On the Hierarchy of Generalizations of One-Unambiguous Regular Languages

Pascal Caron[a], Ludovic Mignot[a], Clément Miklarz[a]

[a] *Laboratoire LITIS - EA 4108 Université de Rouen, Avenue de l'Université 76801 Saint-Étienne-du-Rouvray Cedex, France*

Abstract

A regular language is lookahead (resp. block) deterministic if it is specified by a lookahead (resp. block) deterministic regular expression. These two subclasses of regular languages have been respectively introduced by Han and Wood for lookahead determinism and by Giammarresi *et al.* for block determinism, as a possible extension of one-unambiguous languages defined and characterized by Brüggemann-Klein and Wood.

In this paper, we study the hierarchies and inclusion links of these families. We first show that each block deterministic language is the alphabetical image of some one-unambiguous language. Moreover, we show that deciding the block determinism of a regular language from its minimal DFA does not only require state elimination. Han and Wood state that there is a proper hierarchy in block deterministic languages, but prove it using this erroneous requirement. However, we prove their statement by giving our own parametrized family. We also prove that there is a proper hierarchy in lookahead deterministic languages by studying particular properties of unary regular expressions. Finally, using our valid results, we confirm that the family of block deterministic languages is strictly included in the one of lookahead deterministic languages by showing that any block deterministic unary language is one-unambiguous.

Keywords: One-unambiguous languages; block determinism; lookahead

Email addresses: pascal.caron@univ-rouen.fr (Pascal Caron), ludovic.mignot@univ-rouen.fr (Ludovic Mignot), clement.miklarz@univ-rouen.fr (Clément Miklarz)

1. Introduction

Every XML document is defined by an XML schema language. This makes XML schema languages useful for validation check, data translation and query optimization of XML documents. Among these schema languages, let us cite DTD [1], W3C XML Schema [2] and Relax NG [3]. Since people often use XML for representing and storing data, it is desirable to handle XML schemas and XML documents efficiently in many applications. Thus, we need to understand the fundamentals on XML schemas to design better algorithms. For instance, it is well known that XML DTD is an LL(1) context-free grammar. These grammars are made of rules whose right-hand parts are restricted regular expressions. Brüggemann-Klein and Wood [4] have formalized these regular expressions and showed that the set of languages specified by these restricted regular expressions is strictly included in the set of regular ones. The distinctive aspect of such expressions is the one-to-one correspondence between each letter of the input word and a unique occurrence of an alphabetical symbol in the expressions. The resulting Glushkov automata [5] for these regular expressions are deterministic and the specified languages are called one-unambiguous.

Several extensions of one-unambiguous regular expressions have been considered:

- a k-block deterministic regular expression [6] is such that while reading an input word, there is a one-to-one correspondence between the next at most k input symbols and the same number of symbols in the expression. The Glushkov automata of these expressions have transitions labelled by words of length at most k such that for any two distinct transitions going out of a same state, their labels are not prefix from each other.

- a k-lookahead deterministic regular expression [7] is such that the reading of the next k symbols of the input word allows to totally determine the following position in the expression.

- a (k, l)-unambiguous regular expression [8] is such that the next k symbols may induce several paths, but with a unique successor at a distance smaller than l.

These three families of expressions fit together as families of languages in the way that a language is k-block deterministic (resp. k-lookahead deterministic, (k, l)-unambiguous) if there exists a k-block deterministic (resp. k-lookahead deterministic, (k, l)-unambiguous) regular expression to represent it.

Han and Wood [7] show that there is a proper hierarchy in block deterministic languages and a strict inclusion of the family of block deterministic languages in the one of lookahead deterministic languages. However, they base their proofs on an erroneous statement due to Giammarresi *et al.* [6], invalidating them. In this paper, we first show that there is indeed a proper hierarchy in block deterministic languages by giving our own parametrized family. Then, we show that there is also a proper hierarchy in lookahead deterministic languages by studying the structural properties of unary lookahead deterministic Glushkov automata. Finally, using our valid results, we demonstrate that the family of block deterministic languages is strictly included in the one of lookahead deterministic languages by showing that any block deterministic unary language is one-unambiguous.

Preliminaries are gathered in Section 2. In Section 3, we recall several results from Giammarresi *et al.* [6] and Han and Wood [7] on which we question their truthfulness. Indeed, we show in Section 4 that due to an erroneous statement of Lemma 3, the witness family given as a proof of Theorem 3 is invalid; and present an alternative family, proving the infinite hierarchy of k-block deterministic languages with respect to k. In Section 5, we give another witness family to prove that there is also an infinite hierarchy in lookahead deterministic languages. Then, in Section 6, we give our own proof that the family of block deterministic languages is a proper subfamily of the one of lookahead deterministic languages.

2. Preliminaries

2.1. Languages and Automata Basics

Let Σ be a non-empty finite *alphabet* and w be a word over Σ. The *length* of w is denoted by $|w|$, and the *empty word* is denoted by ε.

Let Σ^* be the set of all words over Σ. A *language over* Σ is a subset of Σ^*. Usual operations on sets, like $\cup$, $\cap$, $\setminus$ (set difference) and $\triangle$ (symmetrical difference) are also defined on languages. Let L and L' be two languages over Σ. The *concatenation* $L \cdot L'$ is the set $\{w \cdot w' \mid w \in L \wedge w' \in L'\}$ and the *Kleene star* L^* is the set $\bigcup_{k\in\mathbb{N}} L^k$ with $L^0 = \{\varepsilon\}$ and $L^{k+1} = L \cdot L^k$.

A *regular expression over* Σ is built from $\emptyset$ (the empty set), ε, and symbols in Σ using the binary operators $+$ and $\cdot$, and the unary operator *. The *language* $\mathrm{L}(E)$ *specified by a regular expression* E is recursively defined as $\mathrm{L}(\emptyset) = \emptyset$, $\mathrm{L}(\varepsilon) = \{\varepsilon\}$, $\mathrm{L}(a) = \{a\}$, $\mathrm{L}(F + G) = \mathrm{L}(F) \cup \mathrm{L}(G)$, $\mathrm{L}(F \cdot G) = \mathrm{L}(F) \cdot \mathrm{L}(G)$, $\mathrm{L}(F^*) = \mathrm{L}(F)^*$ with a in Σ, and F, G some regular expressions over Σ. Given a language L, if there exists a regular expression E such that $\mathrm{L}(E) = L$, then L is *regular*. Thereafter, we consider only regular languages.

A regular expression is trim if it is equal to $\emptyset$ or does not contain any occurrence of $\emptyset$. In the following of the paper, only trim regular expressions will be considered.

A *finite automaton* A is a 5-tuple $(\Sigma, Q, I, F, \delta)$ with Q a finite set of states, $I \subset Q$ the set of initial states, $F \subset Q$ the set of final states, and $\delta \subset Q \times \Sigma \times Q$ the set of transitions. The set δ is equivalent to a function of $Q \times \Sigma \to 2^Q$: $(p, a, q) \in \delta \iff q \in \delta(p, a)$. This function can be extended to $2^Q \times \Sigma^* \to 2^Q$ such that for any subset Q' of Q, for any symbol a in Σ, for any word w in Σ^*: $\delta(Q', \varepsilon) = Q'$, $\delta(Q', a) = \bigcup_{q\in Q'} \delta(q, a)$, $\delta(Q', a \cdot w) = \delta(\delta(Q', a), w)$. Finally, we set $\delta(q, w) = \delta(\{q\}, w)$.

