AMBIGUITY IN FINITE AUTOMATA

by

Erik Meineche Schmidt

DAIMI PB-82

September 1977

Ambiguity in Finite Automata [†]

by

Erik Meineche Schmidt

---

† Most of this work was done while the author was a graduate student at Cornell University. It will appear as part of his Ph.D. dissertation, which will be submitted to Cornell in the fall, 1977.

CONTENTS

1. INTRODUCTION

Ambiguity has been studied extensively in connection with context-free languages, where the existence of (inherently) ambiguous languages, and the undecidability of most properties involving ambiguity are among the most important results ([4], [9]). Recently Valiant [11] and Schmidt & Szymanski [10] examined ambiguity from the point of view of succinctness of descriptions of context-free languages. They showed that there is no recursive function bounding the amount of compactness gained when using unambiguous pushdown automata (pda's) rather than deterministic pda's in the description of deterministic context-free languages, and similarly for ambiguous pda's and unambiguous ones.

In this paper we consider similar questions for finite automata. Since every regular set is accepted by a deterministic finite automaton, there are no (inherently) ambiguous regular languages. Succinctness-wise, however, the behaviour of deterministic, unambiguous and ambiguous finite automata is similar to that of the pda's provided the term "nonrecursive" is replaced by "nonpolynomial". Since every nondeterministic finite automaton with n states has a deterministic equivalent with no more than 2^n states, the gain in compactness in descriptions between the different types of finite automata is always bounded by an exponential, so the best we can hope for is nonpolynomial succinctness.

It was shown by Meyer & Fisher in [7] that this type of succinctness exists between deterministic and ambiguous finite automata, and in this paper we show that the same property holds for deterministic and unambiguous automata, as well as for unambiguous and ambiguous machines.

We also examine the role of ambiguity in connection with the complexity of problems involving regular expressions. We specifically consider the problem of deciding whether the complement of the language generated by a regular expression is nonempty. This problem is complete for polynomial space (PSPACE) when arbitrary expressions are considered, but since the proof involves ambiguous regular expressions it is natural to ask how difficult the problem is when only unambiguous expressions are considered. We show that in this case the problem is in NP, and hence that the absence of ambiguity makes it easier (unless, of course, PSPACE happens to be equal to NP). Finally we prove that deciding whether a regular expression is ambiguous is complete for nondeterministic logarithmic space (NLOGSPACE).

The paper is divided into four sections of which this is the first. Section 2 contains the proof of the nonpolynomial succinctness between ambiguous and unambiguous finite automata and section 3 the result that nonemptiness of complement for unambiguous finite automata is in NP. Finally, it is shown in section 4 that the ambiguity problem for regular expressions is complete for NLOGSPACE.

2. SUCCINCTNESS

In this section we prove that unambiguous finite automata,in terms of succinctness,lie between deterministic and nondeterministic machines.

The reader is assumed to be familiar with standard concepts from automata theory and complexity theory, and is referred to [2] and [1] for definitions not presented in the paper.

We shall use notation and terminology from [2]. Hence, a nondeterministic finite automaton, NFA, is a system $M = (Q, \Sigma, \delta, q_0, F)$ consisting of states, input alphabet, transition function (from $Q \times \Sigma$ to subsets of Q), startstate and final states. M is in configuration $(q, x) \in Q \times \Sigma^*$ if it is in state q and x is the part of the input that remains to be read. $\vdash$ is the usual "transition relation" between configurations and $\overset{*}{\vdash}$ is its reflexive and transitive closure. The language accepted by M is the set $T(M) = \{x \in \Sigma^* \mid \exists\, q \in F : (q_0, x) \overset{*}{\vdash} (q, \lambda)\}$+. M is said to be an unambiguous finite automaton, UFA, if no word is accepted in more than one way, i.e. for no $x = a_1 a_2 \dots a_n$ is there more than one sequence of states $q_0, q_1, \dots, q_n$ such that $q_n \in F$ and $(q_{i-1}, a_i \dots a_n) \vdash (q_i, a_{i+1} \dots a_n)$ for $1 \le i \le n$. M is a deterministic finite automaton, DFA, if for all $q \in Q$ and $a \in \Sigma$ $\delta(q, a)$ contains at most one element.

