Efficient Equivalence and Minimization for Non Deterministic Xor Automata

Jean Vuillemin, Nicolas Gama

Efficient Equivalence and Minimization for Non Deterministic Xor Automata

Jean Vuillemin, Nicolas Gama
Ecole Normale Supérieure, Paris, France

December 18, 2009

Abstract

A word $w \in L$ in a regular language $L \in \mathcal{RL}$ is *or-accepted* $w \in \Lambda_\cup(A)$ by a non-deterministic finite *or-automaton* $A \in$ NFA if there exists a path along w from some initial state to some final state in A. The same word w is *xor-accepted* $w \in \Lambda_\oplus(A)$ by the same non-deterministic finite *xor-automaton* $A \in$ NXA if the number of accepting pathes is *odd*.

In the *deterministic* case $A \in$ DFA, accepting pathes are unique and both definitions coincide: $\Lambda_\cup(A) = \Lambda_\oplus(A)$. They differ the general *non-deterministic* case, when $\Lambda_\oplus(A) \subset \Lambda_\cup(A)$.

No polynomial time algorithm is known to minimize NFAs or to compute their equivalence. By contrast, we present a cubic time algorithm to reduce a xor-automaton $A \in$ NXA to a minimal form $M = \mathrm{MXA}(A)$ which accepts $\Lambda_\oplus(M) = \Lambda_\oplus(A)$, within the least possible number of states. It is a *finite strong normal form* ***SNF***: $\Lambda_\oplus(A) = \Lambda_\oplus(A') \Leftrightarrow \mathrm{MXA}(A) = \mathrm{MXA}(A')$ and automata equivalence is efficiently decided through reduction to the ***SNF***. The minimal number of states $d = |\mathrm{MXA}(L)|$ is equal to the *dimension* $d = \dim(L)$ of the language L, and the dimension $d \in \mathbf{N}$ is finite if and only if $L \in \mathcal{RL}$ is regular.

The only other known finite strong normal form for a regular language L is the classical minimal deterministic automaton $\mathrm{MDA}(L)$, whose number s of states can be exponentially larger than the dimension: $d \leq s \leq 2^d$.

Besides efficient minimization and equivalence test (which NFAs don't seem to possess), we show that all other natural language operations which are efficiently handled by NFAs are just as efficiently handled by NXAs.

Contents

1 Introduction **3**
- 1.1 Contributions . . . 4
- 1.2 Example X . . . 4
- 1.3 Plan . . . 5

2 Regular Languages **5**
- 2.1 Letters and Words . . . 5
- 2.2 Languages . . . 7
- 2.3 Truth Table and Matrix . . . 7
- 2.4 Regular Languages . . . 9
- 2.5 Minimal Deterministic Automaton MDA . . . 10

3 Regular Dimension **10**
- 3.1 Regular Base . . . 11
- 3.2 Minimal Xor Automata . . . 12
- 3.3 From MDA to MXA . . . 12

4 Non Deterministic Xor Automata **13**
- 4.1 Multiple Acceptance . . . 14
- 4.2 Deterministic Automata . . . 16
- 4.3 Atomic NXA Operations . . . 16

5 Minimizing Nondeterministic Xor Automata **17**
- 5.1 Linearly Reduced Automata . . . 18
- 5.2 Efficient Minimization . . . 18
- 5.3 Efficient NXA Operations . . . 21

6 Conclusion **22**

	DFA	NFA	NXA	MXA
$A \neq B$	*quadratic*	*exponential*	*cubic*	*constant*
$minimize(A)$	$lin - log^{(1)}$	*exponential*	*cubic*	*constant*
$\neg A$	*linear*	*linear*	*linear*	*quadratic*
ρA	*exponential*	*linear*	*linear*	*cubic*
A^*	*exponential*	*linear*	*linear*	*cubic*
$A \cdot B$	*exponential*	*linear*	*linear*	*cubic*
$A \cup B$	*quadratic*	*linear*	*quadratic*	*cubic*
$A \oplus B$	*quadratic*	*quadratic*	*linear*	*cubic*
$A \cap B$	*quadratic*	*quadratic*	*quadratic*	*cubic*

(1) Minimizing DFAs has complexity $O(n \log(n))$ by Hopcroft's algorithm [13].

Figure 1: Complexity chart for language operations, over 4 species of automata.

1 Introduction

Regular Languages RL are the hard-core of theoretical *Computer Science*. The efficient manipulation of very large automata plays a key role in more recent applications than we care to reference here - from circuits and software verification to searching huge web indexes, through logic, linguistics, speech recognition/synthesis [23] and many others.

Kleene's theorem broadly characterizes regular languages by equivalent finite representations as: *Regular Expressions*; *Deterministic* DFA and *Nondeterministic* NFA *Finite Automata*; *Finite Circuits* and *Finite Kahn Networks* [17], i.e. communicating programs which only possess a finite local memory.

Since all of the above representations are finite, there exist *minimal size representations* among each. The tricky part is to effectively find them! Among the above representations, DFAs are the only one which is known to have an easily computable minimal form, namely the *Minimal Deterministic Automaton* MDA. The MDA is a finite *Strong Normal Form* ***SNF***: it is unique and characteristic of the regular language: $L = L' \Leftrightarrow MDA(L) = MDA(L')$. With such ***SNF***s [4, 21], one maps different states of the MDA to different computer memory addresses, and equivalent states to a single shared memory address. Testing for equivalence is reduced to testing equality of addresses, in constant time.

While that most language operations are efficiently performed (fig. 1) on MDAs, it is well-known that the number of states in DFAs is, in general [16], exponential in the number of states for equivalent NFAs. So, some practical applications derived from very large automata rely on nondeterministic NFAs rather deterministic ones. Indeed, most language operations are efficiently performed (fig. 1) on NFAs, except for two: minimization and testing for equivalence which have a exponential worst case complexity over NFAs.

Few explicitly *proven minimal* NFAs are known [16]. Some are found through exhaustive search (up to 5 states [7, 30], see X. 2), and others through mathe-

The language $O \subset O^*$ over the one-letter alphabet $\{0\}$ is equivalently defined by: $O = \{0^n : (n \bmod 7) \in \{0, 1, 2, 4\}\}$. The graph of its minimal deterministic automaton MDA(O) is drawn above to the left. The graph of its minimal nondeterministic Xor automaton MXA(O) is to the right.

Figure 2: Single letter alphabet, an example from [32].

matical arguments [29]. Yet minimizing NFAs or computing their equivalence has remained computationally intractable [15] for well over half a century [18].

1.1 Contributions

Our contribution is to show that, unlike NFAs, non-deterministic Xor automata NXA [8, 31] have a unique minimal form which can be effectively computed through linear algebra. Our minimization algorithm proceeds in two mirror passes; it has a cubic bit-level time complexity, and it only requires quadratic memory, once we compute in-place.

The result is new for alphabet sizes over two letters. It is known for the single letter alphabet, where minimization is achieved in [32] by reduction to *linear feed-back shift registers* and the Berlekamp-Massey algorithm [22] (fig. 2).

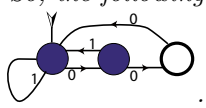
We finally show that all the language operations which are efficiently performed over NFAs are also efficiently performed over NXAs (fig. 1), while keeping an efficient equivalence test and minimization procedure.

1.2 Example X

We illustrate the various constructions in this paper through the example of language $\mathbf{X} \subset \{0, 1\}^*$, which is equivalently defined here in various ways.

X 1 *Our regular example language is first defined by* $\mathbf{X} = \Lambda(REX)$*, from the regular expression:* $REX = (1 + 0 \cdot (1 + 0 \cdot 0))^* + (1 + 0 \cdot (1 + 0 \cdot 0))^* \cdot 0$*. Note that the regular expression REX is un-ambiguous, and both its OR and XOR interpretations (see sect. 4.1) coincide:* $\mathbf{X} = \Lambda_\cup(REX) = \Lambda_\oplus(REX)$.

X 2 (Minimal NFA for X) *An enumeration of all languages recognized by* NFA*s with 2 states or less shows that* $\mathbf{X}$ *is not one of them. So, the following 3 states* NFA *for* $\mathbf{X}$ *is* minimal*:*



.

1.3 Plan

Section 2 introduces some notations and concepts from the classical theory of regular languages, including the MDA. Section 3 defines the dimension of a regular language, and its MXA. Section 4 introduces nondeterministic Xor automata by their matrix representations. Section 5 characterizes minimal NXAs, and present efficient algorithms for operating on MXAs, and we conclude in section 6.

2 Regular Languages

The purpose of this section is to establish the terminology and notations used here for *classical* automata theory.

2.1 Letters and Words

Our *alphabet* $\mathcal{A} = \{0 \cdots \alpha - 1\}$ has $\alpha = |\mathcal{A}| > 0$ *letters*: $l \in [0 \cdots \alpha - 1]$. Note that letters are ordered by $0 < \cdots < \alpha - 1$, when $\alpha > 1$.

Note: Our reader will focuss on the binary alphabet $\mathcal{A} = \mathbf{B} = \{0, 1\}$, to gain insights and loose no generality. The special case of the unary alphabet $\alpha = 1$ is treated in details by [32].