The *language* $\mathrm{L}(A)$ *recognized by* A is the set $\{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$. Two automata are *equivalent* if they recognize the same language. The *right language of a state* q *of* A is denoted by $\mathrm{L}_q(A) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$. Two states of an automaton are *equivalent* if they have the same right language.

Let p and q be two states in Q. Then q is a *direct successor of* p if there exists a transition (p, a, q) in δ.

An automaton is *trim* if all states can be reached from an initial state and can reach a final state. Every automaton considered in this paper is trim

An automaton A is *standard* [9] if it has only one initial state with no incoming transition. The operation of *standardization* allows one to obtain an equivalent standard automaton $(\Sigma, Q \cup \{i_s\}, \{i_s\}, F_s, \delta_s)$ with $F_s = F \cup \{i_s\}$ if $I \cap F \neq \emptyset$, F otherwise; and $\delta_s = \delta \cup \{(i_s, a, q) \mid \exists i \in I, (i, a, q) \in \delta\}$ from an automaton $(\Sigma, Q, I, F, \delta)$.

A *deterministic finite automaton* (DFA) is an automaton $A = (\Sigma, Q, I, F, \delta)$ such that $|I| = 1$ and for any two distinct transitions (p, a, q_1) and (p, b, q_2) in δ, $a \neq b$.

A DFA is *minimal* if there is no equivalent DFA with fewer states. Computing an equivalent minimal DFA is always possible by merging equivalent states [10]. Notice that two equivalent minimal DFA are isomorphic.

A subset O of Q is an *orbit* if it is a strongly connected component. An orbit is *trivial* if it consists of only one state with no self-loop. Let O be an orbit of A and p be a state of O. The state p is an *out-gate* (resp. *in-gate*) of O if p is final (resp. initial) or if there exists a transition (p, a, q) (resp. (q, a, p)) in δ such that q is not in O.

Kleene's Theorem [11] asserts that the set of languages specified by regular expressions is the same as the set of languages recognized by finite automata. The conversion of regular expressions into automata has been deeply studied [12, 13, 14, 5, 15]. In the following, we focus on the Glushkov construction [5] which compute an automaton reflecting the structural properties of the input regular expression E. It is based on functions considering the positions of the alphabetical symbols of E. To differentiate each occurrence of a same symbol in a regular expression, a *marking* is performed by indexing them with their relative position in the expression. The marking of a regular expression E produces a *marked regular expression* denoted by $E^\sharp$ over an alphabet of indexed symbols denoted by Π_E. Thus, any element of Π_E occurs exactly once in $E^\sharp$

and represents a unique position in E. The reverse of marking is the *dropping* of subscripts, denoted by $^\natural$, such that if $x \in \Pi_E$ and $x = a_k$, then $x^\natural = a$.

Let E be a regular expression over an alphabet Σ. The following functions are defined:

- $\mathrm{Null}(E) = \{\varepsilon\}$ if $\varepsilon \in \mathrm{L}(E)$, $\emptyset$ otherwise,
- $\mathrm{First}(E) = \{a \in \Sigma \mid \exists w \in \Sigma^*,\ aw \in \mathrm{L}(E)\}$,
- $\mathrm{Last}(E) = \{a \in \Sigma \mid \exists w \in \Sigma^*,\ wa \in \mathrm{L}(E)\}$,
- $\mathrm{Follow}(E, a) = \{b \in \Sigma \mid \exists u, v \in \Sigma^*,\ uabv \in \mathrm{L}(E)\}$, for any a in Σ.

From these functions, an automaton recognizing $\mathrm{L}(E)$ can be computed:

Definition 1. *The* Glushkov automaton of a regular expression E over an alphabet Σ *is denoted by* $G_E = (\Sigma, Q, \{0\}, F, \delta)$ *with:*

- $Q = \Pi_E \cup \{0\}$,
- $F = \mathrm{Last}(E^\sharp) \cup \{0\}$ *if* $\mathrm{Null}(E^\sharp) = \{\varepsilon\}$, $\mathrm{Last}(E^\sharp)$ *otherwise,*
- $\delta = \{(x, a, y) \mid y \in \mathrm{Follow}(E^\sharp, x) \wedge a = y^\natural\}$
  $\cup\ \{(0, a, y) \mid y \in \mathrm{First}(E^\sharp) \wedge a = y^\natural\}$.

Finally, an automaton is a *Glushkov automaton* if it is the Glushkov automaton of a regular expression E.

Example 1. *Let* $E = (a + b)^* a + \varepsilon$. *Then* $E^\sharp = (a_1 + b_2)^* a_3 + \varepsilon$ *with* $\Pi_E = \{a_1, b_2, a_3\}$, *and* G_E *is given in Figure 1.*

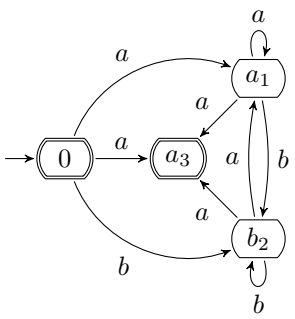

Figure 1: The Glushkov automaton G_E of $E = (a + b)^* a + \varepsilon$

2.2. One-Unambiguous Regular Languages

Brüggemann-Klein and Wood [4] define a regular expression as being *one-unambiguous* if its Glushkov automaton is deterministic. A language is then *one-unambiguous* if it can be specified by a one-unambiguous regular expression.

They show that the one-unambiguity of a regular language is structurally decidable over its minimal DFA. This decision procedure is related to the orbits of the underlying graph and their links with the outside: an automaton has the *orbit property* if every orbit has out-gates with identical connections to the outside. More formally:

Definition 2. *An automaton $A = (\Sigma, Q, I, F, \delta)$ has the orbit property if, for any orbit O of A and for any two out-gates g_1 and g_2 of O, the two following conditions are satisfied:*

- $g_1 \in F \iff g_2 \in F$,
- $\forall q \in (Q \setminus O), \forall a \in \Sigma, (g_1, a, q) \in \delta \iff (g_2, a, q) \in \delta$.

Let $q \in Q$ be a state. The *orbit of a state* q, denoted by $\mathrm{O}(q)$, is the orbit to which q belongs. The *orbit automaton A_q of the state q in A* is the automaton obtained by restricting the states and the transitions of A to $\mathrm{O}(q)$ with initial state q and with the out-gates as final states. Every language $\mathrm{L}(A_q)$, for q in Q, is called an orbit language of A.

A symbol a in Σ is *A-consistent* if there exists a state q_a in Q such that all final states of A have a transition labelled by a to q_a. A set of symbols S is A-consistent if every symbol is A-consistent. The *S-cut A_S* of A is constructed from A by removing all transitions labelled by a symbol in S that leave a final state of A. All these notions are used to characterize one-unambiguous languages:

Theorem 1 ([4])**.** *Let M be a minimal DFA and S be an M-consistent set of symbols. Then, $\mathrm{L}(M)$ is one-unambiguous if and only if:*

1. *the S-cut M_S of M has the orbit property,*

2. *all orbit languages of M_S are one-unambiguous.*

Furthermore, if M consists of a single non-trivial orbit and $\mathrm{L}(M)$ is one-unambiguous, then M has at least one M-consistent symbol.