We now introduce some notation which will make it easier to talk about succinctness. Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be classes of descriptors (such as machines or grammars) and assume that the family of languages generated by $\mathfrak{M}_1$ is equal to the family generated by $\mathfrak{M}_2$. Furthermore let <u>size</u>$_1$ and <u>size</u>$_2$ be functions mapping $\mathfrak{M}_1$ and $\mathfrak{M}_2$ to the nonnegative integers. Then, by $\mathfrak{M}_1 \xrightarrow{f(n)} \mathfrak{M}_2$ we denote <u>f(n)-succinctness</u> between $\mathfrak{M}_1$ and $\mathfrak{M}_2$, meaning that there are languages defined by small $\mathfrak{M}_1$-descriptors which require large $\mathfrak{M}_2$-descriptors. The difference between small and large is determined by $f(n)$. Formally $\mathfrak{M}_1 \xrightarrow{f(n)} \mathfrak{M}_2$ means that there is a family of languages $\{L(n) \mid n \in N\}$ defined by elements $\{m_1(n) \mid n \in N\}$ of $\mathfrak{M}_1$ such that for any family of $\mathfrak{M}_2$-descriptors $\{m_2(n) \mid n \in N\}$ defining the same languages, $\underline{\text{size}}_2(m_2(n)) = O(f(\underline{\text{size}}_1(m_1(n))))$.

+ λ is the empty word.

Theorem 1 The classes of deterministic, unambiguous and nondeterministic finite automata satisfy the following succinctness diagram

$$\begin{tikzcd}
 & \mathrm{UFA} \arrow[dl, "2^{c_1\sqrt{n}}"'] & \\
\mathrm{DFA} & & \mathrm{NFA} \arrow[ul, "2^{c_2\sqrt{n}}"'] \arrow[ll, "2^n"]
\end{tikzcd}$$

where c_1 and c_2 are positive constants.

Proof a) The 2^n-succinctness between NFA and DFA was proved in [7].

b) The $2^{c_1\sqrt{n}}$-succinctness between UFA and DFA is proved by considering the family of languages $\{L_1(n) \mid n \in N\}$ where

$$L_1(n) = \{x \# a^m \mid x \in \{0,1\}^n,\ 1 \le m \le n, \text{the m'th bit of } x \text{ is } 1\}$$

Since a deterministic automaton must distinguish between all x-prefixes of the words $x \# a^m$, it is clear that a deterministic machine accepting $L_1(n)$ must have at least 2^n states. An unambiguous machine, on the other hand, can recognize $L_1(n)$ by guessing which bit is the m'th, skipping the input until it sees #, and then check that the guess was correct. It is easy to see that such a machine needs no more than $O(n^2)$ states.

c) To get the NFA $\xrightarrow{2^{c_2\sqrt{n}}}$ UFA result we consider the languages $\{L_2(n) \mid n \in N\}$ where

$$L_2(n) = \{x \# y \mid x, y \in \{0,1\}^n,\ x \neq y\}$$

Using again the guess-and-check technique it is straightforward to construct a nondeterministic automaton with no more than $O(n^2)$ states accepting $L_2(n)$, but since the prefix x and the postfix y can differ in many ways, the obvious machine is ambiguous. In the following lemma we show that ambiguity is unavoidable unless the machine is allowed to have 2^n states. This proves the result. □

What the next lemma says is essentially that a nondeterministic (but unambiguous) machine accepting $L_2(n)$ has to distinguish between all prefixes of the form $x\#$.

Lemma 2 Any unmabiguous finite automaton recognizing the language $L_2(n) = \{x\#y \mid x, y \in \{0,1\}^n,\ x \neq y\}$ has at least 2^n states.

Proof Assume that $M = (Q, \Sigma, \delta, q_0, F)$ is such an automaton. We show that at least 2^n states are reachable from the startstate via prefixes of the form $x\#$.

