COMPLEXITY OF SOME PROBLEMS CONCERNING L SYSTEMS
(Preliminary Report)

by

Neil D. Jones
and
Sven Skyum

DAIMI PB-67
November 1976
1. INTRODUCTION

Recently, considerable interest has been shown in questions concerning the complexity of the membership problem for various types of L systems. Van Leeuwen showed in [11] that there are ET0L systems G such that L(G) is complete for \( \text{HP} \) (nondeterministic polynomial time). Opatrný and Culik showed in [7] that E0L membership (for fixed grammars) may be decided deterministically in time \( n^4 \), and Sudborough gave a \((\log n)^2\) space algorithm for the same problem in [10], based on a construction by van Leeuwen [12]. Sudborough also gave a deterministic \( \log n \) space algorithm for ED0L membership in [10], and showed in [9] that some linear languages (and hence some E0L and deterministic ET0L languages) are complete for nondeterministic log space. In a companion paper [4], we have shown that each deterministic ET0L language can be recognized nondeterministically in \( \log n \) space, and therefore deterministically in polynomial time.

In this paper we study the complexity of the emptiness and finiteness questions for each of these classes (ET0L, E0L, and their deterministic counterparts), as well as the general membership problem.

Let \( \overrightarrow{\sigma} \) be a linearly encoded form of an ET0L system over a fixed alphabet independent of G. (E.g. represent symbols \( v_1, v_2, \ldots, v_m \) in the form \( \uparrow i \) where \( i \) is the binary representation of \( i, 1 \leq i \leq m \).) The problems we discuss may all be represented in terms of membership in the following sets. C denotes any of the system classes just mentioned.

1. \( \text{NONEMPTY}^C = \{ \overrightarrow{\sigma} \mid G \text{ is in } C \text{ and } L(G) \neq \emptyset \} \)

2. \( \text{INFINITE}^C = \{ \overrightarrow{\sigma} \mid G \text{ is in } C \text{ and } L(G) \text{ is infinite} \} \)

3. \( \text{MEMBER}^C = \{ \overrightarrow{\sigma}, x \mid G \text{ is in } C \text{ and } x \text{ is in } L(G) \} \)

4. \( L(G) \) for a fixed grammar \( G \) in \( C \)
The work referenced above establishes upper and lower complexity bounds on problems of type 4 (except for a lower bound on deterministic E0L membership). We shall outline a series of constructions which suffice to establish both upper and lower bounds on the remaining problems (in most cases rather tight). As we shall see, the complexity of the general membership problem (in which the input is the system as well as the terminal string) can be much higher than that of determining whether $x$ is in $L(G)$ for some fixed $G$. In the most extreme case, if $C$ is the class of deterministic ET0L grammars, membership for fixed systems may be determined in $\log n$ space, while the general problem requires essentially linear space (both by non-deterministic algorithms).
The results may be presented in the form of a table as follows. For the sake of comparison we have included the context-free and context-sensitive classes as well. In the system class names, D indicates "deterministic", and P indicates "propagating" (i.e. the absence of productions with the empty string on the right side).

<table>
<thead>
<tr>
<th>GRAMMAR CLASS</th>
<th>MEMBER (FIXED O)</th>
<th>MEMBER (GENERAL)</th>
<th>NONEMPTY</th>
<th>INFINITE</th>
<th>BOUNDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXT SENSITIVE</td>
<td>NSPACE(n)</td>
<td>NSPACE($n \log n$)</td>
<td>UNDECIDABLE</td>
<td>UNDECIDABLE</td>
<td>UPPER</td>
</tr>
<tr>
<td></td>
<td>NSPACE(n)</td>
<td></td>
<td></td>
<td></td>
<td>LOWER</td>
</tr>
<tr>
<td>ETOL, EPTOL</td>
<td>hP</td>
<td>NSPACE($n \log n$)</td>
<td>NSPACE(n)</td>
<td>NSPACE(n)</td>
<td>UPPER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NSPACE($n^{1-\varepsilon}$)</td>
<td>NSPACE($n^{1-\varepsilon}$)</td>
<td>NSPACE($n^{1-\varepsilon}$)</td>
<td>LOWER</td>
</tr>
<tr>
<td>EDTOL, EPDTOL</td>
<td>hP</td>
<td>NSPACE($n \log n$)</td>
<td>NSPACE(n)</td>
<td>NSPACE(n)</td>
<td>UPPER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NSPACE($n^{1-\varepsilon}$)</td>
<td>NSPACE($n^{1-\varepsilon}$)</td>
<td>NSPACE($n^{1-\varepsilon}$)</td>
<td>LOWER</td>
</tr>
<tr>
<td>EOL, EPOL</td>
<td>DSPACE($\log^2 n$)</td>
<td>DSPACE(n)</td>
<td>NSPACE(n)</td>
<td></td>
<td>UPPER</td>
</tr>
<tr>
<td></td>
<td>DTIME($n^4$)</td>
<td>hP</td>
<td>hP</td>
<td>hP</td>
<td>LOWER</td>
</tr>
<tr>
<td>EDOL, EPDOl</td>
<td>hP</td>
<td>hP</td>
<td>hP</td>
<td></td>
<td>UPPER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LOWER</td>
</tr>
<tr>
<td>CONTEXT FREE</td>
<td>hL</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>UPPER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LOWER</td>
</tr>
</tbody>
</table>
1. **TERMINOLOGY**