A *word* $u \in \mathcal{A}^n$ has $n = |u|$ letters $u_i \in \mathcal{A}$: $u = u_1 \cdot \cdots \cdot u_n$.

The *empty* word $\epsilon \in \mathcal{A}^0$ is the one and only of length $0 = |\epsilon|$.

Finite words $\mathcal{A}^* = \bigcup_{n \in \mathbf{N}} \mathcal{A}^n$ form a *Monoid* under *concatenation*:

$$\forall u, v, w \in \mathcal{A}^* : \; u = \epsilon \cdot u = u \cdot \epsilon \text{ and } (u \cdot v) \cdot w = u \cdot (v \cdot w).$$

The *mirror* $\rho(u_1 \cdots u_n) = u_n \cdots u_1$ reverses the order of letters in $u \in \mathcal{A}^n$.

Note that $|u \cdot v| = |u| + |v|$ and $\rho(u \cdot v) = \rho(v) \cdot \rho(u)$ for all $u, v \in \mathcal{A}^*$.

The *null* word $\emptyset$ is such that: $\forall u \in \mathcal{A}^* : \; \emptyset = \emptyset \cdot u = u \cdot \emptyset$; as for its length, we find that $|\emptyset| = \log_2(0) = -\infty$.

Total Orders on Words Lexicographic total orders on finite words are well-know, but the terminology varies between authors - e.g. [19, 9].

Definition 1 *The* prefix order $\prec$ *is defined on words* $u, v \in \mathcal{A}^*$ *by:*

1. *If* $|u| < |v|$ *then* $u \prec v$.
2. *If* $|u| = |v|$, *extract the first letters* $u = l \cdot u', v = l' \cdot v'$ *with* $l, l' < \alpha$, *and let* $u \prec v \Leftrightarrow (l < l') \cup ((l = l') \cap (u' \prec v'))$.

   We define: $u \preceq v \Leftrightarrow (u \prec v) \cup (u = v)$.
3. *The* suffix order $<$ *is given by mirror:* $\forall u, v \in \mathcal{A}^* : \; u < v \Leftrightarrow \rho\, u \prec \rho\, v$. *We define:* $u \leq v \Leftrightarrow (u < v) \cup (u = v)$.

Both orders are total and well-founded [19]. For the binary alphabet, we find:

$$\emptyset \prec \epsilon \prec 0 \prec 1 \prec 00 \prec 01 \prec 10 \prec 11 \prec 000 \prec 001 \prec \cdots$$

$$\emptyset < \epsilon < 0 < 1 < 00 < 10 < 01 < 11 < 000 < 100 < \cdots$$

Using the integer order symbol $<$ rather than, say $<'$ to denote the suffix order is justified by showing next that: $\forall u, v \in \mathcal{A}^* : \; u <' v \Leftrightarrow \mathrm{N}(u) < \mathrm{N}(v) \Leftrightarrow \rho\, u \prec \rho\, v$.

Word Number Numbering words by successive integers $\mathrm{N}(\emptyset) = 0$, $\mathrm{N}(\epsilon) = 1, \cdots$ establishes a one-to-one correspondence, between finite (or null) words $\mathcal{A}^* \cup \emptyset$ and integers $\mathbf{N}$, with both of the above orders.

Definition 2 (Suffix Number) *The* suffix number $\mathrm{N}(u) \in \mathbf{N}$ *of a finite (or null) word* $u \in W = \mathcal{A}^* \cup \emptyset$ *counts the number* $\mathrm{N}(u) = |\{v \in W : v < u\}|$ *of words which precede* u *in suffix order. Each finite word* $u \in \mathcal{A}^*$ *has a unique number* $n = \mathrm{N}(u) \in \mathbf{N}^+$*, and conversely we let* $\mathrm{W}(n) = u$.

Note: An explicit formula for word $u = u_0 \cdots u_{l-1} \in \mathcal{A}^l$ of length $l = |w|$ is

$$n = \mathrm{N}(u) = 1 + \sum_{k<l} (1 + u_k) 2^k = 1 + \frac{b^l - 1}{b - 1} + \sum_{k<l} u_k 2^k.$$

So, n is one off ($u\eta = \mathrm{N}(u) - 1$) the *reversed bijective interpretation*: [9], ch. V.

In the binary alphabet $2 = \alpha$, word $w = w_0 \cdots w_{l-1} \in \mathbf{B}^l$ is numbered by the integer $n = \mathrm{N}(w) \in \mathbf{N}^+$ whose binary representation $bin(n) = w \cdot 1$ concatenates 1 to the right of w, i.e. $n = \mathrm{N}(w) = \sum_{k<l} w_k 2^k + 2^l$. Conversely, we write $n = \sum_{k<l+1} n_k 2^k$ in binary to extract $bin(n) = n_0 \cdots n_{l-1} \cdot 1$; we remove the most significant 1 bit and keep the rest: $\mathrm{W}(n) = w \cdot 1^- = n_0 \cdots n_{l-1}$.

In general, def. 2 establishes an isomorphism between finite words $u, u' \in \mathcal{A}^* = \mathrm{W}(\mathbf{N}^+)$ and positive integers $n, n' \in \mathbf{N}^+ = \mathrm{N}(\mathcal{A}^*)$, such that

$$u < u' \Leftrightarrow \mathrm{N}(u) < \mathrm{N}(u') \text{ and } n < n' \Leftrightarrow \mathrm{W}(n) < \mathrm{W}(n').$$

In this paper, suffix numbering lets us use integer $n = \mathrm{N}(u) \in \mathbf{N}^+$ rather than word $u = \mathrm{W}(n) \in \mathcal{A}^*$ as an index to tables, matrices and series. To remain consistent, we also extend word operations to integers:

Definition 3 (Integer Mirror and Concatenation) *Let the positive integers uniquely number finite words in* $\mathcal{A}^*$ *by def. 2. For* $n, n' \in \mathbf{N}^+$*, we define:*

- *The lexicographic order by* $n \prec n' \Leftrightarrow \mathrm{W}(n) \prec \mathrm{W}(n')$.
- *The mirror by* $\rho(n) = \mathrm{N}(\rho(\mathrm{W}(n)))$.
- *The concatenation by* $n \cdot n' = \mathrm{N}(\mathrm{W}(n) \cdot \mathrm{W}(n'))$.

Both lexicographic $\prec$ and integer order $<$ apply to both integers and words: $\mathbf{w}(n) < \mathbf{w}(n') \Leftrightarrow n < n'$ and $\mathbf{w}(n) \prec \mathbf{w}(n') \Leftrightarrow n \prec n'$. Both orders are total and well-founded.

Mirror is an involution over the integers: $\forall n \in \mathbf{N} : n = \rho\rho(n)$. In the case of the binary $\alpha = 2$ alphabet, mirror is the well-known *odd-even* permutation [20].

Positive integers form a *Monoid* under concatenation. For the binary alphabet $\alpha = 2$, an explicit formula for the concatenation of $n, m > 0$ is:

$$n \cdot m = n + 2^l(m-1) \text{ where } l = |n| = \lfloor \log_2(n) \rfloor.$$

2.2 Languages

A language is a (possibly infinite) set $L = \{u : u \in \mathcal{A}^*\} \subseteq \mathcal{A}^*$ of finite words. We identify the empty language with the null word: $\{\} = \emptyset$.

Mirror $\rho L = \{\rho(v) \in \mathcal{A}^* : v \in L\}$ is an involution $L = \rho\rho(L)$ over $L \subseteq \mathcal{A}^*$.

So is the set-theoretic *complement* $\neg L = \{v \in \mathcal{A}^* : v \notin L\}$: $L = \neg\neg(L)$.

Languages are a *Monoid* under *concatenation*: $L \cdot L' = \{u \cdot u' : u \in L, u' \in L'\}$.

The *star* $L^* \subseteq \mathcal{A}^*$ of a language $L \subseteq \mathcal{A}^+$ (i.e. $\epsilon \notin L$) is the least solution of the equation $L^* = \epsilon \cup (L \cdot L^*)$, namely $L^* = \sum L^{\mathbf{N}}$ where $L^0 = \epsilon$ and $L^{\mathbf{N}+1} = L \cdot L^{\mathbf{N}}$ for $\mathbf{N} \in \mathbf{N}$.

It is well-known [12] that languages form a *Boolean Algebra* under set operations: complement $\neg$, intersection $\cap$ and union $\cup$. The Boolean Algebra is isomorphic to that of the subsets of the integers. This follows by identifying languages with integer sets through def. 2.

Languages form a *Boolean Ring* [12] under intersection $\cap$ and exclusive sum

$$L \oplus L' = \{u \in \mathbf{B}^* : u \in L \cup L',\, u \notin L \cap L'\} = (L \cap \neg L') \cup (\neg L \cap L').$$

A Boolean ring is one in which the sum is self-inverse $\emptyset = L \oplus L$ (minus equals plus) and the product is idempotent $L = L \cap L$ (square is the identity).

