{ then } (1, j) \textit{ else } (0, j+1)$*;*

$Q = [p_1 \cdots p_k\, q]$ *for* $q = b + 2p_j$*;*

NDXA $B = U \cdot A \cdot V$ *is similar to* A*, for matrices* $(U, V) = \mathbf{P}(k, m, J)$ *given by alg. 4.*

Lemma 3 shows that $\boldsymbol{HNF}$ *reduces* A *to* $r = |C| = \mathbf{rank}\ \boldsymbol{ram}(A)$ *states in time* $O(rn^2)$ *and memory* $O(n^2)$*. Efficiency comes from the sparseness of the chosen pivots.*

Algorithm 4 (Pivot) *The input is two numbers* k *and* m *such that* $k \leq m \leq n$*, and a row-matrix* $J \in \mathbf{F}_2[1, n]$ *such that* $1 = J_m$*. The output* $(U, V) = \mathbf{P}(k, m, J)$ *is the pair of inverse matrices* $U = P \cdot N$ *and* $V = N \cdot P$ *defined by:*

- *Matrix P permutes columns k and m.*
- *Matrix N derives from the permuted $R = J \cdot P$ (with $1 = R_k$) by:*

$$N^r_c = \textit{if } (r = k) \textit{ then } R_c \textit{ else } \delta^r_c, \textit{ for } 1 \leq r, c \leq n.$$

6.3 Analysis of the Algorithms

The minimization algorithm 1 is presented top-down*. It's correctness proof is best understood* bottom-up*, in order: pivot (alg. 4),* ***HNF*** *(alg. 3) and* ***EDR*** *(alg. 2).*

Lemma 1 *For $J \in \mathbf{F}_2[1 \cdots n]$ and m such that $1 = J_m$, the matrices $(U, V) = \mathbf{P}(k, m, J)$ computed by alg. 4 are inverse: $\mathbf{Id}_n = U \cdot V \pmod 2$. Pivot $V = J \cdot U$ is diagonal:*

$$V = [\delta^k_c : 1 \leq c \leq n].$$

Proof: Permutation P and normalization N matrices in alg. 4 are self-inverse: $\mathbf{Id}_n = P \cdot P = N \cdot N \pmod 2$. So $\mathbf{Id}_n = U \cdot V \pmod 2$ and pivot $V = R \cdot N = J \cdot U$ is diagonal. $\diamond$

Lemma 2 *The inputs $(b, j, k, P = [p_0 \cdots p_k], A = (n, I, T_0, T_1, F))$ to alg. 3 satisfy:*

$$\begin{array}{l} b \in \mathbf{B},\ j \leq k+1 \textit{ and } k \leq n = |A|; \\ \textit{sequence } P = [p_0 \cdots p_k] \textit{ is increasing, } 0 = p_0 \textit{ and } p_i < p_{i+1} \textit{ for } i < k, \\ \textit{and } P \textit{ is closed by suffix: } \forall j > 0, \exists i < j : p_i = p_j \div 2. \\ \textit{for } q = b + 2p_j : \mathbf{v}(q-1) = \mathbf{h}(k); \\ \textit{row } \mathbf{r}(p_i) = [\delta^i_c : 1 \leq c \leq n] \textit{ is diagonal, for } i \in [1 \cdots k]. \end{array} \tag{11}$$

In (11), we let $\mathbf{v}(k) = \mathbf{F}_2 \langle \mathbf{r}(1) \cdots \mathbf{r}(k) \rangle$ and $\mathbf{h}(k) = \mathbf{F}_2 \langle \mathbf{r}(p_1) \cdots \mathbf{r}(p_k) \rangle$, where

$$\mathbf{r}(k) = I \cdot T(\mathrm{w}\rho k) \pmod 2 \in \mathbf{F}_2[1, n] \textit{ for } k \in \mathbf{N}^+.$$

Proof: Conditions (11) are trivially satisfied at the initial call $\boldsymbol{HNF}(1, 0, 0, [0], A)$, and they remain invariant through subsequent recursions: For $q = b + 2p_j$ the row J computed by alg. 3 is $J = \mathbf{r}(q)$ by (11). Condition $\exists m \geq k : 1 = J_m$ is thus equivalent to $J \notin \mathbf{h}(k)$. Condition $J \in \mathbf{h}(k)$ implies that (11) holds for (b', j', k, P, A), since $\mathbf{v}(q'-1) = \mathbf{h}(k)$ with $q' = b' + 2p_{j'}$ and (b', j') are the successors of (b, j) in alg. 3. Condition $J \notin \mathbf{h}(k)$ implies that (11) holds for $(b', j', k+1, Q, B)$, by alg. 3 and lemma 1. $\diamond$

Lemma 3 *The output $C = \boldsymbol{EDR}(A)$ of alg. 2 is an equivalent NDXA $\lambda_\oplus C = \lambda_\oplus A$ of size $r = |C| = \mathbf{rank}\ \boldsymbol{ram}(A)$. It is computed in $O(rn^2)$ bit-operations and memory $O(n^2)$.*

Proof: By lemma 2, the intermediate results $(r, [p_0 p_1 \cdots p_r], B) = \boldsymbol{HNF}(1, 0, 0, [0], A)$ in alg. 2 are such that: $r = \mathbf{rank}\ \boldsymbol{ram}(A) \leq n = |B| = |A|$, and B is similar to A. The row-access-matrix $R = \boldsymbol{ram}(B)$ has rank $r = \mathbf{rank}\ R$, since

R^{p_i} is diagonal for $i \in [1 \cdots r]$. All columns $r+1$ through n of R are null: $\forall n > 0, c > r :\ R^n_c = 0$. It follows that B is equivalent to the reduced output C, without (useless) rows & columns $[r+1 \cdots n]$.