Let $x \in \{0,1\}^n$ be arbitrary and assume that $K_x = \{q_1, \ldots, q_{k_x}\}$ is the set of states reachable from q_0 via $x\#$. Define for each i $(1 \leq i \leq k_k)$ the set $A_x^i = \{y \in \{0,1\}^n \mid \exists q \in F : (q_i, y) \vdash^* (q, \lambda)\}$ consisting of the words in $\{0,1\}^n$ which lead from the state q_i (in K_x) to acceptance. Consider, for x varying over $\{0,1\}^n$, the total collection of these sets $A = \{\{A_x^i\}_{i=1}^{k_x}\}_{x \in \{0,1\}^n}$ and let $B_1, B_2, \ldots, B_m$ be a listing of the sets in A without repetitions.

Let K be the set of all states reachable from q_0 via some prefix of the form $x\#$ and consider the function which maps each state q in K to the set of words in $\{0,1\}^n$ which lead from q to a final state. It is easy to see that this function maps K onto the listing $B_1, \ldots, B_m$. Hence in order to show that there are at least 2^n states in K it is sufficient to show that there are at least that many B_i's.

We do this by interpreting subsets of $\{0,1\}^n$ as elements of the 2^n-dimensional vector-space over the field of characteristic 2. Assume that $x_1, x_2, \ldots, x_{2^n}$ is an enumeration of $\{0,1\}^n$. With each $C \subseteq \{0,1\}^n$ we associate the vector $\vec{C} = (c_1, c_2, \ldots, c_{2^n})$ where for $1 \leq j \leq 2^n$ $c_j = 1$ iff $x_j \in C$.

Now consider for each x the set $A_x = \mathop{}_{j=1}^{k_x} A_x^i$. We claim that since the automaton is unambiguous, the sets $A_x^1, \ldots, A_x^{k_x}$ are mutually disjoint. This follows since if there is i, j and y such that $y \in A_x^i \cap A_x^j$ then we have

$$(q_0, x\#y) \vdash^* (q_i, y) \vdash^* (p_1, \lambda)$$

and

$$(q_0, x\#y) \vdash^* (q_j, y) \vdash^* (p_2, \lambda)$$

where both p_1 and p_2 are in F. But then $x\#y$ is accepted in two different ways, which is a contradiction. Now since $A_x^1, \ldots, A_x^{k_x}$ are disjoint and furthermore all occur among the B_i's, the <u>vector</u> $\vec{A}_x$ can be written as a linear combination of the <u>vectors</u> $\vec{B}_i$ i.e.

$$(*) \qquad \vec{A}_x = \sum_{j=1}^{m} t_j \vec{B}_j \qquad t_j \in \{0,1\} \text{ for } 1 \leq j \leq m$$

Let us assume that x is the i'th element in the enumeration of $\{0,1\}^n$, i.e. $x = x_i$. Since all words of the form $x_i \# x_j$ are in $L_2(n)$, unless $i = j$, it follows that A_{x_i} is equal to $\{0,1\}^n - \{x_i\}$, thus $\vec{A}_{x_i} = (1, 1, \ldots, 1, 0, 1, \ldots, 1)$, where 0 is the i'th coordinate. But it is easy to see that the vectors $\{A_{x_i}\}_{i=1}^{2^n}$ are linearly independent and since (*) shows that they all can be written as linear combinations of $\vec{B}_1, \ldots, \vec{B}_m$ it follows that $m \geq 2^n$. Hence there are at least 2^n B_i's, consequently also at least 2^n states in K, and the lemma is proved. □

3. COMPLEXITY

Here we use an argument similar to the proof of lemma 2 to show that ambiguity also plays a role in connection with the complexity of finite automata.

Consider the problem of deciding whether a finite automaton accepts all words over its input alphabet. This problem is complete for PSPACE when arbitrary nondeterministic automata are considered [8], and it has a polynomial time algorithm in case the automata are deterministic, (actually one which runs in nondeterministic logarithmic space [5]). The reason the problem is so difficult in the general case is that the length of the shortest string rejected by a nondeterministic finite automaton can be exponential in the size of the machine. The following theorem shows that ambiguity is essential in this connection.

Theorem 3 Let $M = (Q, \Sigma, \delta, q_0, F)$ can be an unambiguous finite automaton with m states. The shortest word not accepted by M is no longer than $m + 1$.