Definition 4 (Suffix, Prefix) *Let $L \subseteq \mathcal{A}^*$ be a language over alphabet $\mathcal{A}$.*

- *The suffix $u \in \mathcal{A}^*$ of L is the language: $u^- \cdot L = \{v \in \mathcal{A}^* : u \cdot v \in L\}$.*
- *Let $\mathrm{suff}(L) = \bigcup_{u \in \mathcal{A}^*}(u^- \cdot L)$ denote the set of all the suffixes of L.*
- *The prefix $u \in \mathcal{A}^*$ of L is the mirror $L \cdot u^- = \rho(u^- \cdot \rho(L))$, that is $L \cdot u^- = \{v \in \mathcal{A}^* : v \cdot u \in L\}$. We let $\mathrm{pref}(L) = \rho(\mathrm{suff}(\rho(L)))$.*

2.3 Truth Table and Matrix

We first represent a language by its infinite binary characteristic table.

Definition 5 (Truth Table) *The truth-table $\mathrm{truth}(L) \in \mathbf{F}_2[\infty]$ of a language $L \subseteq \mathcal{A}^*$ is the infinite binary vector $\mathrm{truth}(L) = [L_1 \cdots L_n \cdots]$ defined, for $n > 0$ by $L_n = 1 \Leftrightarrow \mathbf{w}(n) \in L$.*

$$\mathrm{mat}(\mathbf{X}) = \begin{bmatrix} 1110111101\cdots \\ 1011101100\cdots \\ 1110111101\cdots \\ 0101010001\cdots \\ 1110111101\cdots \\ 1011101100\cdots \\ 1110111101\cdots \\ 1110111101\cdots \\ 0000000000\cdots \\ \cdots \end{bmatrix}. \tag{1}$$

Figure 3: The truth-matrix of language $\mathbf{X}$.

X 3 *The truth-table of our example language is*

$$\mathrm{truth}(\mathbf{X}) = [11101111010110111110001\cdots].$$

The truth table is a *Strong Normal Form* for $L, L' \subseteq \mathcal{A}^*$:

$$L = L' \Leftrightarrow \mathrm{truth}(L) = \mathrm{truth}(L').$$

Infinite binary vectors form a Boolean Ring $\langle \oplus, \cap \rangle$ under the bit-wise operations, defined for $a, b \in \mathbf{F}_2$ by $a \cap b = ab$ and $a \oplus b = a + b - 2ab = a + b \pmod 2$. They form a Boolean Algebra $\langle \neg, \cup, \cap \rangle$ if we let $\neg a = 1 - a$ and $a \cup b = a + b - ab$.

Proposition 1 *The truth-table is an isomorphism between the Boolean Ring and Algebra over languages $L, L' \subseteq \mathcal{A}^*$, and those over infinite binary vectors:*

$$\begin{aligned} \mathrm{truth}(\neg L) &= \neg \mathrm{truth}(L), & \mathrm{truth}(L \oplus L') &= \mathrm{truth}(L) \oplus \mathrm{truth}(L), \\ \mathrm{truth}(L \cup L') &= \mathrm{truth}(L) \cup \mathrm{truth}(L), & \mathrm{truth}(L \cap L') &= \mathrm{truth}(L) \cap \mathrm{truth}(L). \end{aligned}$$

f We now represent a language by the truth-tables of its suffixes/prefixes.

Definition 6 (Truth Matrix) *The truth-table* $\mathrm{mat}(L) \in \mathbf{F}_2[\infty, \infty]$ *of* $L \subseteq \mathcal{A}^*$ *is the infinite binary matrix* $\mathrm{mat}(L) = [L^r_c \in \mathbf{F}_2 : r, c > 0]$ *defined by*

$$L^r_c = 1 \Leftrightarrow \mathtt{w}(\rho(r) \cdot c) \in L.$$

Note: For a binary alphabet, the third row $L^3_{1\ldots} = \mathrm{truth}(1^- \cdot L)$ of the truth-matrix $\mathrm{mat}\, L$ is identical to the *automatic sequence* associated to $L \subseteq \mathcal{A}^*$ by [1, 6]. Although L^3 is *not* a normal form for L, [6] shows that the *automatic sequence* of a regular language is algebraic, i.e. root of some polynomial equation. It is further shown in [33, 34] that table $\tau L = L^1_{1\ldots}$ is algebraic if and only if L is regular, and that the characteristic $P = \mathbf{pol}\ (L)$ is a finite ***SNF***: P is the minimal polynomial which has table $Y = \mathrm{truth}L$ for root $0 = P(Y) \pmod 2$.

The truth matrix is again a *Strong Normal Form* for $L, L' \subseteq \mathcal{A}^*$:

$$L = L' \Leftrightarrow \mathrm{mat}(L) = \mathrm{mat}(L').$$

Indeed, row one of $\mathrm{mat}(L)$ is the truth table $[L_1^1 \cdots L_{\mathbf{N}}^1 \cdots] = \mathrm{truth}(L)$ of L. **Note:** : The formula of the truth matrix contains a mirror, whereas Hankel's matrix which is defined by $H[i,j] = 1 \iff w_i.w_j \in \mathcal{L}$ by Fliess [10] doesn't. The following result shows the significance of the mirror in the truth table.

Proposition 2 *Let* $[L_c^r : r, c > 0] = \mathrm{mat}(L) \in \mathbf{F}_2[\infty, \infty]$ *be the truth matrix of language* $L \subseteq \mathcal{A}^*$. *For all* $r, c > 0$*:*

1. *Row* r *of* $\mathrm{mat}(L)$ *is the truth table* $\mathrm{truth}(\rho(r)^- \cdot L)$ *of the suffix language* $u^- \cdot L$, *for word* $u = \rho_{\mathbf{w}}(r)$.
2. *Column* c *is the the truth table of the suffix* $u^- \cdot \rho\, L$, *for* $u = \mathbf{w}(c)$.
3. *Matrix bit* L_c^r *derives from the first row (or column) by:*

$$L_c^r = L_{\rho r \cdot c}^1 = L_1^{r \cdot \rho c}.$$

4. *The truth matrix of the mirror language is the transposed of the truth matrix:* $\mathrm{mat}(\rho(L)) = \mathrm{mat}(L)^{tr}$.

Proof: Points 1 and 3 follow by definition: $\mathrm{mat}(L)_{r,c} = 1 \iff \rho r.c \in L$ for $r, c \in \mathbf{N}$. Points 1 and 3 follow by mirror: $\rho(\rho r.c) = \rho c \cdot r \in \rho L$. Q.E.D.

2.4 Regular Languages

Automata Theory [14, 24, 26] characterizes *Regular Languages* $L \in \mathrm{RL}$ by equivalent finite representations as regular expressions or automata (deterministic or not). The following classic result gives us an intrinsic characterization:

Theorem 1 (Nerode [25], Moore [24]) *Language* $L \subseteq \mathcal{A}^*$ *is* regular $L \in \mathrm{RL}$ *if and only if it has finitely many* $s = |\mathrm{suff}(L)| \in \mathbf{N}$ *suffix languages* $\mathrm{suff}\, L = \{u^-.L : u \in \mathcal{A}^*\}$. *Integer* s *is the minimal number of states among all (complete) deterministic automata which accept* L.

It follows from prop. 2 that $s = |\mathrm{suff}(L)|$ is equal to the number of different rows in the truth-matrix $\mathrm{mat}(L)$: $s = |\{L^r : r > 0\}|$. Indeed, suffix $u \in \mathcal{A}^*$ of L is uniquely identified by row $n = \mathrm{N}(\rho\, u) > 0$ of the truth-matrix:

$$L^n = \mathrm{truth}(u^-.L) \text{ where } u^-.L = \{v : u \cdot v \in L\}.$$

Definition 7 (Regular Index) *Let* $L \in \mathrm{RL}$ *be a regular language, with* $s = |\mathrm{suff}(L)|$ *different suffix languages, corresponding to* $s = |\{L^r : r > 0\}|$ *different rows in the truth-matrix* $\mathrm{mat}(L)$.

- *The* minimal access path *to row* L^n *is* $\mathrm{map}(n) = min\left\{k : L^k = L^n\right\}$.

- *The* regular index *of* L *is the sequence* $\text{index}(L) = [n_1 \cdots n_s]$ *obtained by sorting all minimal pathes* $\{\text{map}(n) : n > 0\} = \{n_1 \cdots n_s\}$ *in increasing order:* $n_1 < n_2 < \cdots < n_s$*; clearly* $1 = n_1$*, since* $L = L^1$*.*

Note: Since $s = |\text{index}(L)|$, the *regular index* is a refined version of the classical definition in [25] who uses $s = |\text{suff}(L)|$. Through the mirror, index words $u_k = \rho\, \mathbf{w}(n_k)$ uniquely appear in lexicographic order: $u_1 = \epsilon \prec u_2 \prec \cdots \prec u_s$.

X 4 *The regular index of our example language is* $\text{index}(\mathbf{X}) = [1, 2, 4, 9]$*, which correspond to the suffixes:* $X^1 = \mathbf{X}$*,* $X^2 = \mathbf{X} \oplus 0\mathbf{X}$*,* $X^4 = 0\mathbf{X}$ *and* $X^9 = \emptyset$*.*

2.5 Minimal Deterministic Automaton MDA

Our reader will accept that the following definition of the MDA precedes the formal introduction of deterministic automata in sect. 4.2.