This theorem suggests an inductive algorithm to decide, given a minimal DFA M, whether $\mathrm{L}(M)$ is one-unambiguous: the *BW-test*. Furthermore, it defines a sufficient condition over non-minimal DFA:

Lemma 1 ([4])**.** *Let A be a DFA and M be its equivalent minimal DFA.*

1. *If A has the orbit property, then so does M.*
2. *If all orbit languages of A are one-unambiguous, then so are all orbit languages of M.*

Consequently, the BW-test is extended to DFA which are not minimal. Reinterpreting the results in [4], it comes

Corollary 1. *The Glushkov automaton of a one-unambiguous regular expression passes the BW-test.*

2.3. Lookahead Deterministic Regular Languages

Lookahead determinism is a generalization of one-unambiguity suggested by Brüggemann-Klein and Wood and studied by Han and Wood [7]. It is specified in the same way *i.e.* a language is k-lookahead deterministic if it can be specified by a k-lookahead deterministic regular expression. The basic idea is that the reading of the next k symbols of the input word allows to totally determine the following position in the expression or in the automaton.

Definition 3. *An automaton $A = (\Sigma, Q, I, F, \delta)$ is* k-lookahead deterministic *for a positive integer k if $|I| = 1$ and for any two distinct transitions (p, a, q_1) and (p, b, q_2) in δ, either a is different from b, or for any words w in Σ^{k-1}, $\delta(q_1, w) = \emptyset$ or $\delta(q_2, w) = \emptyset$.*

Thus, a regular expression is k-lookahead deterministic if its Glushkov automaton is k-lookahead deterministic.

Since a 1-lookahead deterministic automaton is a DFA, the family of 1-lookahead deterministic languages is the same as the one of one-unambiguous languages.

Example 2. *Let $E = b^*a(b^*a)^*(a + b)$, G_E is given in Figure 2. Notice that the states a_2 and a_4 admit two successors by a and b. But $\delta(a_5, a) = \delta(a_5, b) = \delta(b_6, a) = \delta(b_6, b) = \emptyset$. Thus, following Definition 3, G_E and E are 2-lookahead deterministic.*

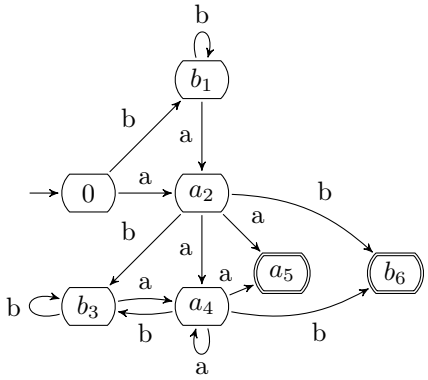

Figure 2: The 2-lookahead deterministic Glushkov automaton G_E

Thus, the language $\mathrm{L}(b^*a(b^*a)^*(a + b))$ is 2-lookahead deterministic and since it is not one-unambiguous [7], the family of one-unambiguous languages is strictly included in the one of lookahead deterministic languages.

2.4. Block Deterministic Regular Languages

Block determinism is another generalization of one-unambiguity suggested and studied by Giammarresi *et al.* [6]. As for lookahead determinism, k-block determinism regular languages are defined as those which can be specified by a k-block deterministic regular expression. The principle here is to read words of length at most k instead of just one letter.

Let Σ be an alphabet and k be an integer. The *set of blocks* $B_{\Sigma,k}$ is the set of non-empty words over Σ of length at most k. The notions of regular

expression and automaton can be extended to ones over set of blocks. Let Γ be a set of blocks, E be a regular expression over Γ and $A = (\Gamma, Q, I, F, \delta)$ be an automaton. Let Σ be an alphabet and k be an integer such that $\Gamma \subset B_{\Sigma,k}$, then E and A are (Σ, k)-*block*. And since $\Gamma \subset B_{\Sigma,k} \subset \Sigma^*$, a language over Γ is also a language over Σ. To distinguish blocks as syntactic components in block regular expressions, we write them between square brackets. Those are omitted for one letter blocks.

Since $\Sigma = B_{\Sigma,1}$, regular expressions and automata can be considered as ones over a set of blocks. Moreover, blocks can be treated as single symbols, as we do when we refer to the elements of an alphabet. With this assumption, the marking of block regular expressions induces the construction of their Glushkov automata, and the usual transformations on automata such as determinization and minimization can be easily performed.

Example 3. *Let* $E = [aa]^*([ab]b + ba)b^*$. *Then* $E^\sharp = [aa]_1^*([ab]_2 b_3 + b_4 a_5)b_6^*$, *and* G_E *is given in Figure 3.*

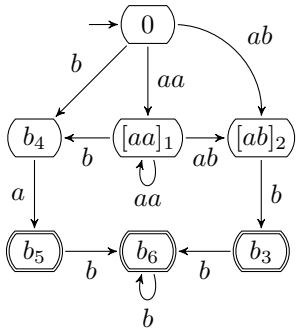

Figure 3: The $(\{a, b\}, 2)$-block Glushkov automaton G_E

The notion of determinism can also be extended to block determinism as follows:

Definition 4. *An automaton* $A = (\Gamma, Q, I, F, \delta)$ *is* k-block deterministic *if there exists an alphabet* Σ *such that* A *is* (Σ, k)*-block,* $|I| = 1$ *and for any two distinct transitions* (p, b_1, q_1) *and* (p, b_2, q_2) *in* δ, b_1 *is not a prefix of* b_2.

Thus, a block regular expression is k-block deterministic if its Glushkov automaton is k-block deterministic.

Since a 1-block deterministic automaton is a DFA, the family of 1-block deterministic languages is the same as the one of one-unambiguous languages.

Example 4. *Since the Glushkov automaton in Figure 3 is* 2*-block deterministic, the language* $\mathrm{L}([aa]^*([ab]b + ba)b^*)$ *is* 2*-block deterministic.*

Let $A = (\Sigma, Q, I, F, \delta)$ be an automaton and Γ be a set. The automaton $B = (\Gamma, Q, I, F, \delta')$ is *an alphabetical image of* A if there exists an injection φ from Σ to Γ such that $\delta' = \{(p, \varphi(a), q) \mid (p, a, q) \in \delta\}$. In this case, we set $B = \varphi(A)$.

Caron and Ziadi [16] show that an automaton is a Glushkov one if and only if the two following conditions hold:

- it is homogeneous (for any state q, for any two transitions (p, a, q) and (r, b, q), the labels a and b are the same),
- it satisfies some structural properties over its transitions [1], such as the orbit property.

One can check that any injection φ from Σ to Γ preserves such conditions, since the alphabetical image preserves the transition structure by only changing the labels of the transitions. Therefore

Lemma 2. *The alphabetical image of an automaton* A *is a Glushkov automaton if and only if* A *is a Glushkov automaton.*

Let us show that the BW-test can be used to determine the block determinism of a regular language:

Theorem 2. *A language is* k*-block deterministic if and only if it is recognized by a* k*-block deterministic automaton which is the alphabetical image of a DFA passing the BW-test.*

[1] these properties are called stability and transversality. More details are given in [16]

Proof. Let us show the double implication.

