Information and Computation 206 (2008) 1117–1125

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Generalizations of 1-deterministic regular languages

Yo-Sub Han [a,*,1], Derick Wood [b]

[a] Intelligence and Interaction Research Center, Korea Institute of Science and Technology, P.O. Box 131, Cheongnyang, Seoul, Republic of Korea
[b] Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 28 June 2007
Revised 18 February 2008
Available online 6 June 2008

Keywords:
One-unambiguous regular languages
k-Lookahead determinism
k-Block determinism

ABSTRACT

We examine two generalizations of 1-deterministic regular languages that are used for the content models of DTDs in XML. They are *k-lookahead determinism* and *k-block-determinism*. The k-lookahead determinism uses the first k symbols $w_1w_2\cdots w_k$ of the current input string as lookahead to process the first symbol w_1. On the other hand, the k-block-determinism takes k $w_1w_2\cdots w_k$ as lookahead and process the whole k symbols. We show that there is a hierarchy in k-lookahead determinism and there is a proper hierarchy in k-block-determinism. Moreover, we prove that k-block-deterministic regular languages are a proper subfamily of deterministic k-lookahead regular languages.

1. Introduction

When people make a document, each one has his/her own writing style. However, we often need a fixed format to make documents consistent especially when these documents are company reports or research papers with many co-authors. People realized the need for a standard rule and introduced the Standard Generalized Markup Language (SGML) [14]. Later, when the web became popular and lots of documents were displayed on the web or processed online, researchers removed some complicated parts in SGML and added new features that are suitable for the web and introduced the Extensible Markup Language (XML) [2].

Both SGML and XML have the Document Type Definition (DTD), which is a grammar to describe how documents to be written. The DTD is an extended context-free grammar [17]. Given a DTD, its left-hand side is a nonterminal and its right-hand side, which is called a *content model*, is an extended regular expression. However, not all regular languages can be used for content models. The SGML and XML standards require regular expressions for content models to be *unambiguous* in the sense that, using formal language terminology, “a symbol that occurs in a string accepted by a regular expression E must be able to satisfy only one occurrence of this symbol in E without looking ahead in the input string.” Hence, DTDs are LL(1) grammars [17]. It turns out that the unambiguity in DTDs is equal to the 1-determinism studied by Brüggemann-Klein and Wood [3,4]. A regular language L is 1-deterministic if there is a regular expression E such that $L = L(E)$ and the corresponding position automaton [10,11,15] is deterministic. In other words, a lookahead of one symbol when processing a string from left to right determines a unique next position in E. Note that not all regular languages are 1-deterministic; namely, 1-deterministic regular languages are a proper subfamily of regular languages [4].

LL(1) languages are the counterpart of 1-deterministic regular languages in context-free languages. Since there is a proper hierarchy in LL(k) languages [1], it is natural to generalize 1-determinism in regular languages and examine whether or not there is a similar hierarchy in the generalized determinism. Giammarresi et al. [9] pointed out that there are two possible

* Corresponding author.
E-mail addresses: emmous@kist.re.kr (Y.-S. Han), dwood@cs.ust.hk (D. Wood).
[1] Han was supported by the KIST TSI Grant.

0890-5401/$ - see front matter © 2008 Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.03.013

generalizations of 1-determinism. The first is based on a lookahead of at most $k \geq 1$ symbols to determine the next, at most one, matching position in a given regular expression. The second is similar except that when we use a lookahead of l symbols, we must match the next l positions uniquely at a single step. The first notion defines *deterministic k-lookahead* regular expressions and the second notion defines *k-block-deterministic* regular expressions. The two notions are certainly different. We compare the two notions and investigate, for each notion, whether or not there exists a proper hierarchy in regular languages.

In Section 2, we define some basic notions. In Section 3, we give a formal definition of deterministic k-lookahead regular languages and show that there is a hierarchy in k-lookahead determinism. Then, we examine k-block-deterministic regular languages and demonstrate that there is a proper hierarchy in k-block-determinism in Section 4. Moreover, we prove that k-block-deterministic regular languages are a proper subfamily of deterministic k-lookahead regular languages. We conclude the paper with an open problem in Section 5.

2. Preliminaries

Let Σ denote a finite alphabet of characters and Σ^* denote the set of all strings over Σ. A language over Σ is any subset of Σ^*. The symbol $\emptyset$ denotes the empty language and the symbol λ denotes the null string. Given two strings x and y over Σ, we say that x is a prefix of y if there is a string $w \in \Sigma^*$ such that $xw = y$. Given a set X of strings, X is *prefix-free* if no string in X is a prefix of any other string in X.

A finite-state automaton (FA) A is specified by a tuple (Q,Σ,δ,s,F), where Q is a finite set of states, Σ is an input alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. Given a transition (p,a,q) in δ, where $p,q \in Q$ and $a \in \Sigma$, we say that p has an *out-transition* and q has an *in-transition*. Furthermore, p is a *source state* of q and q is a *target state* of p. A string x over Σ is accepted by A if there is a labeled path from s to a state in F such that this path spells out the string x. Thus, the language $L(A)$ of an FA A is the set of all strings that are spelled out by paths from s to a final state in F. We assume that an FA A has only *useful* states; that is, each state appears on some path from the start state to some final state.