Definition 8 (Minimal Deterministic Automaton) *Let* $L \in \text{RL}$ *be a regular language, with regular* $\text{index}(L) = [n_1 \cdots n_s]$*, and minimal access path* $\text{map}(n) \in \text{index}(L)$ *for* $n > 0$*. The* Minimal Deterministic Automaton $A = (s, I, T(), F) = \text{MDA}(L)$ *for* L *is:*

1. *The states* $S = \{n_1, \cdots, n_s\}$ *are the elements* $index(L) = [n_1 \cdots n_s]$*.*
2. *The initial state is* $I = 1 = n_1$*.*
3. *The final (accepting) states are* $F = \{n_k : \epsilon \in L^{n_k}\} \subseteq S$*.*
4. *Transition* $q' = T_l(q)$ *is defined by* $q' = \text{map}(l + b \times q) \in S$*, for each letter* $l < \alpha$ *and state* $q = \text{map}(q) \in S$*.*

Note: The MDA is classically [24] defined up to state isomorphism. Definition 8 makes it unique, and a *Strong Normal Form* for regular languages $L, L' \in \text{RL}$:

$$L = L' \Leftrightarrow \text{MDA}(L) = \text{MDA}(L').$$

The dual normal form for $L \in \text{RL}$ is the mirror $\text{MDA}(\rho\, L)$, with $r = |\text{index}(\rho\, L)|$ states. Yet, r is exponential in s for $U_n = \mathbf{B}^n \cdot 1 \cdot \mathbf{B}^*$, and conversely for ρU_n.

X 5 *The* MDA *of* $\mathbf{X}$ *is represented by the graph in fig. 4.*

3 Regular Dimension

A set $S = \{L(1) \cdots L(n) \cdots\}$ of languages $L(n) \subseteq \mathcal{A}^*$ generates a vector space $V = \mathbf{F}_2\langle S\rangle = \left\{\bigoplus_{i \in S} L(i) : S \subseteq \mathbf{N}^+\right\}$, under the exclusive set sum $\oplus$ and the (trivial) scalar product: $0 \times V = \emptyset$ and $1 \times V = V$.

By prop. 1, the vector space $V = \mathbf{F}_2\langle S\rangle$ generated by languages in S is isomorphic to the vector space $V' = \mathbf{F}_2\langle \text{truth}(L) :\ L \in S\rangle$ generated by their truth-tables $\text{truth}(L) \in \mathbf{F}_2[\infty]$.

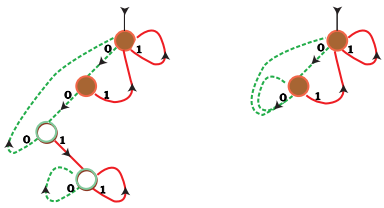


Figure 4: The labeled graph representations of MDA($\mathbf{X}$) and MXA($\mathbf{X}$).

The dimension $d = \dim(V) = |\text{base}(V)|$ of a finite vector space $V = \mathbf{F}_2\langle S\rangle$ is the minimal size of any base for V. By th. 1, the truth-matrix of L has finitely many different rows if and only if $L \in \text{RL}$ is regular. It follows that the infinite truth-matrix $\text{mat}(L)$ has a finite rank over $\mathbf{F}_2$ if and only if $L \in \text{RL}$.

Definition 9 (Regular Dimension) *The dimension* $d = \dim(L) \in \mathbf{N}$ *of a regular language* $L \in \text{RL}$ *is the rank of the truth-matrix:*

$$d = \dim(L) = \text{rank}_{\mathbf{F}_2}(\text{mat}(L)).$$

Proposition 3 *The dimension* $d = \dim(L)$ *of a regular language* $L \in \text{RL}$ *is equivalently characterized by:*

1. $d = \text{rank}_{\mathbf{F}_2}(\text{mat}(L))$ *the rank of the truth matrix over* $\mathbf{F}_2$.
2. $d = \dim(\rho L)$ *is the dimension of the mirror language.*
3. $d = \dim \mathbf{F}_2\langle \text{suff}(L)\rangle$ *is the dimension of the suffix vector space.*
4. $d = \dim \mathbf{F}_2\langle \text{pref}(L)\rangle$ *is the dimension of the prefix vector space.*

Proof: Def. (1) is definition 9. By prop. 2, the truth-matrix of the mirror language is the transposed matrix of $\text{mat}(L)$. Standard linear algebra [12] shows that the rank of a matrix is invariant by transposition, hence (2). Likewise, the rank of a matrix is equal to the dimension of the vector space generated by its rows or columns, hence (3) and (4). Q.E.D.

3.1 Regular Base

The following algorithm computes the dimension of a regular language by extracting a minimal base from the suffixes.

Algorithm 1 (Regular Base) *Let* $L \in \text{RL}$ *be a regular language, with regular* $\text{index}(L) = [n_1 \cdots n_s]$ *and* $\text{map}(n) \in \text{index}(L)$ *for* $n > 0$. *Let* $V = \mathbf{F}_2\langle L^{n_1} \cdots L^{n_s}\rangle$ *be the vector space generated by the* s *different suffix languages. For* $k \in [1 \cdots s]$, *we compute:*

1. *The dimension* $d_k = \dim V_k \leq k$ *of the vector space* $V_k = \mathbf{F}_2\langle L^{n_1} \cdots L^{n_k}\rangle$.
2. *The minimal* $\text{base}(k) = [B(1) \cdots B(d_k)]$ *such that* $\mathbf{F}_2\langle \text{base}(k)\rangle = \mathbf{F}_2\langle V_k\rangle$.

3. *The unique decomposition* $ud(k) \subseteq \{1 \cdots d_k\}$ *s.t.* $L^{n_k} = \bigoplus_{i \in ud(k)} B(i)$.

The process is initialized by: ($d_0 = 0$, $\text{base}_0 = \emptyset$, $ud(0) = 0$). *We compute* (d_k, $B(k)$, $\text{ubd}(k)$) *at stage* $k \in [1 \cdots n]$ *by considering two cases:*

Case $L^{n_k} \notin V_k = \mathbf{F}_2\langle \text{base}(k-1) \rangle$ *We let:* $d_k = d_{k-1} + 1$*;* $base_k = \text{base}_{k-1} \cup B(k)$, $B(k) = L^{n_k}$*;* $ud(k) = \{k\}$.

Case $L^{n_k} \in V_k = \mathbf{F}_2\langle \text{base}(k-1) \rangle$ *We let:* $d_k = d_{k-1}$, $\text{base}_k = \text{base}_{k-1}$ *and* $ud(k) = S \subseteq \text{base}_{k-1}(L)$ *is the unique decomposition* $L^{n_k} = \bigoplus_{i \in S} B(i)$.

At stage s*, we find: the dimension* $d = d_n = \dim(V)$*, the* Regular Base $\text{base}(V) = [B(1) \cdots B(d)] = [L^{b_1} \cdots L^{b_d}]$ *and the unique base decomposition* $\text{ubd}(n) = bd(\text{map}(n)) \subseteq \{1 \cdots d\}$ *such that* $L^n = \bigoplus_{i \in \text{ubd}(n)} L^{b_i}$ *for all* $n > 0$.

X 6 *The dimension of language* $\mathbf{X}$ *is* $2 = \dim(\mathbf{X})$*, and its regular base is* $\text{base}(\mathbf{X}) = [\mathbf{X}, X^2]$ *since (see* **X***4)* $X^2 = \mathbf{X} \oplus 0 \cdot \mathbf{X}$ *and* $X^4 = \mathbf{X} \oplus X^2$.

3.2 Minimal Xor Automata

We anticipate again on the formal definition of automata in sec. 4, and use alg. 1 to construct the minimal non-deterministic Xor automaton $\text{MXA}(L)$ of the regular language $L \in \text{RL}$. We then relate $\text{MXA}(L)$ to $\text{MDA}(L)$.

Definition 10 (Minimal Xor Automaton) *Let* $L \in \text{RL}$ *be a regular language, with dimension* $d = \dim(L)$*, regular base* $\text{base}(L) = [L^{b_1} \cdots L^{b_d}]$ *and unique base decomposition* $\text{ubd}(n) \subseteq \{b_1 \cdots b_d\}$ *for* $n > 0$.

The Minimal Xor Automaton $A = \text{MXA}(L) = (d, I, T(), F) \in \text{NXA}$ *is:*

1. *The states* $S \subseteq B$ *are subsets of* $\mathcal{B} = \{b_1, \cdots, b_d\}$.
2. *The initial state is* $I = \{1\} = \{b_1\}$.
3. *Transition* $q' = T_l(q)$ *is defined by* $q' = \bigcup_{i \in q} \text{ubd}(l + b \times i) \subseteq \mathcal{B}$*, for each letter* $l < \alpha$ *and state* $q \subseteq \mathcal{B}$.
4. *The final (accepting) states are* $F = \left\{ b_k : \epsilon \in L^{b_k} \right\} \subseteq \mathcal{B}$.