There are two types of recursive calls to ***HNF*** *in alg. 3: the first increases variable k, the second only increases $b+2j$. There are r calls of the first type, and at most $2r$ calls of the second, since $b+2j \leq 2r$. In both cases, vector J is computed in n bit-operations. In addition, similarity $U \cdot A \cdot V$ is computed r times, in $O(n^2)$ bit-operations due to the sparse nature of the pivot matrices $U = N \cdot P$ and $V = N \cdot P$. Altogether, alg. 3 runs in time $O(rn^2)$. All successively computed (similar) NDXA have size n. All can share space: similarities are computed* in-place, *within a common pre-allocated $O(n^2)$ memory.* $\diamond$

7 Conclusion

The **mxa** *is an attractive alternative to the* **mda** *: never bigger, sometimes exponentially smaller. In theory, the* **mxa** *is superior to the* **mda** *in order to represent, recognize, construct and automatically process regular languages. In practice, this point has to be validated by real-world applications, possibly first with Boolean functions.*

While the canonical **mxa** *has a minimal number of states, it need not have a minimal number of edges (ex. 6 and 4). Finding a NDXA with minimal number of states & edges in polynomial time remains an open problem.*

Another question relates the structure of the characteristic 2-polynomial $P =$ **pol** *L [20, 21] to that of* **mxa** *L, for a regular language L. It is shown in [21] that the 2-degree of P (ex. 5) is bounded by the dimension of L (ex. 5, fig. 4). Unlike the* **mxa** *, no efficient algorithms is known for constructing or minimizing characteristic 2-polynomials.*

Acknowledgments *This work has benefited from valuable contributions by Jean-Baptiste Note, and Gérard Huet.*

References

[1] *J. P. Allouche. Automates finis en théorie des nombres.* Expositiones Mathematicae, *5:239–266, 1987.*

[2] *R. E. Bryant. Symbolic boolean manipulations with ordered binary decision diagrams.* ACM Comp. Surveys, *24:293–318, 1992.*

[3] *J.A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events.* Mathematical theory of automata, *Brooklyn, Polytechnic Institute of Brooklyn, New York, (Symposia Series, 12):529–561, 1962.*

[4] *G. Christol, T. Kamae, M. Mendès France, and G. Rauzy. Suites algèbriques, automates et substitutions.* Bull. Soc. Math. France, *pages 401–419, 1980.*

[5] *Michael Domaratzki, Derek Kisman, and Jeffrey Shallit. On the number of distinct languages accepted by finite automata with n states.* Journal of Automata, Languages and Combinatorics*, 7(4):469–486, 2002.*

[6] *S. Eilenberg.* Automata, Languages, and Machines*, volume I. Academic Press, 1974.*

[7] *M. Fliess. Matrices de* Hankel*.* J. Math. pures et appl.*, 53:197–224, 1974.*

[8] *C. Hermite. Sur l'introduction des variables continues dans la theorie des nombres.* J. Reine Angew. Math.*, 41:191–216, 1851.*

[9] *G. Huet.* Topics in Sanskrit Computational Linguistics*, volume 5402. Springer-Verlag Lecture Notes, eds. G. Huet A. Kulkarni P. Scharf, 2008.*

[10] *Tao Jiang and B. Ravikumar. Minimal nfa problems are hard.* SIAM Journal on Computing*, 22(6):1117–1141, 1993.*

[11] *Tao Jiang and Bala Ravikumar. Minimal nfa problems are hard. In* ICALP*, pages 629–640, 1991.*

[12] *T. Kameda and P. Weiner. On the state minimalization of nondeterministic finite. automata.* IEEE Transactions on Computers*, C-19(7):617–627, September 1970.*

[13] *S. C. Kleene. Representation of events in nerve nets and finite automata. eds. C. E. Shannon J. McCarthy, Princeton U. Press.*

[14] *M. Mohri.* Weighted automata algorithms - in Handbook of weighted automata, eds. M. Droste and W. Kuich and H. Vogler*. Springer, 2009.*

[15] *M. P. Schützenberger. On the definition of a family of automata.* Information and Control*, 4:245–270, 1961.*

[16] *Hellis Tamm and Esko Ukkonen. Bideterministic automata and minimal representations of regular languages. In* CIAA*, pages 61–71, 2003.*

[17] *Hellis Tamm and Esko Ukkonen. Bideterministic automata and minimal representations of regular languages.* Theor. Comput. Sci.*, 328(1-2):135–149, 2004.*

[18] *Lynette van Zijl. Succinct descriptions of regular languages with binary +-nfas. In* CIAA*, pages 72–82, 2003.*

[19] *J. Vuillemin. On circuits and numbers.* IEEE Trans. on VLSI*, 43:8:868–879, 1994.*

[20] *J. Vuillemin. Finite circuits are characterized by 2-algebraic truth-tables. In* Advances in Computing Science - ASIAN 2000*, volume 1961 of* L.N.C.S.*, pages 1–12. Springer-Verlag, 2000.*

[21] *J. Vuillemin. Digital algebra and circuits. In N. Dershowitz, editor,* Verification – Theory & Practice*, volume 2772 of* L. N. C. S.*, pages 733 – 746. Springer-Verlag, 2004. Essays Dedicated to Zohar Manna on the Occasion of his 64th Birthday.*