1. Let L be a k-block deterministic language over Σ. By definition, there exists a k-block deterministic Glushkov automaton $K = (B_{\Sigma,k}, Q, \{0\}, F, \delta_K)$ that recognizes L. Let $\Pi = \{[b] \mid b \in B_{\Sigma,k}\}$ be an alphabet and $\varphi : \Pi \to B_{\Sigma,k}$ be the bijection such that for any $[b]$ in Π, $\varphi([b]) = b$. Let $A = (\Pi, Q, \{0\}, F, \delta_A)$ be a Glushkov automaton such that $K = \varphi(A)$. Let us suppose that A is not deterministic. Since A admits a unique initial state, there exist two distinct transitions (p, a, q) and (p, a, r) in δ_A. Thus, the transitions $(p, \varphi(a), q)$ and $(p, \varphi(a), r)$ are distinct and belong to δ_K, which contradicts the fact that K is k-block deterministic. So, A is a deterministic Glushkov automaton, and therefore passes the BW-test following Corollary 1.
2. Let $A = (\Pi, Q_A, \{i_A\}, F_A, \delta_A)$ be a DFA which passes the BW-test, $K = \{\Gamma, Q_A, \{i_A\}, F_A, \delta_K)$ be a k-block deterministic automaton, and $\varphi : \Pi \to \Gamma$ be an injection such that $K = \varphi(A)$. Now, $\varphi : \Pi \to \Gamma$ is extended into the morphism $\varphi : \Pi^* \to \Gamma^*$ such that for any letter a in Π and any word w in Π^*, we have $\varphi(a \cdot w) = \varphi(a) \cdot \varphi(w)$ and $\varphi(\varepsilon) = \varepsilon$. Obviously, $\mathrm{L}(K) = \varphi(\mathrm{L}(A))$. Since A passes the BW-test, $\mathrm{L}(A)$ is one-unambiguous and there exists a deterministic Glushkov automaton $G = (\Pi, Q_G, \{i_G\}, F_G, \delta_G)$ such that $\mathrm{L}(G) = \mathrm{L}(A)$. Following Lemma 2, the automaton $H = (\Gamma, Q_G, \{i_G\}, F_G, \delta_H)$ such that $H = \varphi(G)$ is a Glushkov one, and $\mathrm{L}(H) = \varphi(\mathrm{L}(G))$. Since A and G are equivalent DFA, $\varphi(\mathrm{L}(G)) = \varphi(\mathrm{L}(A))$. And so, $\mathrm{L}(H) = \mathrm{L}(K)$.

   Let us suppose that H is not k-block deterministic. Following Definition 4, there exist two distinct transitions $(p_H, \varphi(a), q_H)$ and $(p_H, \varphi(b), r_H)$ in δ_H such that either $(\varphi(a) = \varphi(b)) \wedge (q_H \neq r_H)$ or $\varphi(a)$ is a prefix of $\varphi(b)$. By definition, (p_H, a, q_H) and (p_H, b, r_H) belong to δ_G. Since G and A are equivalent DFA, there exist two distinct transitions (p_A, a, q_A) and (p_A, b, r_A) in δ_A. And by definition, $(p_A, \varphi(a), q_A)$ and $(p_A, \varphi(b), r_A)$ belong to δ_K. Two cases may occur. Let us suppose that:

   (a) $(\varphi(a) = \varphi(b)) \wedge (q_H \neq r_H)$. Since φ is an injection, we have $(a = b)$

and $(q_H \neq r_H)$, which means that G is not deterministic.

(b) $\varphi(a)$ is a prefix of $\varphi(b)$. Following Definition 4, K is not k-block deterministic.

Both cases lead to a contradiction. Therefore, H is a k-block deterministic Glushkov automaton, and $\mathrm{L}(K)$ is k-block deterministic.

□

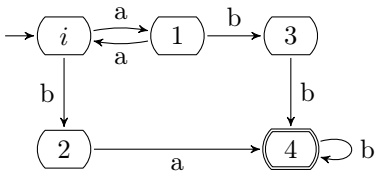

Figure 4: The minimal DFA of $\mathrm{L}(E)$

It has been proved that the family of one-unambiguous languages is a proper subfamily of the one of k-block deterministic languages. As an example, the language $\mathrm{L}([aa]^*([ab]b + ba)b^*)$ is 2-block deterministic but not one-unambiguous since its minimal DFA given in Figure 4 does not pass the BW-test. Therefore one can wonder whether there exists an infinite hierarchy in k-block deterministic languages regarding k. That has been achieved by Han and Wood [7], but with an invalid assumption.

3. Previous Results on Block Deterministic Regular Languages

Giammarresi *et al.* [6] present a method for creating from a block automaton an equivalent block automaton with larger blocks by eliminating states while preserving the right languages of other states.

Let $A = (\Gamma, Q, I, F, \delta)$ be a block automaton. The *state elimination of q in A* creates a new block automaton computed as follows: first, the state q and all transitions going in and out of it are removed; second, for any two transitions (r, u, q) and (q, v, s) in δ, the transition (r, uv, s) is added. This transformation is illustrated in Figure 5.

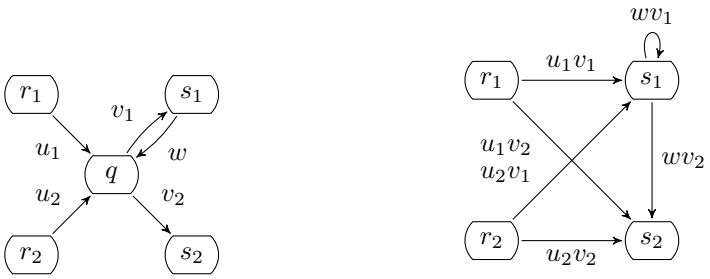

Figure 5: The state elimination of q

Definition 5. *Let A be a block automaton. A state q of A satisfies the* state elimination precondition *if it is neither initial nor final and has no self-loop.*

State elimination is extended to a set of states S if every state in S satisfies the state elimination precondition, and if the subgraph induced by S is acyclic. If so, we can eliminate every state in S in any order. Giammarresi *et al.* [6] suggest that state elimination is sufficient to decide the block determinism of a language.

Lemma 3 ([6, 7])**.** *Let M be the minimal DFA of a k-block deterministic language. We can transform M to a k-block deterministic automaton that satisfies the orbit property using state elimination.*

Let $k > 1$ be an integer and L_k be the language specified by the regular expression $(a^k)^*(a^{k-1}bb + ba)b^*$. From the minimal DFA M_k of L_k represented in Figure 6, Han and Wood state that:

Theorem 3 ([7])**.** *There is a proper hierarchy in k-block deterministic languages.*

Proof. It is already known that there exist 1-block deterministic languages since they are one-unambiguous. Thus, we only need to check that there exist k-block deterministic languages for any integer $k > 1$, which are not $(k-1)$-block deterministic. Since L_k can be specified by the k-block deterministic regular expression $([a^k])^*([a^{k-1}b]b + ba)b^*$, it is k-block deterministic. Moreover, the minimal DFA M_k has two non-trivial orbits. Following Lemma 3, there is no

other choice but to eliminate $k-1$ states (q_1 to q_{k-1}), in any order, to have the orbit property. Thus, L_k is not $(k-1)$-block deterministic. □

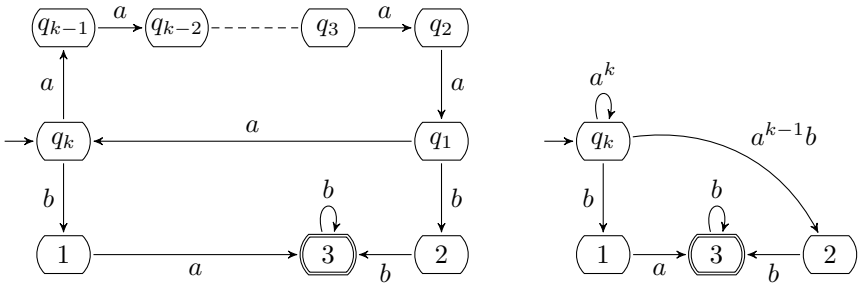

Figure 6: The minimal DFA M_k and its equivalent k-block deterministic automaton after having eliminated states q_1 to q_{k-1}

4. A Witness for the Infinite Hierarchy of Block Deterministic Regular Languages

In this section, we exhibit a counter-example for Lemma 3. We can find a block deterministic language with a minimal DFA from which we cannot get any block deterministic automaton that satisfies the orbit property. In Figure 7, the leftmost automaton is minimal and none of its states can be eliminated. However, by applying standardization, we create an equivalent DFA from which we can eliminate the state i to get the rightmost equivalent 2-block deterministic automaton.