X 7 *The graph of* $\text{MDA}(X)$ *appears in fig. 4.*

3.3 From MDA to MXA

For the clarity of this section, we simply consider the binary alphabet, where both the NDA and the MXA can be represented by complete binary trees (as opposed to α-ary trees, for $\alpha > 2$).

A characteristic property [9] of the regular $\text{index}(L) = [n_1 \cdots n_s]$ of $L \in \text{RL}$ is to be closed by suffix. In the binary case, this is simply stated by: $\forall k > 1 : n_k \div 2 \in [n_1 \cdots n_{k-1}]$.

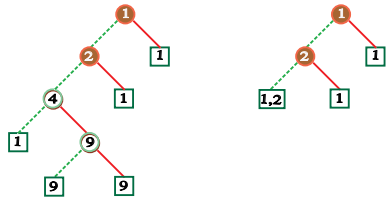


Figure 5: The tree representations of MDA(X) and MXA(X).

The tree nodes of MDA(L) (def. 8) are labeled by indexes. The root is $n_1 = 1$. For $k > 1$, node n_k is the left (resp. right) son of $n_k \div 2$ if n_k is even (resp. odd). Each of the $s + 1$ leaves contains a back-pointer to a tree node.

The index $\mathcal{B} = \{b_1 \cdots b_d\}$ of the regular base of L is closed by suffix, and it is a subset of the index numbers.

The tree nodes of MXA(L) (def. 10) are labeled by base indexes $\{b_1 \cdots b_d\}$. The root is $b_1 = 1$. For $k > 1$, node b_k is the left (resp. right) son of $b_k \div 2$ if b_k is even (resp. odd). Each of the $d + 1$ leaves contains a (non-deterministic) set of back-pointers to tree nodes.

The tree for MXA(L) is a *prefix* of the tree for MDA(L). When $s > d$, we reduce the MDA by replacing finding the least $n_k \notin \mathcal{B}$, and by replacing the subtree rooted at n_k by the (non-deterministic) leaf node $\mathrm{ubd}(n_k)$. We repeat the process until there are exactly d base nodes left: the result is MXA(L).

X 8 *The tree representations of* MDA(**X**) *and* MXA(**X**) *appear in fig. 5.*

4 Non Deterministic Xor Automata

Non deterministic finite automata NFAs are equivalently represented by their labeled graphs, or by their adjacency matrices [9].

We introduce *Nondeterministic Xor Automata* NXAs here by their matrix representation. The correspondence with the graph representation should be clear, from NFA experience, and from the given examples.

Definition 11 *A Nondeterministic Xor Automaton* $A = (n, I, T(), F) \in \mathrm{NXA}$ *is given by:*

- $n \in \mathbf{N}$ *is the number of states.*
- $I \in \mathbf{F}_2[1, n]$ *is the row of initial states.*
- $T_l \in \mathbf{F}_2[n, n]$ *is the transition matrix for each letter* $l \in B$.
- $F \in \mathbf{N}[n, 1]$ *is the column of final states.*

Transition $T() \in \mathbf{N} \mapsto \mathbf{F}_2[n, n]$ *is extended to integers by:*

$$\begin{array}{rcl} T(1) & = & \mathbf{Id}(n) \text{ is the } n \times n \text{ identity matrix;} \\ T(l+1) & = & T_l \text{ for each letter } l \in B; \\ T(n \cdot m) & = & T(n) \circ T(m) \text{ is the matrix product, for } n, m > 0. \end{array}$$

For $\mathbf{w}(n) = w_1 \cdots w_k \in \mathcal{A}^k$, *we find:* $T(n) = T_{w_1} \circ \cdots \circ T_{w_k} \in \mathbf{F}_2[n, n]$. *The language accepted by* A *is the set of words* u *with unit trace in* A*:*

$$\Lambda_{\oplus}(A) = \{u \in \mathcal{A}^* : \, 1 = trace(I \circ T(\mathrm{N}\, u) \circ F)\}\,.$$

X 9 *The minimal nondeterministic Xor automaton for* $\mathbf{X}$ *(fig. 4) is presented by the matrices:* $\mathrm{MXA}(\mathbf{X}) = (2, [10], \begin{bmatrix} 01 \\ 11 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix})$

Definition 11 gives a method for computing bits of truth-table of the language accepted by the automaton. The bits of the truth-matrix are computed by:

Proposition 4 *Let* $A = (n, I, T(), F) \in \mathrm{NXA}$ *be a nondeterministic Xor automaton for* $L = \Lambda_{\oplus}(A) \in \mathrm{RL}$. *The truth-matrix* $\mathrm{mat}(L) = [L^r_c : r, c > 0]$ *of the accepted language* L *is equal to:*

$$\forall r, c > 0 : \; L^r_c = trace(I \circ T(\rho\, r) \circ T(c) \circ F). \tag{2}$$

Proof: By definitions 10 and 5 we know that: $\mathbf{w}(c) \in L \Leftrightarrow L^1_c = 1$, for all $c > 0$; it follows that $L^1_c = trace(I \circ T(1) \circ T(c) \circ F)$, where $T(1) = \mathbf{Id}(n)$ is the identity. So, $[trace(A(c)) : c > 0]$ is the first row of the truth-matrix, i.e. the truth table of $L = L^1$. Proposition 2 derives the other rows by $L^r_c = L^1_{\rho\, r \cdot c}$, and it simply follows that $L^r_c = trace(I \circ T(\rho\, r \cdot c) \circ F) = trace(I \circ T(\rho\, r) \circ T(c) \circ F)$. Q.E.D.

4.1 Multiple Acceptance

Definition 11 uses the $\langle \oplus, \cap \rangle$ scalar product over the Boolean Ring $\mathbf{F}_2$:

$$R \circ_{\oplus} C = \bigoplus_{1 \le k \le n} r_k c_k = (S \bmod 2) \text{ where } S = \sum_{1 \le k \le n} r_k c_k. \tag{3}$$

Regarding $\mathbf{F}_2$ as a Boolean Algebra under $\langle \cup, \cap \rangle$ changes the scalar product to:

$$R \circ_{\cup} C = \bigcup_{1 \le k \le n} r_k c_k = (S > 0) \text{ where } S = \sum_{1 \le k \le n} r_k c_k. \tag{4}$$

Both are derived from the usual scalar product $\langle +, \times \rangle$ over the integers:

$$R \circ_{+} C = \sum_{1 \le k \le n} r_k c_k. \tag{5}$$

Let us read again def. 11, and consider in turn the implications of each of the three choices for the scalar product:

- $\circ_+$ With ordinary matrix product $\langle +, \times \rangle$ over integers, the trace $\mu(n, A) = trace(I \circ_+ T(\mathrm{N}\, u) \circ_+ F)$ of u in A is known [27, 9] as the *multiplicity*, since integer $\mu(n, A)$ counts the number of pathes along word u from all initial to all final states in the graph of A.

$\circ_\cup$ With matrix product $\langle \cup, \cap \rangle$, the trace $to = trace(I \circ_\cup T(\aleph\, u) \circ_\cup F)$ of u in A is 1 if and only if there exists a path along u from some initial state to some final state, i.e. $to = 1 \Leftrightarrow \mu(n, A) > 0$. The accepted language $\Lambda_\cup(A)$ is the classically defined ([9] ch. VI) one for $A \in$ NFA.

$\circ_\oplus$ With product $\langle \oplus, \cap \rangle$, the trace $tx = trace(I \circ_\oplus T(\aleph\, u) \circ_\oplus F)$ of u in A is 1 if and only if the number of pathes along u from initial states to final states is odd, i.e. $tx = (\mu(n, A)\, mod\, 2)$. By def. 11, the language $\Lambda_\oplus(A)$ is accepted by the *Nondeterministic Xor Automaton* $A \in$ NXA.

The two dual ***or*** / ***xor*** languages accepted by automaton A are:

1. $\Lambda_\cup(A) = \{u \in \mathcal{A}^* :\ \mu(u, A) > 0\}$, as defined for $A \in$ NFA.
2. $\Lambda_\oplus(A) = \{u \in \mathcal{A}^* :\ 1 = (\mu(u, A)\, mod\, 2)\}$, as defined for $A \in$ NXA.

Both languages are such that $\Lambda_\oplus(A) \subseteq \Lambda_\cup(A)$.

Both are different if and only if the multiplicity of some word is not zero but even: $\Lambda_\oplus(A) \neq \Lambda_\cup(A) \Leftrightarrow \exists u \in \mathcal{A}^* : \mu(u, A) = 2k$ for $k > 0$.

Both languages are equal $\Lambda_\oplus(A) = \Lambda_\cup(A)$ when automaton $A \in$ DFA is *deterministic* (sect. 4.2). Indeed, it follows in this case [9] that the multiplicity of each word is either 0 or 1: $\forall u \in \mathcal{A}^* :\ \mu(u, A) \in \mathbf{B}$, hence $\Lambda_\oplus(A) = \Lambda_\cup(A)$. **Note:** Linear automata are introduced by Schützenberger [28] over $\mathbf{Z}$. They are extended to rings K by Fliess [11]. Definitions 9 and 11 of the dimension and ***xor*** acceptance are special cases of [11], once we specialize $K = \mathbf{F}_2$ to the Boolean Ring. Likewise, the xor automata are a special case for $\mathbf{F}_2$ of the *weighted automata* in [3, 23].