For complete background knowledge in automata theory, the reader may refer to textbooks [13,16].

3. Deterministic k-lookahead regular languages

We examine the first alternative of generalization of 1-determinism suggested by Giammarresi et al. [9]: If we use a lookahead of k symbols, we must determine at most one matching position.

3.1. Deterministic k-lookahead regular expressions

Given a regular expression E, we assign a unique positive integer for each character appearance, from 1 to m, where m is the total number of appearances and denote it by E'. We call E' a *marked* regular expression and the integers *positions*. For example, if $E = (a+b)^*a(a+b)$, then $E' = (1+2)^*3(4+5)$. Note that E' is defined over $\mathbb{N} = \{1,2,\ldots,m\}$ and there is a mapping *i2c* from E' to E. The mapping $i2c(p)$ gives the corresponding character in E for a given position p. We can restore characters from a marked regular expression using the *i2c* mapping. We denote this operation as $\natural$. If E is a regular expression over $\mathbb{N}$, then $E^\natural$ is the regular expression over Σ that is obtained from E by restoring characters from positions using the *i2c* mapping. The restoring operation can be extended to strings and languages. Then, we have $L(E^\natural) = L(E)^\natural$. Let w_i denote the ith character of string w.

Definition 1. A regular expression E is deterministic k-lookahead if and only if, for all strings u,v,w over $\mathbb{N} = \{1,2,\ldots,m\}$ and all strings x,y of length k over $\mathbb{N}$, the conditions $uxv,uyw \in L(E')$ and $x \neq y$ imply either $x^\natural \neq y^\natural$ or $x^\natural = y^\natural$ and $x_1 = y_1$, where m is the total number of character appearances of E. A regular language is deterministic k-lookahead if it is defined by some deterministic k-lookahead regular expression.

In other words, for each string w of the language denoted by a deterministic k-lookahead regular expression E, there is one marked string v in $L(E')$ such that $v^\natural = w$. Furthermore, v can be computed incrementally by examining the next k symbols of w. It is not difficult to see that this definition is independent of the marking E' chosen for E.

Given a regular expression E and a regular language L, we define $E_\# = E \cdot \#^k$ and $L_\# = \{x\#^k \mid x \in L\}$, where $\# \notin \Sigma$ to ensure that $E_\#$ has at least k character appearances and each string in $L_\#$ has at least length k. Note that this appending procedure does not affect E or L at all.

Definition 2. For a language L, we define the following three sets:

$first_k(L) = \{b \mid \text{there is a string } w \text{ such that } bw \in L_\#, |b| = k\}$.
$last_k(L) = \{b \mid \text{there is a string } w \text{ such that } wb \in L_\#, |b| = k\}$.
$follow_k(L,a) = \{b \mid \text{there are strings } v \text{ and } w \text{ such that } vabw \in L_\#, |b| = k\}$, for each symbol a.

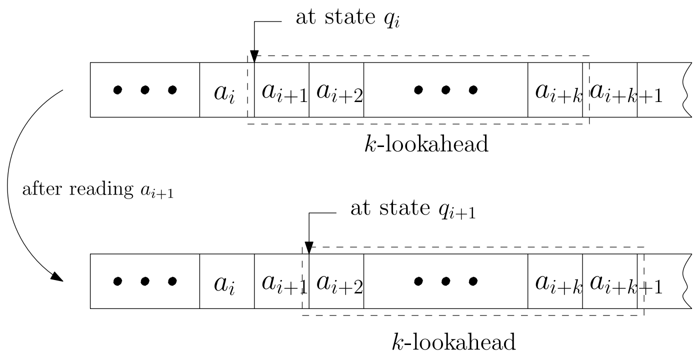

Fig. 1. From the current state q_i, we determine a next state q_{i+1} using the k-lookahead from the current input string position a_{i+1}. Note that after reading the single character a_{i+1}, we append a_{i+k+1} to make a new k-lookahead to determine the next state from state q_{i+1}.

We can extend these sets to regular expressions; for example, given a regular expression E, $\mathit{first}_k(E) = \mathit{first}_k(L(E))$.

Lemma 3. *For a marked regular expression $E'_\#$ and an integer $k \geq 1$, an integer string $x_1 \cdots x_n$ belongs to $L(E'_\#)$ if and only if the following three conditions hold*:

(1) $x_1x_2 \cdots x_k \in \mathit{first}_k(E'_\#)$
(2) $x_{n-k+1}x_{n-k+2} \cdots x_n \in \mathit{last}_k(E'_\#)$
(3) $x_{i+1}x_{i+2} \cdots x_{i+k} \in \mathit{follow}_k(E'_\#,x_i)$, for all, $1 \leq i \leq n-k$

The proof is a straightforward induction on $E'_\#$. It is essential, though, that $E'_\#$ is marked; for the regular expression aaa, the string aa is not in $L(aaa)$, although $aa \in \mathit{first}_2(aaa) \cap \mathit{last}_2(aaa) \cap \mathit{follow}_2(aaa,a)$.

