Deterministic Regular Languages

Anne Brüggemann-Klein* Derick Wood†

October 1991

Abstract

The ISO standard for Standard Generalized Markup Language (SGML) provides a syntactic meta-language for the definition of textual markup systems. In the standard the right hand sides of productions are called *content models* and they are based on regular expressions. The allowable regular expressions are those that are "unambiguous" as defined by the standard. Unfortunately, the standard's use of the term "unambiguous" does not correspond to the two well known notions, since not all regular languages are denoted by "unambiguous" expressions. Furthermore, the standard's definition of "unambiguous" is somewhat vague. Therefore, we provide a precise definition of "unambiguous expressions" and rename them deterministic regular expressions to avoid any confusion. A regular expression E is *deterministic* if the canonical ϵ-free finite automaton M_E recognizing $\mathrm{L}(E)$ is deterministic. A regular language is *deterministic* if there is a deterministic expression that denotes it. We give a Kleene-like theorem for deterministic regular languages and we characterize them in terms of the structural properties of the minimal deterministic automata recognizing them. The latter result enables us to decide if a given regular expression denotes a deterministic regular language and, if so, to construct an equivalent deterministic expression.

Classification: Automata and formal languages, esp. formal models in document processing

*Institut für Informatik, Universität Freiburg, Rheinstr. 10–12, 7800 Freiburg, Germany

†Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

1 Introduction

Document processing systems like editors, formatters, and retrieval systems deal with many different types of documents, like books, articles, memos, dictionaries, or letters, in addition to user-defined document types or customized versions of "public" types. Recently, the Standard Generalized Markup Language (SGML) [ISO86] has been established as a common platform for the syntactic specification of document types and conforming documents. SGML is an ISO standard and has been endorsed by a number of publishing houses throughout North America and Europe, by the European Community, and by the U.S. Department of Defense.

Document types in SGML are defined by context free grammars that are a mixture of Harrison and Ginsburg's bracketed grammars [GH67] and LaLonde's regular right side grammars [LaL77]. Regular expressions form the right-hand sides of productions, but not all regular expressions are allowed, only those that are "unambiguous" in the sense of Clause 11.2.4.3 of the standard. The intent of the standard is to make it easier for a human to write regular expressions that can be interpreted unambiguously. To achieve this the standard requires each regular expression to be "unambiguous" in the sense that "an element ... that occurs in the document instance must be able to satisfy only one primitive content token without looking ahead in the document instance." In other words, only such regular expressions are valid that permit us to uniquely determine which appearance of a symbol in an expression should match a symbol in an input word without looking beyond that symbol in the input word. This requirement specifies exactly the class of *deterministic expressions* that we investigate here.

An alternative motivation for our study is that the theory of regular languages has become a cornerstone in practical applications involving e.g. specification, pattern matching, and the construction of scanners and parsers. Perhaps the most frequently occurring task is to construct, to a specification in the form of a regular expression, an automaton that can recognize the specified objects. Usually, one first constructs a non-deterministic finite automaton with ϵ-transitions (ϵ-NFA) in time linear in the size of E, eliminates the ϵ-transitions in quadratic time, and finally converts the resulting NFA into a deterministic finite automaton (DFA) [HU79]. The intermediate step can be avoided by directly constructing an ϵ-free automaton [BEGO71, ASU86]. It has been claimed [BS86] that this NFA is the canonical representation because it has a natural connection with the derivatives [Brz64] of the original expression. Since it takes exponential time in the worst case to convert an NFA into a DFA, it is natural to ask for which regular expressions E the canonical NFA M_E is already deterministic. Such expressions are exactly what we have called deterministic above. It can be tested in linear time whether a regular expression is deterministic, and if so, the canonical deterministic automaton can also be constructed in linear time [Bru].