   \[ \text{DSPACE}(S(n)) = \{ L \mid L \text{ is accepted by some deterministic offline Turing machine which operates within space } S(n) \} \]

   \[ \text{NSPACE}(S(n)) \text{ is defined analogously for nondeterministic machines, and } \]

   \[ \text{and } \text{DTIME}(S(n)), \text{ NTIME}(S(n)) \text{ are defined similarly for the time measure.} \]

2. \[ \mathcal{L} = \text{DSPACE}(\log n), \quad \mathcal{N} = \text{NSPACE}(\log n) \]

   \[ \mathcal{P} = \bigcup_{k=1}^{\infty} \text{DTIME}(n^k), \quad \mathcal{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k) \]

3. A table entry of the form \( \begin{array}{c} U \\ \hline \end{array} \begin{array}{c} L \\ \hline \end{array} \) for problem \( P \) indicates that

   a) \( P \) is in class \( U \)

   b) If \( L \) is \( \mathcal{N} \), \( \mathcal{P} \) or \( \mathcal{NP} \), then some complete problem (and so any problem) in class \( L \) is reducible to \( P \)

   c) if \( L \) is \( \text{NSPACE}(S(n, \epsilon)) \), then for any \( \epsilon > 0 \), \( P \) is not in \( \text{NSPACE}(S(n, \epsilon)) \)

   d) if \( L \) is \( \mathcal{L} \), then any algorithm which solves \( P \) in \( \text{DSPACE}(S(n)) \)

      must satisfy \( \sup_{n \to \infty} \frac{S(n)}{\log n} > 0 \).

4. A table entry \( \begin{array}{c} LU \\ \hline \end{array} \) for problem \( P \) indicates that \( P \) is complete for class \( LU \).
3. OVERVIEW OF PROOF METHODS

Theorem \( \text{NONEMPTY}^{\text{ED}0L} \) is \( \mathcal{NP} \) hard.

Proof Method By Stockmeyer & Meyer [8] the following problem is \( \mathcal{NP} \)-hard:

Given a regular expression \( R \) of the form
\[
0^{P_1}(0^{Q_1})^* + \ldots + 0^{P_r}(0^{Q_r})^*
\]
to determine whether \( L(R) \neq 0^* \).

Construct an \( \text{ED}0L \) system
\[
G = (\{Z_i^j | 1 \leq j \leq r, 0 \leq j \leq p_i + q_i - 1\}, P, Z^0_1 \ldots Z^0_r, \{Z^j_i | j \neq p_i, 1 \leq i \leq r, 0 \leq j \leq p_i + q_i - 1\})
\]
where \( P \) consists of the productions
\[
(1 = 1, \ldots, r):
\]
\[
Z_i^j \rightarrow Z_i^{j+1} \quad \text{for } j = 0, \ldots, p_i - 1
\]
\[
Z_i^{p_i+q_i-1} \rightarrow Z_i^{p_i}
\]
Then \( L(G) \neq \emptyset \) iff \( L(R) \neq 0^* \); consequently \( \text{NONEMPTY}^{\text{ED}0L} \) is \( \mathcal{NP} \) hard.