Proof Let $w = a_1 \dots a_n$ be one of the shortest words not accepted by M and let $K_0, K_1, K_2, \dots, K_n$ be the set of states reachable from q_0 via λ, $a_1, a_1 a_2, \dots, a_1 \dots a_{n-1}$ and $a_1 \dots a_n$.

We will show that the set of states $K = \bigcup_{i=0}^{n-1} K_i$ contains at least n-1 elements. As in lemma 2 we do this by associating with each state in K a set of words which gets interpreted as a vector in an appropriate vector-space in such a way that n-1 of the vectors become linearly independent.

Here the proper choice of words is the set of suffixes of w. Let for $1 \leq i \leq n$ x_i denote $a_i a_{i+1} \dots a_n$ and let $X = \{x_1, x_2, \dots, x_n\}$. Again we associate subsets of X with the states in K_i. Assume $K_i = \{q_1, \dots, q_{k_i}\}$ and let for $1 \leq j \leq k_i$ A_i^j be the set of words in X leading from q_j to acceptance, i.e. $A_i^j = \{x \in X \mid \exists\, q \in F : (q_j, x) \overset{*}{\vdash} (q, \lambda)\}$. Now consider the union $A_i = \bigcup_{j=1}^{k_i} A_i^j$ of these sets. Since $a_1 \dots a_{i-1}$ leads to the states in K_i and since A_i consists of the words leading from there to acceptance, $x_i = a_i \dots a_n$ cannot be in A_i because then the automaton would accept

$w = a_1 \dots a_{i-1} a_i \dots a_n$. Furthermore since w is the shortest string rejected by M all the words $a_1 \dots a_{i-1} x_{i+1}, \dots, a_1 \dots a_{i-1} x_n$, which are shorter than w, are accepted. But that means that $x_{i+1}, \dots, x_n$ all are in A_i.

Now consider the n-dimensional vector-space over the field of characteristic 2 and interpret subsets of X as vectors in the same way as in lemma 2, i.e. if $C \subseteq X$ then $\vec{C} = (c_1, \dots, c_n)$ where for $1 \le j \le n$ $c_j = 1$ iff $x_j \in C$. By the above argument we know that $x_i \notin A_i$ and that $x_j \in A_i$ for $i < j \le n$. Hence the vector $\vec{A}_i$ is of the form

$$\vec{A}_i = (b_i^1, \dots, b_i^{i-1}, 0, 1, \dots, 1)$$

where the first i-1 coordinates are determined as follows:

$$b_i^j = \begin{cases} 1 & \text{if } a_1 \dots a_{i-1} x_j \text{ is accepted by M} \\ 0 & \text{otherwise.} \end{cases}$$

Let $B_1, B_2, \dots, B_k$ be a listing, without repetitions, of the sets appearing in the total collection of sets $\{\{A_i^j\}_{j=1}^{k_i}\}_{i=1}^{n}$. Again, as in lemma 2, since the automaton is unambiguous each A_i is a disjoint union of $A_i^1, \dots, A_i^{k_i}$, hence $\vec{A}_i$ can be written as a linear combination of the vectors $\vec{B}_1, \dots, \vec{B}_k$. Also the number of states in K is greater than or equal to the number of sets in the listing $B_1, B_2, \dots, B_k$, so all that remains is to show that sufficiently many of the vectors $\vec{A}_1, \dots, \vec{A}_n$ are linearly independent.

Consider the matrix

$$A = \begin{pmatrix} 0 & 1 & \dots\dots\dots\dots & & 1 \\ b_2^1 & 0 & 1 & \dots\dots\dots & 1 \\ b_3^1 & b_3^2 & 0 & 1 \ \dots\dots & 1 \\ \vdots & & & & \\ b_i^1 & \dots\dots\dots & 0 & 1 \ \dots & 1 \\ \vdots & & & & \\ b_n^1 & \dots\dots\dots & & b_n^{n-1} & 0 \end{pmatrix}$$

whose i'th row is $\vec{A}_i$. If we disregard the first column and the last row we get the $(n-1) \times (n-1)$ submatrix

$$A' = \left\{ \begin{array}{llll} 1 & 1 \ldots\ldots\ldots\ldots & & 1 \\ 0 & 1 \ldots\ldots\ldots\ldots & & 1 \\ b_3^2 & 0 \;\; 1 \ldots\ldots\ldots\ldots & & 1 \\ \vdots & & & \\ b_{n-1}^2 & \ldots\ldots\ldots & b_{n-1}^{n-1} & 1 \end{array} \right\}$$

which is easily seen to reduce, by Gaussian elimination over the field of characteristic 2, to the $(n-1) \times (n-1)$ unit matrix. From this we conclude that the matrix A has rank n-1 and hence that n-1 of the vectors $\vec{A}_1, \ldots, \vec{A}_n$ are linearly independent. Now the same argument as in lemma 2 applies, and we conclude that the automaton has at least n-1 states. □