In this paper, we first give a rigorous definition of *deterministic* regular expressions. Then, we investigate the *deterministic* regular languages, i.e. regular languages that can be denoted by a deterministic expression. As we will see, the deterministic regular languages are a proper subclass of the regular languages; for example, for each $n \geq 1$, the expression $(0+1)^*0(0+1)^n$ denotes a regular language that is not deterministic.

First, we state a Kleene-like theorem for the class of deterministic regular languages. Next, we characterize the deterministic regular languages in terms of the minimal deterministic automata that recognize them. To each regular language L, the minimal deterministic finite automaton M_L recognizing L is uniquely determined. We show that deterministic regular languages L can be symbolized by structural properties of M_L. For a state q of M_L, let the *orbit* $\mathcal{O}(q)$ of q denote the strongly connected component of q, i.e. the states of M_L that can be reached from q and vice versa. Some states of $\mathcal{O}(q)$, called *gates*, connect the orbit to the outside world. Now L is deterministic if and only if all orbits of M_L define deterministic regular languages and if for each orbit $\mathcal{O}(q)$, all gates of $\mathcal{O}(q)$ have identical connections to the outside.

Then we show that a deterministic regular language defined by a DFA M with a single orbit is of the form $v\backslash L^*$, where L is deterministic and $v\backslash L^*$ denotes the set of words w such that vw is in L^* (the Brzozowski derivative of L^* by v). Furthermore, a minimal DFA recognizing L can be constructed from M. Together, these results yield an algorithm that decides, given a DFA M, whether its language is deterministic and, if so, constructs an equivalent deterministic expression. The decision algorithm runs in time quadratic in the size of M, but the corresponding deterministic expression can be exponential in the size of M.

To give an example, for each word w, the language $\Sigma^* w \Sigma^*$ of all words over Σ containing w as a subword is a deterministic regular language.

Most proofs in this paper are just sketches. The complete proofs can be found in the full version [BW91].

2 Deterministic regular expressions

The notion of a symbol in a word being satisfied or matched by a symbol in a regular expression has been explained by a number of authors [BEGO71, ASU86, Hen68]. We paraphrase here the description of Hennie [Hen68]. If a word is denoted by an expression, it must be possible to spell out that word by tracing an appropriate "path" through the expression. If we indicate positions in expressions by subscripts, then the word $abba$ is denoted by the expression $(a+b)^*a+\epsilon = (a_1+b_2)^*a_3+\epsilon$ because it corresponds to the path that starts at a_1, visits b_2 twice, and finally arrives at a_3. The set of subscripted symbols in an expression E is denoted by $pos(E)$. Of course, the structure of the expression restricts the positions adjacent symbols of a word can be matched with. For instance, if a symbol in a word is matched by a_3 in $(a_1+b_2)^*a_3+\epsilon$, then no further symbol of the word can be matched with a symbol in the expression. These restrictions have first been formalized by Glushkov [Glu61].

This description suggests viewing a regular expression E as an automaton M_E whose states correspond to the positions or occurrences of symbols in E and whose transitions connect positions that can be consecutive on a path through E. We call M_E the *Glushkov automaton* of E. Figure 1 shows the two Glushkov automata corresponding to the expressions $(a+b)^*a+\epsilon = (a_1+b_2)^*a_3+\epsilon$ and $(b^*a)^* = (b_1^*a_2)^*$. In addition to the states of M_E that correspond to positions in E, the *true* states, there is

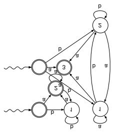


Figure 1 The Glushkov automata for $(a+b)^*a+\epsilon = (a_1+b_2)^*a_3+\epsilon$ and $(b^*a)^* = (b_1^*a_2)^*$.

one unnamed state in the Glushkov automata of Figure 1 which acts as the initial state.

In general, Glushkov automata are non-deterministic, as the example of $(a+b)^*a+\epsilon = (a_1+b_2)^*a_3+\epsilon$ illustrates. After matching an input symbol a with a_1, a further a can either be matched by a_1 or a_3. Thus, there is a transition on a from a_1 to a_1 and to a_3 in the Glushkov automaton. This example leads naturally to a precise definition of what the SGML standard means by a deterministic expression.