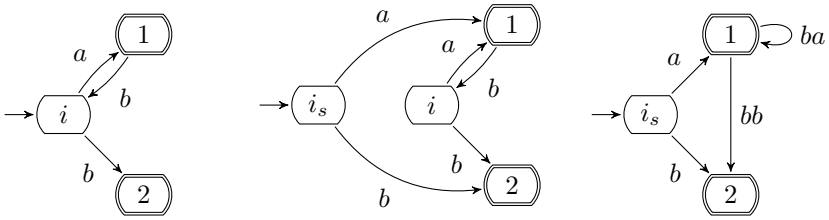

Figure 7: The counter-example by applying standardization

This clearly shows that state elimination alone is not enough to decide whether a language is block deterministic. Using this operation, we show that:

Proposition 1. *For any integer $k > 1$, the language L_k is 2-block deterministic.*

Proof. As showed in Figure 8, we can always standardize M_k, proceed to the state elimination of q_k and get a 2-block deterministic automaton which respects the conditions stated in Theorem 2. Thus, the language L_k is 2-block deterministic and is specified by the 2-block deterministic regular expression $(a^{k-1}([aa]a^{k-2})^*([ab]a + bb) + ba)b^*$.

□

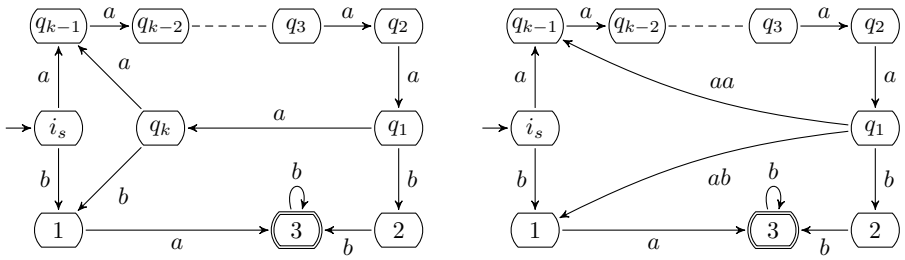

Figure 8: The standardization of M_k followed by the state elimination of q_k

However, Theorem 3 is still correct since we can give proper details about the proof with our own parametrized family of languages.

Definition 6. *Let $n > 1$ be an integer. The automaton $B_n = (\{a, b, c\}, Q, I, F, \delta)$ is defined by:*

- $Q = \{f\} \cup \{\alpha_j, \beta_j \mid 1 \leq j \leq n\}$,
- $I = \{\beta_n\}$,
- $F = \{f\} \cup \{\alpha_n, \beta_n\}$,
- $\delta = \{(\beta_n, a, \alpha_n), (\beta_1, b, f), (\alpha_n, a, \alpha_n), (\alpha_1, b, f), (\alpha_1, c, \beta_n)\}$
  $\cup \{(\alpha_j, b, \alpha_{j-1}), (\beta_j, b, \beta_{j-1}) \mid 2 \leq j \leq n\}$.

The family of automata $\mathcal{B}$ is then defined by $\{B_j \mid j \in \mathbb{N} \setminus \{0, 1\}\}$.

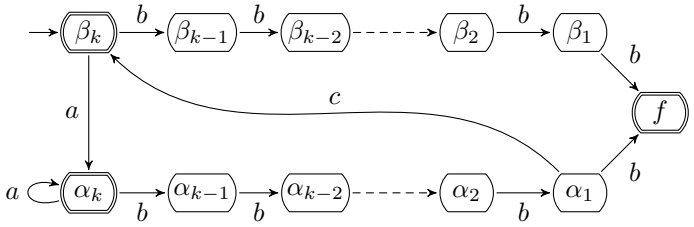

Figure 9: An automaton B_k of the family $\mathcal{B}$

By construction, every automaton of $\mathcal{B}$ is trim and deterministic. Furthermore, let us notice that the word b^j belongs to $\mathrm{L}(B_k)$ if and only if $j = k$. Thus, for any two different integers k and k', the languages $\mathrm{L}(B_k)$ and $\mathrm{L}(B_{k'})$ are different. Moreover, it can be checked that $\mathrm{L}(B_k)$ is specified by the k-block deterministic regular expression $(a(\varepsilon + [b^{k-1}c]))^*(\varepsilon + [b^k])$. Thus,

Proposition 2. *For any automaton B_k of $\mathcal{B}$, the language $\mathrm{L}(B_k)$ is k-block deterministic.*

Finally, let us show that the index cannot be reduced:

Proposition 3. *For any automaton B_k of $\mathcal{B}$, the language $\mathrm{L}(B_k)$ is not $(k-1)$-block deterministic.*

Proof. Let $B_k = (\{a, b, c\}, Q, I, F, \delta)$ be an automaton of $\mathcal{B}$ with $\alpha_1, \ldots, \alpha_k$, $\beta_1, \ldots, \beta_k, f$ its states, and $A = (B_{\Sigma,k-1}, Q_A, \{i_A\}, F_A, \delta_A)$ be a $(k-1)$-block deterministic automaton equivalent to B_k.

We first show that there exists a non-trivial orbit O in A and two states o_1 and o_2 in O such that $\mathrm{L}_{o_1}(A) = \mathrm{L}_{\alpha_k}(B_k)$ and $\mathrm{L}_{o_2}(A) = \mathrm{L}_{\beta_k}(B_k)$. Let us consider the following state sequences: $(\varphi_j)_{j\in\mathbb{N}} \subset F_A$ and $(\psi_j)_{j\in\mathbb{N}} \subset F_A$, such that $\psi_0 = i_A$, $\delta_A(\psi_j, a) = \varphi_j$ and $\delta_A(\varphi_j, b^{k-1}c) = \psi_{j+1}$. It follows that $\delta_A(i_A, (ab^{k-1}c)^j) = \psi_j$ and $\delta_A(i_A, (ab^{k-1}c)^j a) = \varphi_j$. Notice that the existence of φ_j and ψ_j is ensured since $\mathrm{L}(A) = \mathrm{L}(B_k)$. Let us suppose that there exists an integer j such that $\mathrm{L}_{\psi_j}(A) \neq \mathrm{L}_{\beta_k}(B_k)$. Then there exists a word w such that $w \in \mathrm{L}_{\psi_j}(A) \triangle \mathrm{L}_{\beta_k}(B_k)$. And since $\delta(\beta_k, (ab^{k-1}c)^j) = \beta_k$, $(ab^{k-1}c)^j \cdot w \in \mathrm{L}(A) \triangle \mathrm{L}(B_k)$. Thus, $\mathrm{L}(A) \neq \mathrm{L}(B_k)$ which is contradictory. So, we have