Lemma 3 shows that, for each marked regular expression $E_\#$, the conditions $uzv,xzy \in L(E_\#)$ and $|z| = k$ imply $uzy,xzv \in L(E_\#)$. Therefore, we can give an alternative characterization of deterministic k-lookahead regular expressions.

Lemma 4. *Given an integer $k \geq 1$, a regular expression $E_\#$ is deterministic k-lookahead if and only if the following two conditions hold*:

(1) For all $x,y \in \mathit{first}_k(E'_\#)$, $x \neq y$ implies either $x^\natural \neq y^\natural$ or $x^\natural = y^\natural$ and $x_1 = y_1$.
(2) For all $z \in \{1,2,\ldots,m\}$ and $x,y \in \mathit{follow}_k(E'_\#,z)$, $x \neq y$ implies either $x^\natural \neq y^\natural$ or $x^\natural = y^\natural$ and $x_1 = y_1$.

We give an example of a regular expression E and its marked expression $E'_\#$, and demonstrate how to decide whether or not $E_\#$ is deterministic k-lookahead.

Example: Given a regular expression $E = (a+b)^*a(a+b)$, when $k = 2$, $E_\# = (a+b)^*a(a+b)\#\#$ and $E'_\# = (1+2)^*3(4+5)67$. Then, $\mathit{first}_k(E'_\#) = \{11,12,21,22,13,23,34,35\}$. Let $x = 11$ and $y = 34$. Note that $x \neq y$ but $x^\natural = y^\natural = aa$ and $x_1 = 1 \neq 3 = y_1$. Therefore, $E_\#$ is not deterministic 2-lookahead by Lemma 4.

3.2. Deterministic k-lookahead position automata

We examine the structural properties of deterministic k-lookahead regular expressions in position automata. Glushkov [10,11] suggested the position construction[2] in 1960 and McNaughton and Yamada [15] also presented it independently at about the same time. The construction is based on the three sets, $\mathit{first}_1(E)$,$\mathit{last}_1(E)$ and $\mathit{follow}_1(E,i)$ of positions of a regular expression E. For details on the position construction and its structural properties, refer to Caron and Ziadi [6].

Let an FA $A = (Q,\Sigma,\delta,s,F)$ be a deterministic finite-state automaton (DFA) and a string $w = a_1a_2 \cdots a_n$ be in $L(A)$. Then, there exists a unique accepting sequence, $s,q_1,q_2,\ldots,q_n$ such that $(s,a_1,q_1) \in \delta$, $(q_{i-1},a_i,q_i) \in \delta$, for $2 \leq i \leq n$, and $q_n \in F$.

Suppose that we want to find an accepting sequence by scanning an input string w once from left to right in a nondeterministic position automaton A. Assume that we are at state q_i after reading $a_1a_2 \cdots a_i$ in A. It would be desirable if we can determine the next state q_{i+1} and read out a_{i+1} by looking at the next k symbols $a_{i+1}a_{i+2} \cdots a_{i+k}$. If a position automaton has this property for all states, we call it a *deterministic k-lookahead* position automaton.

The intuitive idea in a deterministic k-lookahead position automaton is that if we are constructing an accepting sequence from an input string $w = a_1a_2 \cdots a_n$ and we already have $s,q_1,\ldots,q_i$ after reading $a_1a_2 \cdots a_i$, then we can identify q_{i+1} by looking at the next k symbols of w. Note that if we do not see all of w when determining q_{i+1}, then we do not really know

[2] In the early literature, it was called *Glushkov construction*. Recently, some researchers started to call it position construction since the construction is based on positions of characters in a given regular expression.

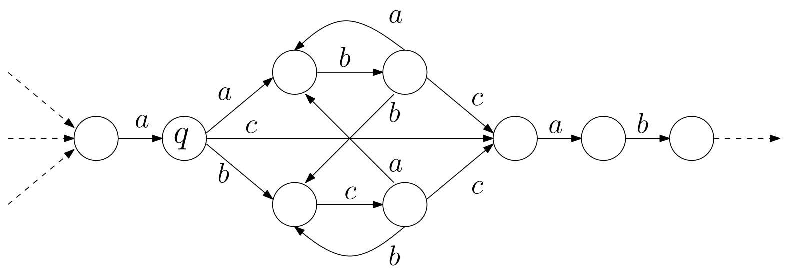

Fig. 2. A part of a position automaton.

whether w will be ultimately accepted or not. Thus, the deterministic k-lookahead condition implies that q_{i+1} is independent (except for the next k symbols) of what is the remaining part of w.

Definition 5. Given a deterministic k-lookahead position automaton A and a state q, we define *window* $\mathbb{F}_k(q)$ to be the set of all strings of length k spelled out from q in A.

For example in Fig. 2, $\mathbb{F}_3(q) = \{aba,abb,abc,bca,bcb,bcc,cab\}$.