Definition 2.1 A regular expression E is *deterministic* if M_E is deterministic, i.e., if M_E is a DFA. A regular language is *deterministic* if there is some deterministic expression that denotes it.

Figure 1 illustrates that $(a+b)^*a+\epsilon$ is not a deterministic expression. Nevertheless, the language denoted by $(a+b)^*a+\epsilon$ is a deterministic regular language, since it is also denoted by $(b^*a)^*$, which is a deterministic expression.

We define M_E inductively, rather than in terms of the formalism introduced by Glushkov [Glu61] that has been used by a number of authors [BEGO71, ASU86, BS86].

M_E has the form $M_E = (Q_E \dot\cup \{q_I\}, \Sigma, \delta_E, q_I, F_E)$, with Q_E comprising the true states corresponding to positions in E, q_I the new initial state, Σ the input alphabet, $\delta_E : (Q_E \cup \{q_I\}) \times \Sigma \longrightarrow 2^{Q_E}$ the transition function, and $F_E \subseteq Q_E \cup \{q_I\}$ the set of final states. To simplify the discussion, we follow the general convention that regular expressions are built from symbols in Σ and the empty string symbol ϵ, but not the empty set symbol $\emptyset$. (Nevertheless, we consider the empty set to be a deterministic regular language.) To *identify* two states in an automaton means to replace them by a new state that has exactly the transitions that both of the old ones had.

Definition 2.2 Given a regular expression E, we define the construction of M_E, illustrated in Figure 2, inductively as follows.

[$E = \epsilon$ or a] M_ϵ and M_a are illustrated in Figure 2.

[$E = F + G$] In M_E the initial states of M_F and M_G are identified. Let

$$Q_E = Q_F \dot\cup Q_G, \quad \text{(disjoint union, possibly after renaming states)}$$

$$F_E = F_F \cup F_G,$$

$$\delta_E(q,a) = \begin{cases} \delta_F(q,a) & \text{if } q \in Q_F \\ \delta_G(q,a) & \text{if } q \in Q_G \\ \delta_F(q_I,a) \cup \delta_G(q_I,a) & \text{if } q = q_I. \end{cases}$$

[$E = FG$] In M_E a copy of the initial state of M_G is identified with each final state of M_F. Let

$$Q_E = Q_F \dot\cup Q_G, \quad \text{(disjoint union, possibly after renaming states)}$$

$$F_E = \begin{cases} F_F \cup (F_G \setminus \{q_I\}) & \text{if } q_I \in F_G, \\ F_G & \text{otherwise,} \end{cases}$$

$$\delta_E(q,a) = \begin{cases} \delta_F(q,a) & \text{if } q \in (Q_F \cup \{q_I\}) \setminus F_F, \\ \delta_F(q,a) \cup \delta_G(q_I,a) & \text{if } q \in F_F, \\ \delta_G(q,a) & \text{if } q \in Q_G. \end{cases}$$

[$E = F^*$] In M_E all transitions from q_I in M_F are added to the final states of M_E. Let

$$Q_E = Q_F, F_E = F_F \cup \{q_I\},$$

$$\delta_E(q,a) = \begin{cases} \delta_F(q,a) \cup \delta_F(q_I,a) & \text{if } q \in F_F, \\ \delta_F(q,a) & \text{otherwise.} \end{cases}$$

Proposition 2.1 *M_E recognizes the language denoted by E.*

Glushkov automata have some peculiar structural properties that are worth investigating.

First, the initial state of M_E has no incoming transitions. This makes the construction correct in the sense of Proposition 2.1. Furthermore, the states directly connected to the initial state correspond exactly to the positions of E that can match the first character of a word of E, and the final states in M_E (besides s_I) correspond exactly to the ones that match the last character of a word of E. Thus, the initial state has transitions to *first positions* in E, and the final states in M_E (besides s_I) are *final positions* of E.