Theorem 3 has the immediate corollary that deciding whether an automaton accepts all words over its input alphabet probably is easier for unambiguous machines than for ambiguous ones.

Corollary 4 There is a nondeterministic polynomial time algorithm for deciding whether an unambiguous finite automaton does not accept all words over its input alphabet.

Proof The method is exactly the same as the one used to show that the problem is in polynomial space for arbitrary finite automata (see lemma 10.3 in [1]). But since, by Theorem 3, the shortest string not accepted by an unambiguous machine is no longer than the number of states in the machine, the algorithm stops after a polynomial amount of time in this case. □

4. REGULAR EXPRESSIONS

In this section we consider succinctness- and complexity questions for the class of regular expressions which we define in the usual way.
Let Σ be an alphabet. The regular expressions over Σ is the smallest set, Rexp_Σ, which satisfies the following two requirements

1) $\emptyset$, e and all elements in Σ are in Rexp_Σ

2) if R_1 and R_2 are in Rexp_Σ then so are (R_1+R_2), $(R_1 \cdot R_2)$ and (R_1^*).

The language denoted by an expression R, L(R), is also defined as usual. The class of regular expressions, Rexp, is the union of the sets Rexp_Σ over all alphabets Σ not containing $\emptyset$, e, (,), $\cdot$, $*$ and $+$. As size of a regular expression we take its length.

The following theorem summarizes the succinctness relations between regular expressions and finite automata. 2) was proved by Ehrenfeucht and Zeiger in [3].

Theorem 5

1) $\mathrm{Rexp} \xrightarrow{n} \mathrm{NFA}$

2) $\mathrm{DFA} \xrightarrow{2^n} \mathrm{Rexp}$

3) $\mathrm{Rexp} \xrightarrow{2^{c\sqrt{n}}} \mathrm{DFA}$ for some $c > 0$.

Proof 1) This is obvious from the usual construction ([2]) of a non-deterministic finite automaton accepting the language generated by a regular expression.

2) See [3].

3) The language $L_2(n)$ in Lemma 2 is generated by the regular expression

$$\sum_{i=0}^{n-1} (\{0,1\}^i 0\{0,1\}^{n-i-1}\#\{0,1\}^i 1\{0,1\}^{n-i-1}+\{0,1\}^i 1\{0,1\}^{n-i-1}\#\{0,1\}^i 0\{0,1\}^{n-i-1})$$

which is of length $O(n^2)$. It was shown there that any unambiguous, hence also any deterministic, finite automaton recognizing $L_2(n)$ must have at least 2^n states. $\square$

Next we turn to ambiguity. A regular expression is said to be ambiguous if there is a word which is generated in more than one way. The motivation for looking at ambiguity in regular expressions is the use of socalled Extended Context-Free Grammars [6], as a means for specifying the syntax of programming languages. In these grammars one is allowed to use regular expressions over terminals and nonterminals as righthandsides of productions. Since there are (normally) semantic actions associated with the process of recognizing the language generated by such grammars, it is important that the expressions are unambiguous, and it might be of interest to know how hard it is to determine if a regular expression is ambiguous, and if so, how big the smallest equivalent unambiguous expression is. Let URexp denote the class of unambiguous regular expressions. The following result is an immediate corollary of Lemma 2 and Theorem 5, 1).

<u>Corollary 6</u>

$$\text{Rexp} \xrightarrow{2^{c\sqrt{n}}} \text{URexp} \quad \text{for some } c > 0. \qquad \square$$

Before we can show how hard it is to determine ambiguity we need the following notation.