X 10 *The n-th power of the transition matrix* $X_0 = \begin{bmatrix} 01 \\ 11 \end{bmatrix}$ *in* MXA($\mathbf{X}$) *(***X***9) is expressed* $(X_0)^n = \begin{bmatrix} F_{n-1}\, F_n \\ F_n\, F_{n+1} \end{bmatrix}$ *by Fibonacci [19] numbers:* $F_0 = 0$, $F_1 = 0$ *and* $F_{n+2} = F_{n+1} + F_n$. *The multiplicity of word* 0^n *in* MXA($\mathbf{X}$) *is equal to* F_n, *which is an even number if and only if* n *is a multiple of 3. Using this fact, we can characterize the words of* $\mathbf{X} \subset \mathbf{B}^*$ *by:*

$$u \in \mathbf{X} \Leftrightarrow u = 1^* 0^{a_1} 1^+ 0^{a_2} \cdots 1^+ 0^{a_k} 1^*,$$

for all $k \in \mathbf{N}$, *with* $1^+ = 1 \cdot 1^*$ *and* $2 \neq (a_i\ mod\ 3)$ *for* $i \in [1 \cdots k]$.

Note: The question of multiple acceptance applies to regular expressions as well as to NFAs. For example, word 0 has multiplicity 2 in the regular expression $0 + 0$. In general, the multiplicity $\mu(u, E) \in \mathbf{N}$ of word u in a regular expression E is the number of ways to parse u according to E. The OR-accepted language is $\Lambda_\cup(E) = \{u \in \mathcal{A}^* :\ \mu(u, E) > 0\}$. The XOR-accepted language is $\Lambda_\oplus(E) = \{u \in \mathcal{A}^* :\ \mu(u, E) = 1 \pmod 2\}$. They are different if and only if some word has an even multiplicity in E. Efficient translations are known between regular expressions and NFA, such as [2] which also preserves multiplicity. It follows that a theory of XOR-accepting regular expressions can be developed, alongside that of NXAs.

X 11 *The regular expression* $RX = (0 + 1 + 00)*$ *is such that* $\Lambda_{\oplus}(RX) = \mathbf{X}$, *while* $\Lambda_{\oplus}(RX) = \mathbf{B}^*$.

4.2 Deterministic Automata

The matrices for the MDA$(L) = (s, I, T(), F)$ in def. 8 are: $I = [10^{s-1}]$ and $T_l[r, c] = 1 \Leftrightarrow c = \text{map}(r)$, for all $r, c \in [1 \cdots s]$ and $l < \alpha$. Note that, by construction, each row contains exactly one bit at 1.

Definition 12 (Stochastic Binary Matrices) *A matrix* $M \in \mathbf{F}_2[n, m]$ *is* row stochastic *if each row contains a single one:*

$$\forall c \in [1 \cdots m] : \; 1 = \sum_{1 \le r \le n} M_c^r.$$

Definition 13 (DFA) *Automaton* $A = (n, I, T(), F) \in$ NXA *is a* Deterministic (complete) Finite Automaton $A \in$ DFA *if its matrices* I *and* T_l *(for* $l < \alpha$*) are row stochastic (exactly one bit at 1).*

X 12 *The minimal deterministic automaton for* $\mathbf{X}$ *is presented by the row stochastic matrices:* $\text{MDA}(\mathbf{X}) = (4, [1000], \begin{bmatrix} 0100 \\ 0010 \\ 1000 \\ 0001 \end{bmatrix}, \begin{bmatrix} 1000 \\ 1000 \\ 0001 \\ 0001 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix})$.

The following is a classical [9] characterization of *minimal* DFAs:

Proposition 5 *A deterministic automaton* $A = (n, I, T(), F) \in$ DFA *accepting* $L = \Lambda(A)$ *is* minimal *if and only if:*

1. *The number of states is equal* $n = |suffix(L)|$ *to the number of suffixes.*
2. *All states are accessible, and all recognize different languages:* $q = q' \Leftrightarrow \Lambda(A[I = q]) = \Lambda(A[I = q'])$.
3. *Automaton* A *is isomorphic to* MDA(A).

4.3 Atomic NXA Operations

All NXA algorithms in sect. 5 rely on three atomic operations:

Definition 14 (Atomic NXA Operations) *Let* $A = (n, I, T(), F)$ *be a xor automaton* $A \in$ NXA.

Mirror *The* mirror *(a.k.a. transposed)* $A' = \rho\, A = A^{tr} \in$ NXA *of* A *is the xor automaton* $A' = (n, F^{tr}, T^{tr}(), I^{tr})$ *where matrix* $T_l' = {T_l}^{tr}$ *is the transposed of matrix* T_l*, for each letter* $l \in B$.

Similitude *Let* $P, P^- \in \mathbf{F}_2[n, n]$ *be an inverse* $P \circ P^- = Idn$ *matrix pair. Automaton* $A' = P \circ A \circ P^- \in$ NXA *is defined by* $A' = (n, I \circ P, T'(), P^- \circ F)$ *where* $T_l' = P^- \circ T_l \circ P$*, for each letter* $l \in B$.

Shrink *Automaton* $A' = \text{shrink}(A, d) = (d, I', T'(), F')$ *results by removing rows & columns* k*, for* $d < k \leq n$*, from all matrices in* A*.*

Note that the transposed automaton $A' = A^{tr}$ recognizes the mirror language $\Lambda_\oplus(A') = \rho\,\Lambda_\oplus(A)$ within the same number of states (and transitions).

Similar automata A and $A' = P \circ A \circ P^-$ recognize the same language $\Lambda_\oplus(A') = \Lambda_\oplus(A')$ within the same number of states.

Automaton $A' = \text{shrink}(A, d)$ recognizes the same language as A, with only d states, if none of the k-th state $D_k = [0^{k-1}10^{n-k}]$ is accessible, i.e. $D_k \notin \{I \circ T(n) : n > 0\}$ for $d < k \leq n$.

5 Minimizing Nondeterministic Xor Automata

Except for the reference to MXA (def. 10), the following characterization of minimal Xor automata is a special case (for the ring $K = \mathbf{F}_2$) of a theorem by Fliess [10] on minimal linear automata.

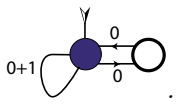
Proposition 6

- *The regular dimension* $\dim(L)$ *is lower than the number of states in all* $A \in \text{NXA}$ *for* $L = \Lambda_\oplus(A)$*.*
- *Automaton* $A = (n, I, T(), F) \in \text{NXA}$ *for* $L = \Lambda_\oplus(A)$ *is* minimal *if and only if:*
  1. *The number* n *of states is equal* $n = \dim(L)$ *to the dimension of* L*.*
  2. *All states are accessible and linearly independent:*

$$\forall q : \Lambda_\oplus(A[I = q]) \notin \mathbf{F}_2\langle\Lambda_\oplus(A[I = q'] : q' \neq q)\rangle.$$

  3. *The mirror* ρA *of* A *is minimal.*
  4. *Automaton* A *is* similar *to* $\text{MXA}(A)$*.*

X 13 *The* NXA $S = P \circ C \circ P^-$ *is similar to* $C = \text{MXA}(\mathbf{X})$ *(see* **X***9) through* $P = \begin{bmatrix} 10 \\ 11 \end{bmatrix} = P^-$ (mod 2)*. It is equivalently represented by its matrices* $S = (2, [10], \begin{bmatrix} 11 \\ 10 \end{bmatrix}, \begin{bmatrix} 10 \\ 00 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix})$*, and by its graph:*



By exhaustive enumeration, we verify that S *has both minimal number of states and transitions among all* NXA*s for* $\mathbf{X}$*.*

Abstract algorithms are known ([10, 3]) for minimizing linear automata over abstract rings K, in polynomial time. We introduce here an efficient (cubic time, quadratic storage) minimization algorithm over the concrete 2-elements field $K = \mathbf{F}_2$. It finds the MXA in two mirror stages, like Brzozowski's [5] (exponential time, see [35]) algorithm for computing the MDA.

5.1 Linearly Reduced Automata

Linear reducibility is to MXAs what accessibility is to DFAs:

Definition 15 (Linearly Reduced) *Let $A = (n, I, T(), F)$ be a* NXA*, and let $\mathcal{A} = \{I \circ T(\rho(m)) : m > 0\}$ represent its accessible configurations.*

- *A is* linearly reduced *if the number of states is equal to the dimension of the vector space generated by the accessible configurations: $n = \dim \mathbf{F}_2\langle\mathcal{A}\rangle$.*
- *A is* linearly co-reduced *if the mirror ρA is linearly reduced.*

Since $n = \dim \mathbf{F}_2\langle\mathcal{A}\rangle$, we can extract a minimal base of size n for $\mathcal{A}$.