Given a set $\mathbb{F}_k = \{\alpha_1,\alpha_2,\ldots,\alpha_m\}$ of strings of length k, and a string β, we define a catenation $\beta \cdot \mathbb{F}_k$ as follows:

$$\beta \cdot \mathbb{F}_k = \{\beta\alpha_1,\beta\alpha_2,\ldots,\beta\alpha_m\}.$$

Definition 6. Let an FA A be a position automaton and q_0 be a state that has (q_0,a_0,q_0), (q_0,a_1,q_1), (q_0,a_2,q_2), ..., (q_0,a_m,q_m) as out-transitions in A, where $q_i \neq q_j$ for $0 \leq i,j \leq m$. Then, A is deterministic k-lookahead if, for each state q_0 in A and $i \neq j$,

$$a_i \cdot \mathbb{F}_{k-1}(q_i) \cap a_j \cdot \mathbb{F}_{k-1}(q_j) = \emptyset.$$

Stated less formally, A is deterministic k-lookahead if given a string $xy \in \Sigma^*$ and the first k symbols of y, there is at most one out-transition that can spell out the k symbols from a state that we reach after reading x in A.

Assume that we have computed the window for all states in a given deterministic k-lookahead position automaton A. Then, when we read a string $w = a_1a_2a_3\cdots$ at a state q in A, we check whether or not $\mathbb{F}_k(q)$ has a string $a_1a_2\cdots a_k$. If $a_1a_2\cdots a_k \notin \mathbb{F}_k(q)$, then we immediately know that $w \notin L(A)$. Otherwise, we compute a set $\{q_1,q_2,\ldots,q_m\}$ of states reachable from q with a_1. If $m = 1$, then we read a_1 and move to q_1 in A. If $m > 1$, then we search for a state q' whose window F_{k-1} has $a_2a_3\cdots a_k$ and move to the state after reading a_1. Note that q' is unique since A is deterministic k-lookahead.

This procedure gives an algorithm that determines whether or not a given position automaton is deterministic k-lookahead in polynomial time. We first construct the window $\mathbb{F}_{k-1}$ for each state q and check the condition in Definition 6 for each pair of out-transitions from q.

Theorem 7. *A regular expression E is deterministic k-lookahead if and only if its position automaton A is deterministic k-lookahead.*

Proof. Assume that A is not deterministic k-lookahead. This implies that there is a state q in A such that

$$a_i \cdot \mathbb{F}_{k-1}(q_i) \cap a_j \cdot \mathbb{F}_{k-1}(q_j) \neq \emptyset,$$

where $(q,a_i,q_i),(q,a_j,q_j) \in \delta$ and $q_i \neq q_j$. Let $w \in a_i \cdot \mathbb{F}_{k-1}(q_i) \cap a_j \cdot \mathbb{F}_{k-1}(q_j)$ and z be the corresponding position of q in E'. Because of w, there are two distinct paths that spell out w from q in A and, thus, there are two integer strings $x,y \in \mathit{follow}_k(E',z)$, where $x^\natural = y^\natural$. (If q is the start state, then we can find such x and y in $\mathit{first}_k(E')$.) Note that $x_1 \neq y_1$ since $q_i \neq q_j$. This violates the condition in Lemma 4—a contradiction.

Assume that E is not deterministic k-lookahead. This implies that there are two integer strings x,y of E' such that $x,y \in \mathit{follow}_k(E',z)$ or $x,y \in \mathit{first}_k(E')$ for a position z of E' and $x^\natural = y^\natural$ but $x_1 \neq y_1$. Let q,q_i and q_j be the corresponding states for z,x_1 and y_1 in A, respectively. Then, $(q,x_1^\natural,q_i) \in \delta$ and $(q,y_1^\natural,q_j) \in \delta$, where $x_1^\natural = y_1^\natural$. This shows that for a state q,

$$x_1^\natural \cdot \mathbb{F}_{k-1}(q_i) \cap y_1^\natural \cdot \mathbb{F}_{k-1}(q_j) \neq \emptyset$$

since $x^\natural$ belongs to both $x_1^\natural \cdot \mathbb{F}_{k-1}(q_i)$ and $y_1^\natural \cdot \mathbb{F}_{k-1}(q_j)$ — a contradiction. □

3.3. Hierarchy of deterministic k-lookahead regular languages

The notion of k-lookahead determinism is very similar to the notion of LL(k) context-free languages. Note that there is a proper hierarchy in LL(k) context-free languages [1]; namely, there are some languages that are defined by LL(k) grammars but cannot be defined by any LL($k-1$) grammars. Brüggemann-Klein and Wood [4] demonstrated that there are some regular languages that are not deterministic 1-lookahead, which is *one-unambiguous* in their terminology. Therefore, it is natural to investigate whether or not there is a proper hierarchy based on k-lookahead determinism in regular languages.

Recall that a regular language L is deterministic k-lookahead if L is denoted by a deterministic k-lookahead regular expression E by Definition 1. Brüggemann-Klein and Wood [4] showed that $L((a+b)^*a(a+b)^n)$, for each $n \geq 1$, is not deterministic 1-lookahead. We, then, examine the regular expression $(a+b)^*a$. Note that the regular expression $(a+b)^*a$ is not deterministic 1-lookahead whereas the corresponding regular language $L = L((a+b)^*a)$ is deterministic 1-lookahead since L can be defined by a deterministic 1-lookahead regular expression; $L = L((a+b)^*a) = L(b^*a(a+bb^*a)^*)$ and the regular expression $b^*a(a+bb^*a)^*$ is deterministic 1-lookahead as shown in Fig. 3.