(N)LOGSPACE denotes the class of sets accepted by (nondeterministic) log-space bounded Turing Machines (TM's) with a read only input tape and one work tape. A set is complete for NLOGSPACE if it is in NLOGSPACE and every other set in NLOGSPACE is log-space reducible to it, see [5].

Let M be a log-space bounded TM with work tape alphabet Γ and state set Q, and let x be an input string. A <u>configuration of M on x</u> is a pair (i, z) where $1 \leq i \leq |x|$ and $z \in \Gamma^*Q\Gamma^*$. i is the position of the input head and z represents in the usual way the content of the work tape, the machine state and the position of the worktape head. A <u>computation of M on x</u> is a sequence $(1, z_1), (i_2, z_2), \ldots, (i_m, z_m)$ where (i_{j+1}, z_{j+1}) follows from (i_j, z_j) by application of M's transition function, z_1 is the start configuration for the worktape and the state in z_m is a final state.

Theorem 7   The set

$$A = \{ R \in \text{Rexp} \mid R \text{ is ambiguous} \}$$

is complete for NLOGSPACE.

Proof   First we show that A is in NLOGSPACE. Let R be a regular expression of length n (over the alphabet Σ). We show that if there is a word which is generated in more than one way, then there is one which is no longer than $2n^2$. We know that there is a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ with no more than n states accepting the language generated by R and also that if a word is generated in two different ways by R then it is accepted in two different ways by M. Now, let w be the shortest word accepted in two different ways by M and assume that $|w| \geq 2n^2+1$. Let the state-sequences corresponding to the two accepting computations be $s_1 = q_0 q_1 \ldots q_m$ and $s_2 = p_0 p_1 \ldots p_m$ where $q_m, p_m \in F$, $p_0 = q_0$ and $m = |w|$.

Since $m \geq 2n^2+1$, some pair of states occurs at least three times in the sequence $(q_0, p_0)(q_1, p_1), \ldots, (q_m, p_m)$. Assume that $(q_i, p_i) = (q_j, p_j) = (q_k, p_k)$ for some i, j and k with $0 \leq i < j < k \leq m$. Since the sequences s_1 and s_2 are different either $q_0 \ldots q_i q_{j+1} \ldots q_m$ is different from $p_0 \ldots p_i p_{j+1} \ldots p_m$ or $q_0 \ldots q_j q_{k+1} \ldots q_m$ is different from $p_0 \ldots p_j p_{k+1} \ldots p_m$. In either case we can, by cutting out the proper piece of the input, obtain a shorter word which is also accepted in two different ways, which contradict the assumption that the shortest word is longer than $2n^2$. This shows that a log-space bounded Turing Machine is powerful enough to guess (symbol by symbol) a word generated in more than one way if there is one.

To see that the machine is also capable of checking that such a word is indeed generated in two ways by the expression, we first note that this would be straightforward if the input had been the automaton M, rather than the expression R. Then we would just guess two sequences of states and accept in case they both end in final states and are different. Now, since states in M correspond to positions in R, we construct the nondeterministic Turing Machine such that it guesses two sequences of positions, each corresponding to a parse of the word. The reason this is not difficult is that R is assumed to be syntactically correct, hence we can always, by counting parentheses, find the subexpression beginning at or ending at a certain position. The details of the construction are left to the reader, but it should be clear that the algorithm works correctly and runs in nondeterministic log-space.

Next we show that the set A is hard for NLOGSPACE. Given a nondeterministic log-space bounded Turing Machine M and an input x we construct two unambiguous regular expressions $\overline{R}_x$ and $\hat{R}_x$ such that $\overline{R}_x \cup \hat{R}_x$ is ambiguous if and only if M accepts x. The technique is the same as in [10], $\overline{R}_x$ represents all the odd-even pairs of consecutive configurations of M on x and $\hat{R}_x$ all the even-odd pairs. Then $\overline{R}_x \cap \hat{R}_x$ is nonempty exactly in case there is a computation of M on x.