Definition 16 (Base Matrices) *Let $A = (n, I, T(), F)$ be* linearly reduced*.*

- *The* Row Access Base $\mathcal{R} = \text{rab}(A) \in \mathbf{F}_2[n, n]$ *of $A \in$ NXA is the matrix whose row $k \in [1 \cdots n]$ is defined by $\mathcal{R}^k_{1\cdots n} = I \circ T(\rho\, r_k)$, for the base index*

$$r_k = \min\left\{m :\ I \circ T(\rho\, m) \notin \mathbf{F}_2\langle\mathcal{R}^1 \cdots \mathcal{R}^{k-1}\rangle\right\}.$$

- *Automaton A is in* diagonal form *if* $\mathbf{Id}(n) = \text{rab}(A)$.

By construction, the $n \times n$ row access matrix $\mathcal{R} = \text{rab}(A)$ of a linearly reduced automaton $A \in$ NXA has full rank: $n = \text{rank}_{\mathbf{F}_2}\mathcal{R}$. It follows from linear algebra [12] that matrix $\mathcal{R}$ has a unique inverse $\mathcal{R}^- \in \mathbf{F}_2[n, n]$ such that $\mathbf{Id}(n) = \mathcal{R} \circ \mathcal{R}^-$. Among all automata which are similar to A, automaton $\mathcal{R} \circ A \circ \mathcal{R}^-$ is the only one which is in diagonal form.

Note the minimal automata $Lx = \text{MXA}(L)$ (def. 10) and $Ld = \text{MDA}(L)$ (def. 8) of a regular language $L \in$ RL are *both* linearly reduced, since both matrix representations are in diagonal form: $\text{rab}(Lx) = \mathbf{Id}(d)$ for $d = \dim L$, and $\text{rab}(Ld) = \mathbf{Id}(s)$ for $s = |\text{suff}(L)|$. It follows that:

Proposition 7 (Minimal Diagonal Forms) *Let $L \in$ RL be a regular language with $s = |\text{suff}(L)|$ suffixes and dimension $d = \dim L$.*

- *Among all minimal (s states) deterministic* DFA*s for L, the* MDA(L) *is the unique automaton in diagonal form.*
- *Among all minimal (d states) nondeterministic* NXA*s for L, the* MXA(L) *is the unique automaton in diagonal form.*

5.2 Efficient Minimization

Our key theoretical contribution is the following:

Theorem 2 *A nondeterministic automaton $A = (n, I, T(), F) \in$ NXA for $L = \Lambda_\oplus(A) \in$ RL is* minimal *$n = d = \dim(L)$ if and only if both A and its mirror ρA are linearly reduced.*

Proof: It follows from prop. 6.2 that a minimal $n = d$ automaton A is linearly reduced; so is the mirror ρA by prop. 6.3. Hence, minimality of A implies that it is linearly reduced, in both ways.

Conversely, we already know that $n \geq d$ by prop. 6. We prove that $n \leq d$ by considering the $n \times n$ row base matrix $\mathcal{R} = \mathrm{rab}(A)$ (def. 16), and the dual column base matrix obtained by mirror and transposition: $\mathcal{C} = \mathrm{rab}(\rho\, A)^{tr}$. We let $\mathrm{ubd}(A) = [r_1 \cdots r_n]$ and $\mathrm{ubd}(\rho A) = [c_1 \cdots c_n]$.

Since A and $\rho\, A$ are linearly reduced, both matrices and their product $P = \mathcal{R} \circ \mathcal{C}$ have full rank: $n = \mathrm{rank}_{\mathbf{F}_2}\mathcal{R} = \mathrm{rank}_{\mathbf{F}_2}\mathcal{C} = \mathrm{rank}_{\mathbf{F}_2} P$. By def. 16, the bits of matrix P are extracted from the truth-matrix of L since $P[i,j] = L^{r_i}_{c_j}$. It follows that the rank n of the sub-matrix P is bounded by that $d = \mathrm{rank}_{\mathbf{F}_2}\mathrm{mat}(L)$ of the full truth-matrix, hence finally $n = d$. Q.E.D.

It follows from th. 2 that we can construct a minimal automaton equivalent to $A \in \mathrm{NXA}$ by linearly reducing its rows and columns. Alg. 3 is an efficient reduction to *diagonal form*, and we use it twice to directly construct the MXA of the accepted language:

Algorithm 2 (Minimization by Mirrors) *Let A be a* NXA *which accepts the language $L = \Lambda_{\oplus}(A)$. Let* diag *be a procedure which reduces an arbitrary* NXA *to its diagonal form. We compute the* Minimal Xor Automaton *of L by:* $\mathrm{MXA}(L) = \mathrm{diag}(\rho\, \mathrm{diag}(\rho\, A))$.

Kronecker's delta symbol is defined by: $\delta(0) = 1$, and $\delta(n) = 0$ for $n > 0$. We let $D_1 \cdots D_n$ denote the rows $D_k[r] = \delta(r - k)$ of the identity matrix.

Algorithm 3 (Diagonal Reduction) *Let $A = (n, I, T(), F) \in \mathrm{NXA}$ be a nondeterministic XOR automaton for $L = \Lambda_{\oplus}(A) \in \mathrm{RL}$ we construct an equivalent $L = \Lambda_{\oplus}(A')$ automaton $A' = \mathrm{diag}(A) \in \mathrm{NXA}$ which is* linearly reduced in diagonal form *by:*

```
fun   M = diag(A) : A, M ∈ MXA
 {    var  Q ∈ FIFO;                                   // Queue Q
      Q := push(A.I);                          // Initialize Q to vector I
      d := 0;                                          // Initial dimension
      while  Q ≠ ∅  do                                 // Repeat until empty:
      {  C := pop(Q)                           // Explore next configuration
        if  ∃k > d : 1 = C[k]  then                  // C ∉ F2[D1 ··· Dd]
        {          // Push the configurations accessed from C onto Q:
          for  (l < α)  do  Q := push(A.T[l]);
          (P, P⁻) := pivot(d, k, C);       // Compute the pivot matrices
          A := P ∘ A ∘ P⁻;                            // Diagonalize row d+1
          d := d + 1;                                 // Increase the dimension
        }  else  {}}                    // Do nothing: C ∈ F2[D1 ··· Dd]
      A := shrink(A, d); // Eliminate un-accessible states [d+1 ··· n]:
      return(A).  }
```

Note: In alg. 3, it is important to store the configurations to be explored in *First-In First-Out* FIFO order. Using a different queue order would still yield a linearly reduced automaton, but it would no longer be in canonical form. The chosen order traverses the accessible configurations $I \circ T(\rho n)$ in breadth first manner, i.e. lexicographic order over words $w\mathbf{w}(\rho n)$, i.e. hierarchical order over nodes in the tree representation.

Algorithm 4 (Pivot) *In the notations of alg. 3 the pivot matrices* $(P, P^-) = pivot(d, k, C)$ *are defined, for a row* $C \in \mathbf{F}_2[n]$ *such that* $1 = C[k]$ *with* $d < k \leq n$*, by the pair of inverse matrices* $P = Q \circ G$ *and* $P^- = G \circ Q$ *where:*

- *Matrix* $Q = Q^-$ *permutes the columns* $d+1$ *and* k*.*
- *Let* $R = C \circ Q \in \mathbf{F}_2[n]$ *be the permuted row, such that* $1 = R[d+1]$*.*
- *Matrix* $G = G^-$ *is the identity matrix, except for row* $d+1$ *which is* R*:* $G[r, c] =$ *if* $(r = d+1)$ *then* $R[c]$ *else* $\delta(r - c)$*, for* $1 \leq r, c \leq n$*.*

X 14 *Alg. 3 reduces the four states deterministic* $A = \mathrm{MDA}(\mathbf{X})$ *(***X***12) to the two states nondeterministic* $C = \mathrm{MXA}(\mathbf{X})$ *(***X***9) as follows:*

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

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

Note that $A = \mathrm{diag}(A)$ *since* $A = \mathrm{MDA}(\mathbf{X})$ *is already in diagonal reduced form.*

Proposition 8 *Algorithm 3 reduces* $A \in \mathrm{NXA}$*, with* n *states, to its diagonal form* $B = \mathrm{diag}(A)$*, with* $m \leq n$ *states, all in time* $O(m \times n^2)$ *(bit operations), and space* $2n(n+1)$ *(bits).*

Proof: Each FIFO entry is written and read exactly once. We let $F_1 \cdots F_s$ represent the entire FIFO history, at the end of the procedure. It contains all visited configurations $F_k = I \circ T(\rho\, n_k) \in \mathbf{F}_2[1, n]$, in order $1 = n_1 < n_2 < \cdots < n_s$. The visit words $w_k = \mathbf{w}(n_k)$ are thus in lexicographic order $\epsilon = w_1 \prec w_2 \prec \cdots \prec w_s$, and they form a prefix-closed set: indeed, all pushed configurations *push*$(A.T[l])$ extend a word by one letter, i.e. $w_i = w_j \cdot l$ for some $j < i$ and letter $l \in \mathcal{A}$. It follows that $\mathbf{F}_2[F_1 \cdots F_s] = \mathbf{F}_2 \langle I \circ T(n) : n > 0 \rangle$.