Lemma 8. *There exists a strictly deterministic 2-lookahead regular language.*

Proof. Let $L = L((a+b)^*a(a+b))$. Note that L is not deterministic 1-lookahead [4]. To show that L is deterministic 2-lookahead, it is enough to give a deterministic 2-lookahead regular expression. Fig. 4 shows the position automaton for $b^*a(a+bb^*a)^*(a+b)$.

It is easy to verify that the position automaton in Fig. 4 is a deterministic 2-lookahead position automaton. Therefore, L is a strictly deterministic 2-lookahead regular language. □

Using an argument similar to that of Lemma 8, we establish the following statement.

Corollary 9. $L((a+b)^*a(a+b)^k)$, *for* $k \geq 0$, *is deterministic* $(k+1)$*-lookahead.*

Corollary 9 shows that there is a hierarchy of deterministic k-lookahead regular languages. However, it is not sufficient to show that the hierarchy is proper. For example, we do not know if the deterministic 3-lookahead regular language $L((a+b)^*a(a+b)^2)$ can be defined by a deterministic 2-lookahead regular expression. Thus, the problem for showing whether or not there is a proper hierarchy in deterministic k-lookahead regular languages is still open.

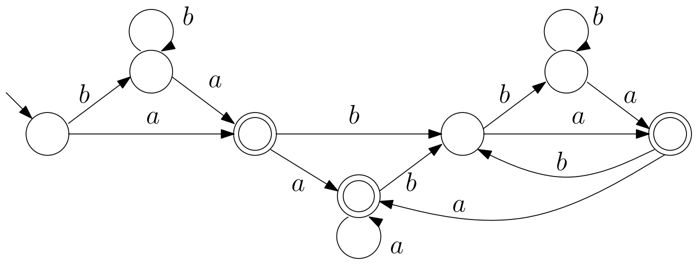

Fig. 3. The position automaton for $b^*a(a+bb^*a)^*$. Note that the FA is deterministic and, thus, $L(b^*a(a+bb^*a)^*)$ is deterministic 1-lookahead.

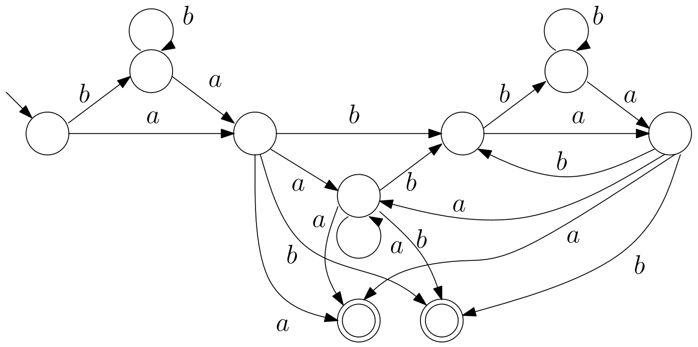

Fig. 4. The position automaton for $b^*a(a+bb^*a)^*(a+b)$.

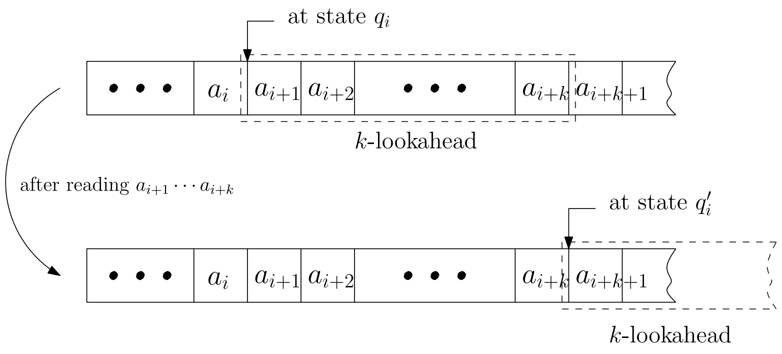
at state q_i
a_i a_{i+1} a_{i+2} a_{i+k} a_{i+k+1}
k-lookahead
after reading $a_{i+1} \cdots a_{i+k}$
at state q_i'
k-lookahead

Fig. 5. We are currently at state q_i after reading $a_1 \cdots a_i$. Now if we use a lookahead of k symbols $a_{i+1} \cdots a_{i+k}$ to determine the next state, then we read the whole $a_{i+1} \cdots a_{i+k}$ and move to a_{i+k+1}. Note that we have read the whole k symbols and, thus, it is certainly different from the case in Fig. 1.

4. *k*-block-deterministic regular languages

We examine the second alternative of generalization of 1-determinism suggested by Giammarresi et al. [9]: If we use a lookahead of k symbols, we must match the next k positions uniquely. Fig. 5 illustrates it.

As depicted in Fig. 5, we read a block of characters (k symbols) at a single step. This leads us to examine *block automata*,[3] which were introduced by Eilenberg [7]. Block automata allow transition labels to be nonempty strings or *blocks* over Σ instead of single characters. A block automaton is specified by a tuple $(Q,\Sigma,\Gamma,\delta,s,F)$, where Γ is a set of subsets Σ^+ of Σ and $\delta \subseteq Q \times \Gamma \times Q$. If the maximum block size of a given block automaton A is k, then we say that A is a *k-block automaton*. For example, a traditional FA is an 1-block automaton.