Second, for a subexpression F of E, the structure of M_F is retained in M_E. Each true state of M_F is also a true state of M_E, and all transitions between true states in M_F belong also to M_E. Furthermore, among all positions of F, exactly the final ones are final states of M_E or have transitions in M_E to positions outside of F. These conditions are even fulfilled uniformly, meaning that either all or none of the final positions of F are final in E, and that either all or none have a transition in M_E on an $a \in \Sigma$ to a position y of E outside of F. Hence, the final positions of F are an interface of M_F to the surrounding parts of M_E.

Finally, we will be especially interested in maximal starred subexpressions of E, i.e. subexpressions of the form F^* that are not subexpressions of another starred subexpression G^* of E. For such an F^*, any transitions in M_E between positions of F^* are already transitions in M_{F^*}. In this sense, M_{F^*} is closed within M_E.

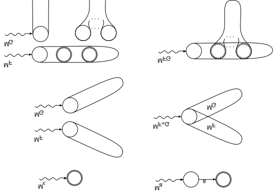


Figure 2 The inductive definition of M_E.

Proposition 2.2 *Given a regular expression E, we can decide if it is deterministic in time linear in the size of E.*

Proof The Glushkov automaton can be constructed from E in such a way that a *new* transition is introduced at each computation step [Bru]. As soon as we encounter a transition that makes the automaton under construction non-deterministic, we stop and report E to be non-deterministic. At this point, only time linear in the size of E has been spent. □

Book et al. [BEGO71] have defined a regular expression E to be unambiguous if the Glushkov automaton M_E is unambiguous, i.e. if for each word w there is at most one computation of M_E that accepts w, or, equivalently, at most one path through E that spells out w. They have shown that each regular language can be denoted by an unambiguous regular expression. A deterministic expression E is an unambiguous one where for each word w the corresponding path through E can be computed incrementally from w with just one symbol of look-ahead. Thus, deterministic regular expressions are related to unambiguous ones in the same way that LL(1) grammars are related to unambiguous context-free grammars. This analogy can be made precise: It is possible to translate a regular expression E in a natural way into an equivalent context-free grammar G_E such that E is deterministic if and only if G_E is LL(1).

3 The characterization theorem

We first state without proof a Kleene-like theorem for deterministic regular languages. We then consider the cyclic structure of M_E that we capture in terms of *orbits*. The structure of the orbits is essentially preserved under minimization, and, hence, orbits turn out to be exactly the right tool for characterizing deterministic regular languages.

We begin by defining three language functions that can also be adapted to apply to regular expressions. These functions are: $\mathit{first}(L)$, the set of symbols that appear as the first symbol of some word in L; $\mathit{last}(L)$, the set of symbols that appear as the last symbol of some word in L; and *followlast*(L), the set of symbols that follow a prefix of some word in L, where the prefix is also a word in L. More formally we have:

Definition 3.1 For $L \subseteq \Sigma^*$, let

$$\mathit{first}(L) = \{a \in \Sigma \mid aw \text{ is in } L \text{ for some word } w\},$$

$$\mathit{last}(L) = \{a \in \Sigma \mid wa \text{ is in } L \text{ for some word } w\},$$

$$\begin{aligned}\mathit{followlast}(L) \;=\; & \{a \in \Sigma \mid vaw \text{ is in } L, \text{ for some word } v \text{ in } L \setminus \{\epsilon\} \\ & \text{and some word } w\}.\end{aligned}$$

Theorem 3.1 *The deterministic regular languages are the smallest class $\mathcal{D}$ of languages that satisfies the following conditions.*