Each visited configuration, for $j \in [1 \cdots s]$, is either a *base* configuration if $F_j \notin \mathbf{F}_2[F_1 \cdots F_{j-1}]$, or it is a *dependent* configuration if $F_j \in \mathbf{F}_2[F_1 \cdots F_{j-1}]$. Base configurations correspond to internal nodes, and dependent ones to leaves, in the complete α-ary tree representation. It follows that the number of visited configurations is proportional $s = 1 + (\alpha - 1)m$ to the final dimension m.

We let $[b_1 \cdots b_m]$ denote the minimal access pathes $b_{d_j} = n_j$ to the base configurations $F_j = I \circ T(\rho\, n_j)$. The set inclusion $\mathrm{ubd}(L) \subseteq \{b_1 \cdots b_m\} \subseteq \mathrm{index}(L)$ can be noted, where $\mathrm{ubd}(L)$ is the minimal base index (alg. 1) and $\mathrm{index}(L)$ is the regular index (def. 7) of $L = \Lambda_{\oplus}(A)$.

We prove, by induction on j, that upon extracting $F_j = pop(Q)$ from the FIFO, we have: the current dimension is $d_j = \dim \mathbf{F}_2[F[1] \cdots F[j-1]]$; the current automaton $A_j = (n, I_j, T_j(), F_j)$ is similar to the initial one $A_0 = A$, hence $\Lambda_\oplus(A_j) = \Lambda_\oplus(A)$; for all $r < d_j$, row $R_r = I_j \circ T_j(\rho\, b_r)$ is diagonal: $R_r[i] = \delta(i - r)$ for all $i \in [1 \cdots n]$.

By the diagonal form up to row d_j, vector $C = F_j \in \mathbf{F}_2[1, n]$ is dependent if and only if $\forall k > d_j : C[k] = 0$. In that case, the induction hypothesis apply directly to $j + 1$.

Otherwise, C is the base element $d_{j+1} = 1 + d_j$ if and only if $\exists k > d_j : C[k] = 1$. It is simply verified that the pivot similitude $A_j = P \circ A_{j-1} \circ P^-$ diagonalize row C, and the induction hypothesis carries over to $k + 1$.

At the final stage s, we have $d_s = m$, and the base rows $R_1 = D_1 \cdots R_m = D_m$ are all diagonal. The vector space generated by the accessible states has dimension 2^m, and $I \circ T(\rho\, n) \in \mathbf{F}_2[D_1 \cdots D_m]$ for all $n > 0$.

None of the remaining states $[m + 1 \cdots n]$ in A_s is accessible, and we can safely reduce the final automaton $B = \text{shrink}(A_s, m)$ to its final diagonal form.

Computing the pivot (alg. 4) requires $O(n^2)$ bit operations, and there are m pivots. So, the time complexity is $O(mn^2)$. All pivots can be performed in-place, within the $O(n^2)$ bits of memory required for representing A. Q.E.D.

5.3 Efficient NXA Operations

Proposition 9 *Let $L, L' \in$ RL be regular languages of respective dimensions $d = \dim(L)$ and $d' = \dim(L')$. All the following (in)equalities are tight:*

1. $d = \dim(\rho L)$.
2. $\dim(\neg L) = \dim(L \oplus \mathcal{A}^*) \leq \dim(L \cup \mathcal{A}^*) \leq d + 1$.
3. $\dim(L^*) \leq d - 1$ *when* $\epsilon \notin L$ *and* $L \neq \emptyset$.
4. $\dim(L \oplus L') \leq d + d'$.
5. $\dim(L \cdot L') \leq d + d'$.
6. $\dim(L \cap L') \leq d \times d'$.
7. $\dim(L \cup L') \leq d \times d'$.

Proof: (1.) is already proved, and (2.) is simple enough. Each of the remaining claims is established as follows: start from the operations defined over classical NFAs, say in [9], and verify for each that the multiplicity is preserved. Apply these constructions to the matrix representations of the minimal Xor automata of each operand; use the Xor interpretation to define the accepted languages; minimize all results by alg. 2; conclude by claims (3.) through (7.). Q.E.D.

6 Conclusion

The MXA is an attractive alternative to the MDA: never bigger, often 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 the minimal number of states, it need not have a minimal number of edges (see **X**13). In general, finding a NXA with minimal number of states & edges in polynomial time remains an open problem.

Another question relates the structure of the characteristic 2-polynomial **pol** (L) (defined in [33, 34]) to that of MXA(L), for a regular language $L \in$ RL. It is shown in [34] that **pol** (L) is a ***SNF*** for L, and that the 2-degree of **pol** (L) is bounded by the dimension dim L. Yet, no efficient algorithm is known for constructing the characteristic 2-polynomial.

In view of their complexity charts (fig. 1), it would be tempting to conclude that NXAs are always better than NFAs.

This would be too hasty, since there exist languages [31] for which the MXA is exponential in the size of some equivalent NFA. Interesting examples in that regard are provided by the regular expressions $E_n = (0{+}1)^* \cdot 1 \cdot (0{+}1)^n \cdot 1 \cdot (0{+}1)^*$ defined for all $n \in \mathbf{N}$. Expression E_0 is *ambiguous*, since the multiplicity of word 111 in E_0 is equal to 2. Likewise, expression E_n is ambiguous for all n, and the languages defined by the classical OR interpretation $O_n = \Lambda_\cup(E_n)$ and by the non-classical XOR interpretation $X_n = \Lambda_\oplus(E_n)$ are all different: $X_n \subset O_n$. It follows from the definition of E_n that O_n is recognized by a NFA with $n+3$ states, and that X_n is recognized by a NXA with $n+3$ states: both automata have the same graph, but both OR/XOR interpretations are different. It can be established, from the presented results, that the size of MXA(O_n) is exponential in n. On the other hand, we conjecture that the size of any NFAs for X_n is exponential in n.

Based on these and other examples, we conclude that, rather than one being better than the other, there is a deep theoretical duality going on between classical OR NFAs, and non-classical XOR NXAs.

To gain the status which NFAs enjoy in the world of automata, it remains to be seen if NXAs eventually find practical applications of similar size and scope.

Acknowledgments This work has benefited from valuable contributions by Jean-Baptiste Note.

References

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

[2] G. Berry and R. Sethi. From regular expressions to deterministic automata. In *Theoretical Computer Science*, volume 48, pages 117–126, 1986.

[3] J. Berstel and C. Reutenauer. *Rational Series and Their Languages*. EATCS Monograph. Springer Verlag, 1988. 2009 re-edition at http://www-igm.univ-mlv.fr/∼berstel/LivreSeries/LivreSeries8march2009.pdf.

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

[5] J. A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical theory of Automata*, pages 529–561. Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1962. Volume 12 of MRI Symposia Series.

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

[7] M. Domaratzki, D. Kisman, and J. 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.

[8] M. Domaratzki, D. Kisman, and J. Shallit. On the number of distinct languages accepted by finite automata with n states. *Journal of Automata, Languages and Combinatorics*, 7:469–486, 2002.

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

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

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

[12] W. J. Gilbert. *Modern Algebra with Applications*. J. Wiley & S., 1976.

[13] J. E. Hopcroft. An n log n algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[14] D. A. Huffman. The synthesis of sequential switching circuits. *The journal of symbolic logic*, 20:69–70, 1955.

[15] T. Jiang and B. Ravikumar. Minimal nfa problems are hard. In *ICALP*, pages 629–640, 1991.

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

[17] G. Kahn and D. B. MacQueen. Coroutines and networks of parallel processes. *Information Processing 77: Proceedings of the IFIP Congress 77*, North-Holland:993–998, 1977.

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

[19] D. E. Knuth. *The Art of Computer Programming, vol. 1, Fundamental Algorithms*. Addison Wesley, 3-rd edition, 1997.

[20] D. E. Knuth. *The Art of Computer Programming, vol. 3, Sorting and Searching*. Addison Wesley, 2-nd edition, 1998.

[21] D. E. Knuth. *The Art of Computer Programming, vol. 4A, Enumeration and Backtracking*. http://www-cs-faculty.stanford.edu/ uno/taocp.html, 2009.

[22] J. Massey. Shift register synthesis and bch decoding. In *Trans. on Information Theory*, volume IT-15, pages 122–127. IEEE, 1969.

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

[24] E. F. Moore. Gedanken-experiments on sequential machines. *The Journal of Symbolic Logic*, 23:60, 1958.

[25] A. Nerode. Linear automaton transformations. In *Proceedings of the AMS 9*, pages 541–544, 1958.

[26] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal*, 3:114–125, 1959.

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

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

[29] H. Tamm and E. Ukkonen. Bideterministic automata and minimal representations of regular languages. In *CIAA*, pages 61–71, 2003.

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

[31] L. van Zijl. On binary xor-nfas and succinct descriptions of regular languages. *Theoretical Computer Science*, 328(1-2):161–170, 2004.

[32] L. van Zijl and G. Muller. Minimization of unary symmetric difference nfas. In *Proc. of SAICSIT*, pages 125–134. ACM, 2004.

[33] 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.

[34] 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.

[35] B. W. Watson. Combining two algorithms by brzozowski. In D. Wood and S. Yu, editors, *Proceedings of the Fifth International Conference on Implementing Automata*, London, Canada, July 2001.