Let $\mathbb{B}(q)$ be the set of blocks of all out-transitions of q in A. Giammarresi and Montalbano [8] defined deterministic block automata as follows:

Definition 10. A block automaton $A = (Q,\Sigma,\Gamma,\delta,s,F)$ is a deterministic block automaton if the following conditions hold:

(1) For two transitions (q,x,p_1) and (q,y,p_2), $x = y$ if and only if $p_1 = p_2$, where x and y are strings over Σ.
(2) For each state q, $\mathbb{B}(q)$ is prefix-free.
A DFA, for instance, is a deterministic block automaton with $k = 1$.

Recently, Han and Wood [12] re-examined expression automata, which allow regular expressions on transitions, and introduced deterministic expression automata based on prefix-freeness.

Giammarresi et al. [9] introduced k-block-deterministic regular languages as an extension of 1-deterministic regular languages. Note that a regular language L is 1-deterministic if there exists a deterministic position automaton A such that $L = L(A)$. For a state q of a nondeterministic finite-state automaton (NFA) A, we define the *orbit* of q, denoted by $\mathbb{O}(q)$, to be the strongly connected component of q in A; that is, it is the set of states of A that can be reached from q and from which q can be reached. We consider the orbit of q to be *trivial* if it consists of only the state q and there are no transitions from q to itself in A.

Definition 11 *(Brüggemann-Klein and Wood [4]).* A state q of an NFA A is a *gate* of its orbit $\mathbb{O}(q)$ if q is a final state or q has an out-transition to a state outside $\mathbb{O}(q)$. A has the *orbit property* if all the gates of each orbit have identical connections to the outside world. More precisely, if any pair q_1 and q_2 of gates in the same orbit satisfies the following two conditions:

(1) q_1 is a final state if and only if q_2 is a final state.
(2) For all states q outside the orbit of q_1 and q_2, there is a transition (q_1,a,q) in A if and only if there is a transition (q_2,a,q) in A.

Brüggemann-Klein and Wood [4] observed that a position automaton always has the orbit property and if a given position automaton is deterministic, then the orbit property is preserved under the state minimization. Based on these observations, they designed an algorithm that determines whether or not a given minimal DFA M defines a 1-deterministic

[3] Block automata were called generalized automata by Eilenberg [7].

regular language; the algorithm checks whether or not M satisfies the orbit property. Furthermore, they showed that if M defines a 1-deterministic regular language, then we can compute a 1-deterministic regular expression for $L(M)$.

Definition 12. We define a regular language L to be k-block-deterministic if there exists a k-block automaton $A' = (Q,\Sigma,\Gamma,\delta,s,F)$ that satisfies the following conditions:

(1) A' is a position automaton over Γ.
(2) A' is a deterministic block automaton.
(3) $L = L(A')$.

It is easy to verify that a position automaton A for a 1-deterministic regular language is 1-block-deterministic.

Lemma 13 *(Giammarresi et al. [9]). Let M be a minimal DFA for a k-block-deterministic regular language. We can transform M to a deterministic k-block automaton that satisfies the orbit property using state elimination.*

Roughly speaking, state elimination of a state q in an FA is the bypassing of state q, q's in-transitions, q's out-transitions and q's self-looping transition with equivalent expression transition sequences. For details on state elimination, refer to Brzozowski and McCluskey Jr. [5] or Wood [16].

Fig. 6 gives an example of Lemma 13: An FA A does not satisfy the orbit property since q_1 and q_2 are gates of the same orbit and they have different target states for the same label b. On the other hand, once we eliminate q_2 from A, then the resulting block automaton A' satisfies the orbit property. Thus, $L(A)$ is 2-block-deterministic but not 1-block-deterministic.

Now a natural question from Fig. 6 is whether or not there exists a k-block-deterministic regular language that is not $(k-1)$-block-deterministic. In other words, whether or not there is a proper hierarchy in k-block-determinism.

Theorem 14. *There is a proper hierarchy in k-block-deterministic regular languages.*

Proof. A $(k-1)$-block-deterministic regular language is k-block-deterministic by definition. Thus, it is enough to show that there is a k-block-deterministic regular language that is not $(k-1)$-block-deterministic.

Let L be $L((a^k)^*(a^{k-1}bb + ba)b^*)$ and A be its minimal DFA as shown in Fig. 7. Note that A has two orbits $\mathbb{O}(q_1)$ and $\mathbb{O}(q_5)$ and does not satisfy the orbit property since two gates q_1 and q_2 of $\mathbb{O}(q_1)$ do not have the same target state for the label b. If L is a $(k-1)$-block-deterministic regular language, then we should be able to transform A to a deterministic $(k-1)$-block automaton that satisfies the orbit property using state elimination due to Lemma 13. However, the only way of transforming A to a block automaton that satisfies the orbit property is to eliminate all states of $\mathbb{O}(q_1)$ except for q_1. Otherwise, there are always two gates for $\mathbb{O}(q_1)$ and the two gates have different target states. Thus, we eliminate $k-2$ states and the maximum block size is k not $k-1$. Namely, we cannot have a $(k-1)$-block automaton that satisfies the orbit property. Therefore, L is k-block-deterministic but not $(k-1)$-block-deterministic. □

We have examined two different ways of generalizing 1-determinism. The two definitions, one is deterministic k-lookahead and the other is k-block deterministic, are certainly different. Then, the next question is how different they are.