$\mathrm{L}_{\psi_j}(A) = \mathrm{L}_{\beta_k}(B_k)$ for any integer j. The proof that $\mathrm{L}_{\varphi_j}(A) = \mathrm{L}_{\alpha_k}(B_k)$ for any integer j is done in the same way. Now, let us suppose that $\varphi_j \neq \varphi_{j'}$ and $\psi_j \neq \psi_{j'}$ for any two different integers j and j'. Then Q_A would be infinite, which would contradict the fact that A is a finite automaton. So, there exist two integers $j < j'$ such that $\varphi_j = \varphi_{j'}$ or $\psi_j = \psi_{j'}$. Thus, either there exist a path going from ψ_j to φ_j and a path going from φ_j to $\psi_{j'} = \psi_j$, which implies that ψ_j and φ_j belong to the same orbit; or there exist a path going from φ_j to ψ_{j+1} and a path going from ψ_{j+1} to $\varphi_{j'} = \varphi_j$, which implies that φ_j and ψ_{j+1} belong to the same orbit.

Finally, let us focus on such an orbit O with two out-gates o_1 and o_2 such that $\mathrm{L}_{o_1}(A) = \mathrm{L}_{\alpha_k}(B_k)$ and $\mathrm{L}_{o_2}(A) = \mathrm{L}_{\beta_k}(B_k)$. We know that $\delta(\beta_k, b^i) = \beta_{k-i}$ for any positive integer $i < k$, with $|\mathrm{L}_{\beta_{k-i}}(B_k)| < \infty$. Since $\mathrm{L}_{o_2}(A) = \mathrm{L}_{\beta_k}(B_k)$ and A is $(k-1)$-block deterministic, there exist a positive integer $j < k$ and a state p in Q_A such that $\delta_A(o_2, b^j) = p$ and $\mathrm{L}_p(A) = \mathrm{L}_{\beta_{k-j}}(B_k)$. This means that $|\mathrm{L}_p(A)| < \infty$, so $p \notin O$. Now, if there does not exist a state q in Q_A such that $\delta_A(o_1, b^j) = q$, then A does not have the orbit property. So, let us suppose that such a state exists. We know that $\delta(\alpha_k, b^i) = \alpha_{k-i}$ for any positive integer $i < k$, with $|\mathrm{L}_{\alpha_{k-i}}(B_k)| = \infty$. Since $\mathrm{L}_{o_1}(A) = \mathrm{L}_{\alpha_k}(B_k)$, we have $\mathrm{L}_q(A) = \mathrm{L}_{\alpha_{k-j}}(B_k)$ and $|\mathrm{L}_q(A)| = \infty$. Thus p and q are distinct and A does not have the orbit property.

Since $\mathrm{L}(B_k)$ cannot be recognized by a $(k-1)$-block deterministic alphabetical image of an automaton passing the BW-test, following Theorem 2 it holds that $\mathrm{L}(B_k)$ is not $(k-1)$-block deterministic. □

5. A Witness for the Infinite Hierarchy of Lookahead Deterministic Regular Languages

In this section, we give a parametrized family $(\mathrm{L_j})_{j\geq 1}$ such that $\mathrm{L_j}$ is $(j+1)$-lookahead deterministic but not j-lookahead deterministic. We show that any j-lookahead deterministic Glushkov automaton can not recognize $\mathrm{L_j}$ which proves our assumption.

Definition 7. *Let n be a positive integer. The unary automaton $A_n = (\{a\}, Q, I, F, \delta)$ is defined by:*

- $Q = \{\alpha_i \mid 0 \leq i \leq 2n\}$,
- $I = \{\alpha_0\}$,
- $F = \{\alpha_0, \alpha_n\}$,
- $\delta = \{(\alpha_i, a, \alpha_{i+1}) \mid 0 \leq i < 2n\} \cup \{(\alpha_{2n}, a, \alpha_0)\}$.

The family of unary automata $\mathcal{A}$ is then defined by $\{A_j \mid j \in \mathbb{N} \setminus \{0\}\}$.

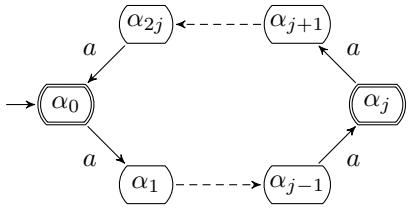

Figure 10: An automaton A_j of the family $\mathcal{A}$

Let us first show that distinct automata of the family $\mathcal{A}$ recognize distinct regular languages and also that these languages are lookahead deterministic.

Proposition 4. *For any automaton A_j of $\mathcal{A}$, A_j is a minimal DFA.*

Proof. By construction, any automaton A_j of $\mathcal{A}$ is trim and deterministic. Moreover, there are always two distinct final states α_0 and α_j such that a^j is in $\mathrm{L}_{\alpha_0}(A_j)$ and not in $\mathrm{L}_{\alpha_j}(A_j)$. Thus, the states α_0 and α_j are not equivalent, and so are all non-final states. Therefore, any automaton A_j of $\mathcal{A}$ is also minimal. □

Thus, for any two different automata A_j and $A_{j'}$ of $\mathcal{A}$, since their set of states are different, the languages $\mathrm{L}(A_j)$ and $\mathrm{L}(A_{j'})$ are different. Furthermore, it can be checked that $\mathrm{L}(A_j)$ is specified by the $(j+1)$-lookahead deterministic regular expression $(a^{2j+1})^* \cdot (\varepsilon + a^j)$. Thus,

Proposition 5. *For any automaton A_j of $\mathcal{A}$, the language $\mathrm{L}(A_j)$ is $(j+1)$-lookahead deterministic.*

Let A_j be an automaton of $\mathcal{A}$ and G be a j-lookahead deterministic Glushkov automaton. We demonstrate that G cannot recognize $\mathrm{L}(A_j)$ and therefore, that $\mathrm{L}(A_j)$ is not j-lookahead deterministic. In order to do so, we consider a property of Glushkov automata from Proposition 4.2 of [16]:

Lemma 4 ([16])**.** *Let G be a Glushkov automaton and O be a non-trivial orbit of G. Then, for any in-gate g_i and any out-gate g_o of O, g_i is a direct successor of g_o.*

Let us first restrain the set of Glushkov automata to consider. We show that a state in a unary lookahead deterministic automaton does not admit two distinct direct successors which have infinite right languages.

Proposition 6. *Let A be a unary lookahead deterministic automaton. Then for any state of A, at most one of its direct successors has an infinite right language.*

Proof. Let $A = (\{a\}, Q, \{0\}, F, \delta)$, and p and q be two distinct states in Q such that they are direct successors of a same state, and have both an infinite right language. Then necessarily, for any positive integer j, we have $\delta(p, a^{j-1}) \neq \emptyset$ and $\delta(q, a^{j-1}) \neq \emptyset$, which contradicts the fact that A is lookahead deterministic. □

As a direct consequence, we can deduce a property over the orbits of unary lookahead deterministic Glushkov automata.