1. *$\emptyset$, ϵ, and $\{a\}$ are in $\mathcal{D}$.*
2. *If $A, B \in \mathcal{D}$ and $\mathit{first}(A) \cap \mathit{first}(B) = \emptyset$, then $A \cup B \in \mathcal{D}$.*
3. *If $A, B \in \mathcal{D}$, $\epsilon \notin A$, and $\mathit{followlast}(A) \cap \mathit{first}(B) = \emptyset$, then $AB \in \mathcal{D}$.*
4. *If $A \in \mathcal{D}$, then $A \setminus \{\epsilon\} \in \mathcal{D}$.*
5. *If $A \in \mathcal{D}$ and $\mathit{followlast}(A) \cap \mathit{first}(A) = \emptyset$, then $A^* \in \mathcal{D}$.*

It is well known that, for each regular language L, the minimum-state deterministic automaton M_L recognizing L is uniquely determined. We argue that, by examining the cyclic structure of M_L, we can decide whether L is deterministic. Furthermore, for a deterministic regular language L, we can construct a deterministic expression for L from M_L.

For each DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizing L, the equivalence class construction [ASU86] results in a DFA $\overline{M} = (\overline{Q}, \Sigma, \overline{\delta}, [q_0], \overline{F})$ which is isomorphic to M_L.[1] For a deterministic expression E denoting a language L, a minimum-state DFA M_L can be constructed directly from M_E via the equivalence class construction. For a non-deterministic expression E, however, the NFA M_E has first to be converted to a DFA via the subset construction [ASU86]. Thus, we are looking for properties of M_E that are preserved

[1]Some minor technicalities are involved here, because the transition functions of M and $\overline{M}$ are only partially defined. The details are in the full version.

under state minimization, but not under subset construction. We start from the structural properties of M_E noted in Section 2.

Definition 3.2 Let $M = (Q, \Sigma, \delta, q_I, F)$ be an NFA. For $q \in Q$, the strongly connected component of q, i.e. the states of M that can be reached from q and from which q can be reached as well, is called the *orbit* of q and denoted by $\mathcal{O}(q)$. We consider the orbit of q to be *trivial* if $\mathcal{O}(q) = \{q\}$ and there are no transitions from q to itself in M.

Definition 3.3 A state q in an NFA $M = (Q, \Sigma, \delta, q_I, F)$ is called a *gate* of its orbit if either q is a final state or there are $q' \in Q \setminus \mathcal{O}(q)$ and $a \in \Sigma$ with $q \xrightarrow{a} q'$. The NFA M has the *orbit property* if each orbit of M is homogeneous with respect to its gates, i.e. if, for all gates q_1 and q_2 with $\mathcal{O}(q_1) = \mathcal{O}(q_2)$, we have:

- q_1 is a final state if and only if q_2 is a final state.
- $q_1 \xrightarrow{a} q$ if and only if $q_2 \xrightarrow{a} q$, for all $q \in Q \setminus \mathcal{O}(q_1) = Q \setminus \mathcal{O}(q_2)$ and for all $a \in \Sigma$.

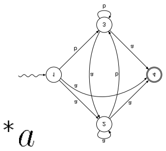
In Figure 3, both automata have three orbits, namely $\{1\}$, $\{2, 3\}$, and $\{4\}$. The singleton orbits are trivial, and each state is a gate of its orbit. The left automaton fulfills the orbit property, the right one does not.

Now we can formulate a necessary condition for a regular language to be deterministic.

Theorem 3.2 *The minimal DFA M_L recognizing a deterministic regular language L has the orbit property.*

This gives us our first example of a regular language that is not deterministic. The rightmost automaton in Figure 3 is the minimal DFA for the language denoted by $(a + b)^*(ac + bd)$, and it does not have the orbit property. Thus, this language cannot be deterministic.