Theorem 15. *k-block-deterministic regular languages are a proper subfamily of deterministic k-lookahead regular languages.*

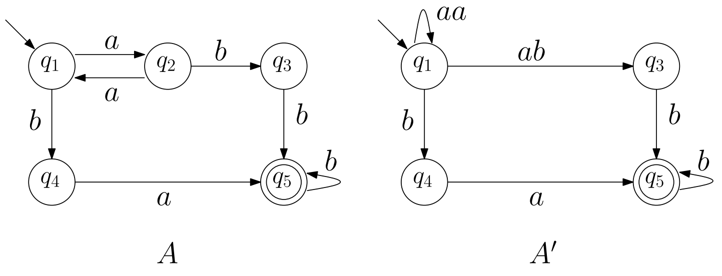

Fig. 6. An FA A is the minimal DFA for $L((aa)^*(abb + ba)b^*)$ and A' is a deterministic block automaton that satisfies the orbit property. Note that we obtain A' from A by the state elimination of q_2.

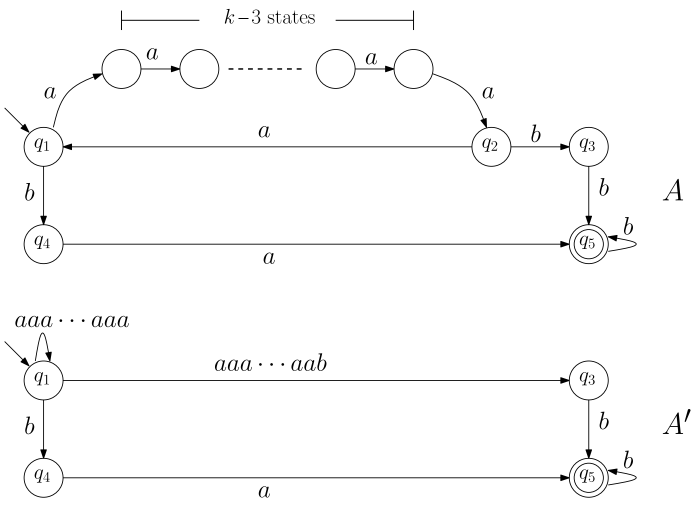

Fig. 7. An FA A is the minimal DFA for $L((a^k)^*(a^{k-1}bb+ba)b^*)$ and A' is a deterministic block automaton that satisfies the orbit property. Note that A' is a k-block automaton and, thus, $L(A')$ is k-block-deterministic by Lemma 13.

Proof. We first define a new operation that is useful for the proof. Given a transition $(p,\alpha\beta,q)$ in a block automaton $A=(Q,\Sigma,\Gamma,\delta,s,F)$, where $\alpha\beta$ is the catenation of two strings α and β, we define the *block expansion* of $(p,\alpha\beta,q)$ to be $A'=(Q\cup\{q'\},\Sigma,\Gamma,\delta\cup\{(p,\alpha,q'),(q',\beta,q)\}\setminus\{(p,\alpha\beta,q)\},s,F)$. Note that block expansion is the reverse operation of state elimination.

Let L be a k-block-deterministic regular language. Then, by definition, there is a k-block automaton $A=(Q,\Sigma,\Gamma,\delta,s,F)$ that satisfies the three conditions in Definition 12. Since A is a position automaton over Γ, we transform A to a traditional FA $A'=(Q',\Sigma,\delta',s,F)$ using block expansion. Since A is a position automaton over Γ and block expansion preserves the structural properties of position automata, A' is also a position automaton over Σ. Therefore, if we can show that A' is deterministic k-lookahead, then L is also deterministic k-lookahead.

We use Definition 6 for showing that A' is deterministic k-lookahead. Since, all states in $Q'\setminus Q$ of A' have a single out-transition, the condition in Definition 6 always holds for these states. Thus, we only need to consider the states from Q in A'. Assume that L is not deterministic k-lookahead and, therefore, there is a state $q\in Q$ in A' such that

$$w_1\cdot\mathbb{F}_{k-1}(p_1)\cap w_1\cdot\mathbb{F}_{k-1}(r_1)\neq\emptyset,$$

where p_1 and r_1 are different target states of q and w_1 is a character over Σ. Let $w\in w_1\cdot\mathbb{F}_{k-1}(p_1)\cap w_1\cdot\mathbb{F}_{k-1}(r_1)$. This implies that there are two sequences of states that spell out w; namely, both $(q,w_1,p_1)\to(p_1,w_2,p_2)\to\cdots\to(p_{k-1},w_k,p_k)$ and $(q,w_1,r_1)\to(r_1,w_2,r_2)\to\cdots\to(r_{k-1},w_k,r_k)$ spell out $w=w_1w_2\cdots w_k$. Let p_i and r_j, for $1\leq i,j\leq k$, be from Q such that $p_{i'}\notin Q$ for $i'<i$ and $r_{j'}\notin Q$ for $j'<j$. Thus, $(q,w_1w_2\cdots w_i,p_i)$ and $(q,w_1w_2\cdots w_j,r_j)$ are in δ of A. There are three possible cases:

(1) $i<j$: $w_1w_2\cdots w_i$ is a prefix of $w_1w_2\cdots w_j$, which violates the second condition in Definition 10.
(2) $i>j$: Symmetry to the first case.
(3) $i=j$: Since $p_i\neq r_j$ in the case, it violates the first condition in Definition 10.