We assume without loss of generality that the machine M always performs an odd number of steps. Consider the following two sets of words

$\overline{W}_x = \{ \# i_1 \$ z_1 \# i_2 \$ z_2 \# i_3 \$ z_3 \# \ldots \# i_{2n} \$ z_{2n} \# \mid$

a) $n \geq 1$

b) (i_1, z_1) is the starting configuration of M on x

c) (i_{2j}, z_{2j}) follows from (i_{2j-1}, z_{2j-1}) by M's transition function (for $1 \leq j \leq n$)$\}$

$\hat{W}_x = \{ \# i_1 \$ z_1 \# i_2 \$ z_2 \# i_3 \$ z_3 \# \ldots \# i_{2n} \$ z_{2n} \# \mid$

a) $n \geq 1$

b) (i_{2n}, z_{2n}) is an accepting configuration of M on x

c) (i_{2j+1}, z_{2j+1}) follows from (i_{2j}, z_{2j}) by M's transition function (for $1 \leq j < n$)$\}$

$\overline{W}_x$ and $\hat{W}_x$ are generated by the following two unambiguous regular expressions

$$\overline{R}_x = \# 1 \$ z_1 \# \cdot N \cdot (P^*)$$
$$\hat{R}_x = B \cdot (P^*) \cdot F$$

where

a) z_1 is the starting worktape configuration

b) N is the sum of all expressions of the form i \$ z # such that (i, z) follows from $(1, z_1)$

c) P is the sum of all expressions of the form i \$ z # i' \$ z' # such that (i', z') follows from (i, z)

d) B is the sum of all expressions of the form # i \$ z #

e) F is the sum of all expressions of the form i \$ z # where the state in z is a final state.

It is clear that $\overline{R}_x$ and $\hat{R}_x$ are unambiguous and also that the expressions N, P, B, F - and therefore also $\overline{R}_x$ and $\hat{R}_x$ - can be computed from M and x by a deterministic log-space bounded Turing Machine. But $\overline{R}_x \cap \hat{R}_x$ is nonempty if and only if it contains a word representing a (halting) computation of M on x. Hence the expression $R_x = \overline{R}_x + \hat{R}_x$ is ambiguous if and only if M accepts x. This shows that the set A is hard for NLOGSPACE. □

Acknowledgements

I wish to thank Steve Fortune for pointing out the usefulness of vector spaces in the proof of Lemma 2 and Tom Szymanski for bringing the problem of complexity of unambiguous regular expressions to my attention.

References

[1] Aho, A.V., J.E. Hopcroft and J.D. Ullman [1974]. The design and analysis of computer algorithms, Addison-Wesley, Reading, Massachusetts.

[2] Aho, A.V. and J.D. Ullman [1972]. The Theory of Parsing, Translation and Compiling, Volume 1: Parsing, Prentice-Hall, Englewood Cliffs, N.J.

[3] Ehrenfeucht, A and P. Zeiger [1976]. "Complexity Measures for Regular Expressions", JCSS 12: 2, 134-146.

[4] Hopcroft, J.E. and J.D. Ullman [1969]. Formal Languages and Their Relation to Automata, Addison-Wesley, Reading, Massachusetts.

[5] Jones, N.D [1975). "Space-Bounded Reducibility among Combinatorial Problems", JCSS 11: 1, 68-85.

[6] Madsen, O.L. and B.B. Kristensen [1976]. "LR-Parsing of Extended Context Free Grammars", Acta Informatica 7: 1, 61-73.

[7] Meyer, A.R. and M.J. Fischer [1971]. "Economy of Description by Automata, Grammars and Formal Systems", Conference Record, IEEE 12th Annual Symposium on Switching and Automata Theory, 188-190.

[8] Meyer, A.R. and L. Stockmeyer [1972]. "The equivalence problem for regular expressions with squaring requires exponential space", Conference Record, IEEE 13th Annual Symposium on Switching and Automata Theory, 125-129.

[9] Reedy, A. and W.J. Savitch [1975]. "The Turing degree of the inherent ambiguity problem for context-free languages", TCS 1: 1, 77-91.

[10] Schmidt, E.M. and T.G. Szymanski [1977]. "Succinctness of Descriptions of Unambiguous Context-free Languages", To appear in SIAM J. Computing.

[11] Valiant, L.G. [1976]. "A Note on the Succinctness of Descriptions of Deterministic Languages", Information and Control, 32: 2, 139-145.