The proof of Theorem 3.2 is in two steps. First, we describe the orbits of the Glushkov NFA M_E constructed in Definition 2.2 and show that M_E has the orbit property for *all* regular expressions. Next, we show that the orbit property is preserved under state minimization. Thus, if a language L is

$(a+b)^*a$ 

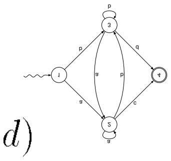
$(a+b)^*(ac+bd)$ 

Figure 3 Two NFAs, one fulfills the orbit property, the other one does not.

symbolized by a *deterministic* regular expression E, the minimal automaton $M_L = \overline{M_E}$ has the orbit property.

Lemma 3.3 *Let E be a regular expression and $x \in pos(E)$. If there is no starred subexpression F^* of E with $x \in pos(F)$, then $\mathcal{O}(x) = \{x\}$ and $\mathcal{O}(x)$ is trivial. On the other hand, if F^* is the maximal starred subexpression of E with $x \in pos(F)$, then $\mathcal{O}(x) = pos(F)$, and $\mathcal{O}(x)$ is not trivial. Finally, the orbit of the initial state q_I is trivial.*

Lemma 3.4 *Let E be a regular expression. Then,*

1. *M_E has the orbit property and*
2. *if F^* is a maximal starred subexpression of E with $pos(F) \neq \emptyset$, then the last positions of F are the gates of the orbit $pos(F)$.*

Figure 4 shows the Glushkov automaton for $a^*(bca^*)^* = a_1(b_2c_3a_4^*)^*$. The gates of orbit $\{2,3,4\}$ are the states 3 and 4, i.e. the last position of the maximal starred subexpression $(bca^*)^* = (b_2c_3a_4^*)^*$.

Now, consider a *deterministic* automaton M and its minimization $\overline{M}$. If states $p_1, \ldots, p_n$ of M form an orbit in M, then the equivalence classes

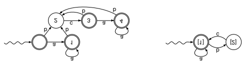


Figure 4 The Glushkov automaton for $a^*(bca^*)^* = a_1(b_2c_3a_4^*)^*$ and its minimization.

$[p_1], \ldots, [p_n]$ belong to the same orbit in $\overline{M}$, which may, however, contain further elements. Figure 4 shows the Glushkov automaton M_E for $E = a^*(bca^*)^* = a_1(b_2c_3a_4^*)^*$ and the minimal automaton $\overline{M_E}$. All states of M_E besides state 2 are equivalent, $\{1\}$ is an orbit of M_E, but $[1]$ does not form a complete orbit in $\overline{M_E}$. Nevertheless, the orbit of $[1]$ in $\overline{M_E}$ is completely generated by another orbit of M_E, namely $\{2, 3, 4\}$. This is a general phenomenon, namely, for each orbit K of $\overline{M}$, there is an orbit C of M that fully generates K, i.e. $K = \{[q] \mid q \in C\}$. C is called a *lift* of K. Now, if a lifted orbit C is homogeneous with respect to its gates, then so is K. Thus, we have the following lemma, which concludes the proof of Theorem 3.2.

Lemma 3.5 *The orbit property is preserved under state minimization.*

The orbit property has gained us a necessary condition for a regular language to be deterministic. Another necessary condition evolves if we examine the orbits themselves in isolation.

Definition 3.4 Let M be a DFA. For $q \in Q_M$, let the *orbit automaton* M_q of q, be the automaton obtained by restricting the state set to $\mathcal{O}(q)$ with initial state q and final states the gates of $\mathcal{O}(q)$ in M. We say the orbit of q is *deterministic* if $\mathrm{L}(M_q)$ is deterministic. A regular language L is said to be an *orbit language* if and only if there is a DFA M with a single orbit that recognizes L.