In any of three cases, A is not a deterministic k-block automaton and it contradicts our assumption that A is a deterministic block-automaton. Therefore, there do not exist such states in A' and, thus, A' is a deterministic k-lookahead position automaton. Hence, if L is a k-block-deterministic regular language, then L is also a deterministic k-lookahead regular language.

Next, we show that there is a k-lookahead regular language that is not k-block-deterministic. In Corollary 9, we have proved that $L((a+b)^*a(a+b)^{k-1})$ is deterministic k-lookahead and Giammarresi et al. [9] demonstrated that the same language is not k-block-deterministic. $\square$

5. Conclusions

DTDs are LL(1) grammars for XML documents and the content models for DTDs must be unambiguous. Brüggemann-Klein and Wood [4] observed it and investigated the 1-deterministic regular languages. They showed that 1-deterministic regular languages are a proper subfamily of regular languages. Since 1-deterministic regular languages are the counterpart of LL(1) languages in context-free languages and there is a proper hierarchy in LL(k) languages, we have generalized 1-determinism and examined whether or not there is a similar hierarchy.

In LL(k) languages, we look at k symbols of an input string to decide the next step of the procedure. Similarly, we can assume that we are allowed to look at k symbols of an input string in a given FA. Giammarresi et al. [9] mentioned two possible ways of processing an input string using k symbols as lookahead.

First, we have studied deterministic k-lookahead determinism and showed that there is a deterministic 2-lookahead regular language that is not deterministic 1-lookahead. In addition, we have also demonstrated that there is a hierarchy in k-lookahead determinism.

Second, we have proved that there is a proper hierarchy in k-block-determinism proposed by Giammarresi et al. [9]. Furthermore, we have shown that k-block-deterministic regular languages are a proper subfamily of deterministic k-lookahead regular languages.

Open problem: Corollary 9 only shows that there is a hierarchy in k-lookahead determinism. However, we do not know if the hierarchy is proper or not. Thus, the main open problem is whether or not there is a proper hierarchy in k-lookahead determinism. We conjecture a positive answer because of Theorems 14 and 15.

References

[1] A. Aho, J. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.
[2] T. Bray, J. Paoli, C.S.-M. Queen, E. Maler, F. Yergeau, Extensible Markup Language (XML) 1.0 (fourth edition), August 2006. Available from: <http://www.w3.org/TR/2006/REC-xml-20060816>.
[3] A. Brüggemann-Klein, Regular expressions into finite automata, Theoretical Computer Science 120 (1993) 197–213.
[4] A. Brüggemann-Klein, D. Wood, One-unambiguous regular languages, Information and Computation 140 (1998) 229–253.
[5] J. Brzozowski, E. McCluskey Jr., Signal flow graph techniques for sequential circuit state diagrams, IEEE Transactions on Electronic Computers EC-12 (1963) 67–76.
[6] P. Caron, D. Ziadi, Characterization of Glushkov automata, Theoretical Computer Science 233 (1–2) (2000) 75–90.
[7] S. Eilenberg, Automata, Languages, and Machines, Academic Press, New York, NY, 1974.
[8] D. Giammarresi, R. Montalbano, Deterministic generalized automata, Theoretical Computer Science 215 (1999) 191–208.
[9] D. Giammarresi, R. Montalbano, D. Wood, Block-deterministic regular languages, in: Proceedings of ICTCS'01, Lecture Notes in Computer Science, vol. 2202, 2001, pp. 184–196.
[10] V. Glushkov, On a synthesis algorithm for abstract automata, Ukrainskii Matematicheskii Zhurnal 12 (2) (1960) 147–156. (in Russian)
[11] V. Glushkov, The abstract theory of automata, Russian Mathematical Surveys 16 (1961) 1–53.
[12] Y.-S. Han, D. Wood, The generalization of generalized automata: Expression automata, International Journal of Foundations of Computer Science 16 (3) (2005) 499–510.
[13] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, second ed., Addison-Wesley, Reading, MA, 1979.
[14] ISO 8879: Information processing—Text and office systems—Standard Generalized Markup Language (SGML), International Organization for Standardization, October 1986.
[15] R. McNaughton, H. Yamada, Regular expressions and state graphs for automata, IEEE Transactions on Electronic Computers 9 (1960) 39–47.
[16] D. Wood, Theory of Computation, John Wiley & Sons, Inc., New York, NY, 1987.
[17] D. Wood, Standard Generalized Markup Language: mathematical and philosophical issues, in: J. van Leeuwen (Ed.), Computer Science Today, Lecture Notes in Computer Science, vol. 1000, Springer-Verlag, 1995, pp. 344–365.