Proposition 7. *Let G be a unary lookahead deterministic Glushkov automaton recognizing an infinite language. Then G has exactly one non-trivial orbit, with a single in-gate and a single out-gate, such that among the direct successors of any state of this orbit, at most one is in the orbit.*

Proof. Since G has only one initial state, following Proposition 6, there is at most one non-trivial orbit in it. Now, let us suppose that G has no non-trivial orbit, then $\mathrm{L}(G)$ is finite. Thus, G must have a single non-trivial orbit.

Let O be this orbit. The fact that O has a single in-gate g_i is a direct consequence of Proposition 6. Furthermore, let us suppose that O has two distinct out-gates g_{o1} and g_{o2}. Since G is a Glushkov automaton, following Lemma 4, g_i is a direct successor of both g_{o1} and g_{o2}. Consequently, there exists a state in O with two direct successors in O, which both have infinite right languages, contradicting Proposition 6.

The fact that every state of O has exactly one direct successor in O can be deduced in the same way. □

Moreover, since the language recognized by an automaton A_j of $\mathcal{A}$ is unary, it can be totally ordered with respect to the lengths of its words. And since $\mathrm{L}(A_j) = \mathrm{L}((a^{2j+1})^* \cdot (\varepsilon + a^j)) = \{\varepsilon, a^j, a^{2j+1}, a^{3j+1}, a^{4j+2}, \cdots\}$, we can give a property over the lengths of three consecutive words in it:

Lemma 5. *Let A_j be an automaton of $\mathcal{A}$ and w_1, w_2 and w_3 be three consecutive words in $\mathrm{L}(A_j)$. Then, either $|w_3| - |w_2| = j + 1$ and $|w_2| - |w_1| = j$, or $|w_3| - |w_2| = j$ and $|w_2| - |w_1| = j + 1$.*

Finally, we show that for any automaton A_j of $\mathcal{A}$, the language $\mathrm{L}(A_j)$ cannot be recognized by a j-lookahead deterministic Glushkov automaton.

Proposition 8. *Let G be a unary Glushkov automaton and A_j be an automaton of $\mathcal{A}$. If G is j-lookahead deterministic, then G does not recognize $\mathrm{L}(A_j)$.*

Proof. Let $G = (\{a\}, Q, \{0\}, F, \delta)$. Since for any automaton A_j of $\mathcal{A}$, the language $\mathrm{L}(A_j)$ is infinite, if $\mathrm{L}(G)$ is finite, then it is different from $\mathrm{L}(A_j)$. So let us suppose that G is j-lookahead deterministic and recognize an infinite language. Following Proposition 7, let O be the single non-trivial orbit of G, $l_O = |O|$ be the size of O, g_i be the single in-gate of O and g_o be the single out-gate of O.

First, let us characterize the words reaching g_o from the initial state 0. Following Proposition 7, every state o of O has exactly one direct successor in O and o reaches itself by a word w if and only if there exists an integer k such that $|w| = k \times l_O$. And following Lemma 4, since g_i is a direct successor of g_o, g_i reaches g_o by a word w if and only if there exists a positive integer k such

that $|w| = k \times l_O - 1$. Now, as a corollary of Proposition 6, and since G has a single initial state, there exists a single state s outside of O such that g_i is a direct successor of s, and s can be reached from the initial state by a single word u such that $|u| = m$. Thus, g_o can be reached from 0 by a word w if and only if there exists a positive integer k such that $|w| = m+1+k\times l_O-1$. Thus,

$$|w| = m + k \times l_O \tag{1}$$

Now, let us show that G recognizes a different language than $\mathrm{L}(A_j)$. Let $Q_o = \delta(g_o, a) \setminus O$ be the set of direct successors of g_o outside of O, and $\mathrm{L_o} = \bigcup_{q\in Q_o} \mathrm{L}_q(G)$ be the union of their right language. Let us consider the set $\mathrm{L_{out}}$ of words reaching a final state from g_o without going through O. By definition, $\mathrm{L_{out}} = \{a\} \cdot \mathrm{L_o} \cup \{\varepsilon\}$ if $g_o \in F$, $\{a\} \cdot \mathrm{L_o}$ otherwise. Since G is j-lookahead deterministic, the length of any word of $\mathrm{L_o}$ is strictly smaller than $j-1$ and the length of any word in $\mathrm{L_{out}}$ is strictly smaller than j. Thus, three cases may occur. Let us suppose that:

1. $\mathrm{L_{out}} = \emptyset$, then $\mathrm{L}(G)$ is finite which is contradictory.
2. there exist two distinct words w_{o1} and w_{o2} in $\mathrm{L_{out}}$ such that $|w_{o2}| > |w_{o1}|$. Since the lengths of w_{o1} and w_{o2} are strictly smaller than j, we have $|w_{o2}|-|w_{o1}| < j$. Thus, for any word w reaching g_o from 0, the words ww_{o1} and ww_{o2} both belong to $\mathrm{L}(G)$, and $|ww_{o2}| - |ww_{o1}| = |w_{o2}| - |w_{o1}| < j$. Following Lemma 5, $\mathrm{L}(G) \neq \mathrm{L}(A_j)$.
3. $\mathrm{L_{out}} = \{w_o\}$ such that $|w_o| = m'$. Since O is the only non-trivial orbit and has a single out-gate, there exists an integer n such that for every word w_G in $\mathrm{L}(G)$, if $|w_G| \geq n$ then $w_G = w_p w_s$ such that w_p reach g_o from 0 and w_s reach a final state from g_o. Since $\mathrm{L_{out}} = \{w_o\}$, we have $w_s = w_o$. And following (1), we have $|w_p| = m + k \times l_O$. Thus, for any word w_G in $\mathrm{L}(G)$ such that $|w_G| \geq n$, there exists a positive integer k such that $|w_G| = m + k \times l_O + m'$. Let w_{G1}, w_{G2} and w_{G3} be three consecutive words of $\mathrm{L}(G)$ such that their lengths are greater than or equal to n. Then there exist three positive integers k_1, k_2 and k_3 such that $|w_{G1}| = m + m' + k_1 \times l_O$, $|w_{G2}| = m + m' + k_2 \times l_O$ and

$|w_{G3}| = m + m' + k_3 \times l_O$. Since w_{G1}, w_{G2} and w_{G3} are consecutive, we have $k_2 = k_1 + 1$ and $k_3 = k_2 + 1$. Thus, $|w_{G3}| - |w_{G2}| = |w_{G2}| - |w_{G1}| = l_O$ and following Lemma 5, $\mathrm{L}(G) \neq \mathrm{L}(A_j)$.

In any case, if G is j-lookahead deterministic, it does not recognize $\mathrm{L}(A_j)$.

□

Consequently:

Proposition 9. *For any automaton A_j of $\mathcal{A}$, the language $\mathrm{L}(A_j)$ is not j-lookahead deterministic.*

We can conclude that:

Theorem 4. *There is a proper hierarchy in k-lookahead deterministic languages.*

6. Block Deterministic Regular Languages Are a Proper Subfamily of Lookahead Deterministic Regular Languages

The fact that k-block deterministic languages are a subfamily of k-lookahead deterministic languages for any positive integer k can be directly deduced from the Glushkov construction. Indeed, let E_b be a k-block deterministic regular expression, and E be a regular expression constructed by replacing every block $[b_1 \cdots b_n]$ in E_b by the expression $b_1 \cdots b_n$. Then E specifies the same language as E_b and for any position x in Π_E, we have $|\mathrm{Follow}(E^\sharp, x)| > 1$ if and only if $x^\natural$ is the end of a block in E_b. And thus, each distinct position following x in E is the beginning of a distinct block in E_b. Moreover, since E_b is k-block deterministic, any two different positions in E_b following a same position refer to blocks of size at most k which are not prefix from each other. Thus, for any position in E, the reading of k symbols totally determines the next following position. This means that the Glushkov automaton of E is k-lookahead deterministic, and so is E.