Theorem 3.6 *For each deterministic language L, the minimal DFA recognizing L has only deterministic orbit languages.*

For the proof, again we first look at the orbit languages of Glushkov automata and then consider state minimization. Let E be a deterministic expression, and let q be a state of M_E with a non-trivial orbit. Then, q is a position of a maximal starred subexpression F^* of E. Let L be the language of F^*, and let v be a word that leads from the initial state to q in M_{F^*}. Since M_{F^*} is closed within M_E, the orbit language of q in M_E is also the orbit language of q in M_{F^*}, which in turn is

$$v\backslash L := \{w \in \Sigma^* \mid vw \in L\}.$$

The language $v\backslash L$ is known as the *derivative* of L by v [Brz64]. Thus, a non-trivial orbit language of M_E is a derivative of a language denoted by a maximal starred subexpression of E.

The proof of the next proposition is in the full paper.

Proposition 3.7 *The derivative of a deterministic regular language is also deterministic.*

As a corollary, we have:

Lemma 3.8 *Let E be a deterministic regular expression. Then, all orbit languages of M_E are deterministic regular languages.*

Again, this property is preserved under minimization, as can be seen from the next lemma.

Lemma 3.9 *Let M be a DFA and $\overline{M}$ be its reduction. Then, for each state of $\overline{M}$ there is an equivalent state q in M such that*

1. *the orbit of q in M is a lift of the orbit of $[q]$ in $\overline{M}$, and*
2. *the orbit languages of q in M and $[q]$ in $\overline{M}$ are identical.*

This concludes the proof of Theorem 3.6.

The necessary conditions for a minimal DFA to recognize a deterministic regular language as given in Theorems 3.2 and 3.6 are also sufficient:

Theorem 3.10 *Let L be a regular language and M be the minimal DFA recognizing L. Then, L is a deterministic regular language if and only if M has the orbit property and all orbits of M are deterministic.*

Proof We show the implication from right to left by induction on the number of orbits of M. Let $M = (Q, \Sigma, \delta, q_I, F)$ have more than one orbit. Furthermore, let $q_1, \ldots, q_n$ be the distinct states outside $\mathcal{O}(q_I)$ that are reachable in one step from a gate of $\mathcal{O}(q_I)$. All gates of $\mathcal{O}(q_I)$ have an a_i-transition to q_i, and no other outgoing transitions from gates of $\mathcal{O}(q_I)$ to the outside exist. The a_i are pairwise distinct and M_{q_I} has no a_i-transition from a final state.

Let M_i be the automaton whose states are the states of M that are reachable from q_i as the initial state. Because M_i has fewer orbits than M, M_i is deterministic. Furthermore,

$$\mathrm{L}(M) = \mathrm{L}(M_{q_I})(a_1\mathrm{L}(M_1) \cup \ldots \cup a_n\mathrm{L}(M_n))$$

or

$$\mathrm{L}(M) = \mathrm{L}(M_{q_I})(a_1\mathrm{L}(M_1) \cup \ldots \cup a_n\mathrm{L}(M_n) \cup \{\epsilon\}),$$

and a deterministic expression for M can be constructed from deterministic expressions for M_{q_I} and $M_1, \ldots, M_n$. □

Theorem 3.10 is the first step of a decision algorithm for deterministic regular languages:

Theorem 3.11 *Given a DFA M, we can decide in time quadratic in the size of M whether the language of M is deterministic. If so, an equivalent deterministic expression can be constructed.*

If a minimal DFA M has the orbit property, then its orbit automata are also minimal. This precludes to apply Theorem 3.10 directly for the orbit automata. On the other hand, we know already that each deterministic orbit language of M has the form $v\backslash\mathrm{L}(F^*)$, where F^* is a deterministic expression. To conclude the proof of Theorem 3.11, we show how a minimal DFA for $\mathrm{L}(F)$ can be constructed from M.

Definition 3.5 A DFA M is *a-consistent*, for $a \in \Sigma$, if there is a state $f_M(a)$ in M such that all final states of M have an a-transition to $f_M(a)$.

Definition 3.6 Let M be a_i-consistent, for $a_i \in \Sigma$, $1 \le i \le n$, $n \ge 1$. The *$a_1, \ldots, a_n$-cut* $M(a_1, \ldots, a_n)$ of M is constructed as follows.

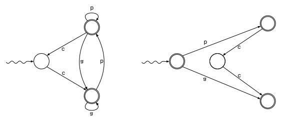


Figure 5 An a, b-consistent DFA and its a, b-cut.

1. A new state q_0 is added to M and it is connected to $f_M(a_i)$ with an a_i-transition, for all i.
2. All transitions with a_i from each final state are removed from M.
3. Finally, q_0 is made initial and final.

Figure 5 gives an example of an a, b-consistent DFA and its a, b-cut.

Theorem 3.12 *Let M be a minimal DFA recognizing a language L. Assume that M consists of a single, non-trivial orbit. Let $a_1, \ldots, a_n$ be the elements of Σ for which M is consistent. Then, L is deterministic if and only if*

1. $n \geq 1$.

2. $\mathrm{L}(M(a_1, \ldots, a_n))$ *is deterministic.*

If L is deterministic and $a_1, \ldots, a_n \in \Sigma$ are chosen as above, we can construct a word $v \in \Sigma^*$ with

$$L = \mathrm{L}(M) = v \backslash \mathrm{L}(M(a_1, \ldots, a_n))^*,$$

and a deterministic expression for language L can be constructed from a deterministic expression for language $\mathrm{L}(M(a_1, \ldots, a_n))$.

In lieu of a proof, we illustrate Theorem 3.12 with two examples. The language recognized by the left automaton in Figure 5 is deterministic, because

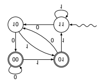


Figure 6 The minimal DFA for $(0+1)^*0(0+1)$.

the a, b-cut is denoted by the deterministic regular expression $a+b(\epsilon+cc)+\epsilon$. One deterministic expression for the whole language is $c(a+b(\epsilon+cc))^*$.

Figure 6 shows the minimal DFA recognizing $(0+1)^*0(0+1)$. It consists of a single orbit with two gates, 00 and 01, but is neither 0- nor 1-consistent. Thus, $(0+1)^*0(0+1)$ does not denote a deterministic language. The same is true for $(0+1)^*0(0+1)^n$ for each $n \geq 1$.

4 Conclusion

In this paper, we have investigated the concept of deterministic regular expressions, which plays a prominent role in the ISO standard SGML. By characterizing the regular languages denoted by deterministic regular expressions, we have contributed to the theoretical foundation of SGML systems supporting the development of document type definitions.

References

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools. Addison-Wesley Series in Computer Science*, Addison-Wesley, Reading, Massachusetts, 1986.

[BEGO71] Ronald Book, Shimon Even, Sheila Greibach, and Gene Ott. Ambiguity in graphs and expressions. *IEEE Transactions on Computers*, C-20(2):149–153, February 1971.

[Bru] Anne Brüggemann-Klein. Regular expressions into finite automata. To appear in the conference proceedings of Latin '92.

[Brz64] Janusz A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, October 1964.

[BS86] Gerard Berry and Ravi Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48:117–126, 1986.

[BW91] Anne Brüggemann-Klein and Derick Wood. On the expressive power of SGML document grammars. In preparation, 1991.

[GH67] Seymour Ginsburg and Michael M. Harrison. Bracketed context-free languages. *Journal of Computer and System Sciences*, 1(1):1–23, March 1967.

[Glu61] V.M. Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16:1–53, 1961.

[Hen68] Frederick C. Hennie. *Finite-State Models for Logical Machines.* John Wiley, New York, 1968.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation. Addison-Wesley Series in Computer Science*, Addison-Wesley, Reading, Massachusetts, 1979.

[ISO86] ISO 8879. Information processing—text and office systems—standard generalized markup language (SGML). October 1986. International Organization for Standardization.

[LaL77] Wilf R. LaLonde. Regular right part grammars and their parsers. *Communications of the ACM*, 20(10):731–741, October 1977.