Han and Wood [7] state that block deterministic languages are a proper subfamily of lookahead deterministic languages. However, their proof cites a

statement made by Giammarresi *et al.* [6] about the family of languages $\mathrm{L}((a+b)^*a(a+b)^{k-1})$ not being block deterministic. But we prove that Lemma 3, which they use as a basis for deciding if a language is block deterministic, is wrong. So we cannot be sure that their example is not block deterministic and give our own proof, using properties of unary block deterministic languages.

Proposition 10. *Let E be a block regular expression over a set of unary blocks. If E is block deterministic, then $|\mathrm{First}(E^\sharp)| \leq 1$ and for any position x in Π_E, $|\mathrm{Follow}(E^\sharp, x)| \leq 1$.*

Proof. If $|\mathrm{First}(E^\sharp)| > 1$, then there exist two distinct positions in $\mathrm{First}(E^\sharp)$. Since E is defined over unary blocks, one of these two positions is a prefix of the other. Following Definition 1 and Definition 4, the resulting Glushkov automaton is not block deterministic, and the same goes for E. Therefore $|\mathrm{First}(E^\sharp)| \leq 1$, and the same reasoning can be applied to $\mathrm{Follow}(E^\sharp, x)$ for any position x of $E^\sharp$. □

Let us notice that if each position of a regular expression E is followed by at most one position, then its Glushkov automaton is necessarily deterministic and E is one-unambiguous. Now, if we consider a block regular expression such that each position is followed by at most one position, then replacing every block $[b_1 \cdots b_n]$ by the expression $b_1 \cdots b_n$ does not alter this property. Thus:

Lemma 6. *Let E be a block regular expression. If $|\mathrm{First}(E^\sharp)| \leq 1$ and for any position x in Π_E, $|\mathrm{Follow}(E^\sharp, x)| \leq 1$, then $\mathrm{L}(E)$ is one-unambiguous.*

Consequently:

Theorem 5. *If a unary language is block deterministic, then it is one-unambiguous.*

Moreover, we show in Section 5 that for any automaton A_j of $\mathcal{A}$ with j a positive integer, the language $\mathrm{L}(A_j)$ is $(j+1)$-lookahead deterministic without being j-lookahead deterministic. Thus, it is not 1-lookahead deterministic (one-unambiguous), and following Theorem 5, it is not block deterministic. Therefore:

Proposition 11. *For any integer $k > 1$, there exist unary languages which are k-lookahead deterministic without being block deterministic.*

Finally:

Theorem 6. *For any integer $k > 1$, the family of k-block deterministic languages is strictly included in the one of k-lookahead deterministic languages.*

7. Conclusion and Perspectives

In this paper, we demonstrate that despite some erroneous results, there exists an infinite hierarchy in block deterministic languages. We show that such an infinite hierarchy also exists in lookahead deterministic languages. And finally, showing that block-deterministic unary languages are one-unambiguous, we give our own proof of the family of block deterministic languages being strictly included in the one of lookahead deterministic languages.

From these results, one can wonder whether there exists a k-lookahead deterministic language which is $(k+1)$-block deterministic without being k-block deterministic. Another open problem is the decidability of the lookahead determinism of a language. Finally, the decidability of the block determinism of a language has been studied by Giammarresi *et al.* but proved with Lemma 3 which we invalidate. Thus, this problem is still open.

Bibliography

[1] A. Layman, E. Jung, E. Maler, H. Thompson, J. Paoli, J. Tigue, N. Mikula, S. DeRose, XML-Data, URL `http://www.w3.org/TR/1998/NOTE-XML-data`, 1998.

[2] W3C XML Schema definition language, URL `http://www.w3.org/TR/xmlschema11-1`, 2012.

[3] RELAX NG home page, URL `http://relaxng.org`, 2014.

[4] A. Brüggemann-Klein, D. Wood, One-Unambiguous Regular Languages, Inf. Comput. 140 (2) (1998) 229–253, URL http://dx.doi.org/10.1006/inco.1997.2688.

[5] V. M. Glushkov, The Abstract Theory of Automata, Russian Mathematical Surveys 16 (5) (1961) 1–53, URL http://dx.doi.org/10.1070/rm1961v016n05abeh004112.

[6] D. Giammarresi, R. Montalbano, D. Wood, Block-Deterministic Regular Languages, in: A. Restivo, S. R. D. Rocca, L. Roversi (Eds.), Theoretical Computer Science, 7th Italian Conference, ICTCS 2001, Torino, Italy, October 4-6, 2001, Proceedings, vol. 2202 of *Lecture Notes in Computer Science*, Springer, 184–196, URL http://dx.doi.org/10.1007/3-540-45446-2_12, 2001.

[7] Y. Han, D. Wood, Generalizations of 1-Deterministic Regular Languages, Inf. Comput. 206 (9-10) (2008) 1117–1125, URL http://dx.doi.org/10.1016/j.ic.2008.03.013.

[8] P. Caron, M. Flouret, L. Mignot, (k, l)-Unambiguity and Quasi-Deterministic Structures: An Alternative for the Determinization, in: A. H. Dediu, C. Martín-Vide, J. L. Sierra-Rodríguez, B. Truthe (Eds.), Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings, vol. 8370 of *Lecture Notes in Computer Science*, Springer, 260–272, URL http://dx.doi.org/10.1007/978-3-319-04921-2_21, 2014.

[9] J. Sakarovitch, R. Thomas, Elements of Automata Theory, Cambridge University Press, ISBN 978-0-521-84425-3, URL http://dx.doi.org/10.1017/cbo9781139195218, 2009.

[10] E. F. Moore, Gedanken-Experiments on Sequential Machines, Automata studies (1956) 129–153.

[11] S. C. Kleene, Representation of Events in Nerve Nets and Finite Automata, Automata Studies Ann. Math. Studies 34 (1956) 3–41, princeton U. Press.

[12] J. A. Brzozowski, Derivatives of Regular Expressions, Journal of the Association for Computing Machinery 11 (4) (1964) 481–494, URL http://doi.acm.org/10.1145/321239.321249.

[13] K. Thompson, Programming Techniques: Regular expression search algorithm, Communications of the ACM 11 (6) (1968) 419–422, URL http://dx.doi.org/10.1145/363347.363387.

[14] R. F. McNaughton, H. Yamada, Regular Expressions and State Graphs for Automata, IEEE Transactions on Electronic Computers 9 (1960) 39–57.

[15] V. M. Antimirov, Partial Derivatives of Regular Expressions and Finite Automaton Constructions, Theor. Comput. Sci. 155 (2) (1996) 291–319, URL http://dx.doi.org/10.1016/0304-3975(95)00182-4.

[16] P. Caron, D. Ziadi, Characterization of Glushkov automata, Theor. Comput. Sci. 233 (1-2) (2000) 75–90, URL http://dx.doi.org/10.1016/S0304-3975(97)00296-X.