Theorem \( \text{NONEMPTY}^{\text{ED}0L} \) is in \( \mathcal{NP} \).

Proof Method Let \( G = (V, P, w, \Sigma) \) be an \( \text{ED}0L \) grammar. Construct a
nondeterministic finite automaton \( M = (V, \{0\}, \delta, S_0, \Sigma) \) where
\[
S_0 = \{a \in V | a \text{ occurs in } w\}, \text{ and } \delta(a) = \{a_1, a_2, \ldots, a_m\} \text{ just in case } a \rightarrow a_1 \ldots a_m \text{ is a production in } P.
\]
It is easily seen that \( L(G) \neq \emptyset \) if and only if \( L(M) \neq 0^* \). By Stockmeyer and Meyer, this test can be carried out nonde-
terministically in polynomial time.

\[\square\]

Corollary \( \text{NONEMPTY}^{\text{ED}0L} \) is \( \mathcal{NP} \) complete.

Theorem \( \text{MEMBER}^{\text{EP}0L} \) is in \( \mathcal{NP} \).
Proof Method  Given $<G,x>$, we can determine whether $x \in L(G)$ as follows:

$$\alpha := \text{Axiom of } G;$$

for $l := 1$ Step 1 until $|x|$ do

begin choose $\beta$ so that $\alpha \Rightarrow^* \beta$ and $|\alpha| = |\beta|$;

if $\beta = x$ then accept;

choose $\gamma$ so $\beta \Rightarrow \gamma$ and $|\beta| < |\gamma|$;

$\alpha := \gamma$

end

This procedure will provide a polynomial time membership algorithm if the step "choose $\beta \ldots" can be done in polynomial time; however there can be nonrepeating derivations of length greater than any polynomial in $|G|$. Let $\alpha = a_1 \ldots a_m$ and $\beta = b_1 \ldots b_m$. Then $\alpha \Rightarrow^p \beta$ iff $a_1 \Rightarrow^p b_1, \ldots$, and $a_m \Rightarrow^p b_m$;
and $\alpha \Rightarrow^* \beta$ iff $\alpha \Rightarrow^p \beta$ for some $p \leq k^m$, where $k$ is the size of the alphabet of $G$. The test $a_i \Rightarrow^p b_i$ can be done by forming a connection matrix $M$ ($M(a,b) = 1$ iff $a \Rightarrow b$ is a production), and calculating $M, M^2, M^4, \ldots, M^{2^\left\lfloor m \log k \right\rfloor}$ by repeated squaring. $M^p$ may be obtained as a product of some of these matrices, chosen nondeterministically; and $\alpha \Rightarrow^p \beta$ may be easily determined from $M^p$. \qed

Theorem  NONEMPTY$^{E0L} \in \text{DSPACE}(n)$.

Proof  Let $G = (V,P,w,\Sigma)$ be given. Define $A_0 = \Sigma$, $A_{i+1} = \{a \mid a \Rightarrow^* \alpha \text{ is a production in } P \text{ such that } \alpha \in A_i^* \}$. Then $L(G) \neq \emptyset$ iff $w \in A_i^*$ for some $i$. The DSPACE($n$) algorithm is simply to calculate $A_0, A_1, \ldots$, storing only the most recent one (as a bit vector), and comparing the letters in $w$ against $A_i$. \qed
Theorem  \( \text{INFINITE}^{\mathbb{ET0L}} \notin \text{NSPACE}(n) \).

Proof Method  \( L(G) \) is infinite if and only if there exists a derivation of a word \( x \in L(G) \) such that \( S \Rightarrow^* v_1a v_2 \Rightarrow^* w_1a w_2 \Rightarrow^* x \), where \( a \Rightarrow^* \alpha \), \( \text{Alph}(v_1a v_2) = \text{Alph}(w_1a w_2) \), and \( \alpha \) contains the letter \( a \) and another occurrence of a letter, say \( b \), yielding a nonempty subword of \( x \).

The algorithm simulates such a derivation by nondeterministically choosing \( v_1a v_2, a, \) and \( b \) and checking whether the statements above are satisfied. The only information needed for that is information about the alphabet of the current sentential form and two letters derived from \( a \) and \( b \). □

Theorem  \( \text{MEMBER}^{\mathbb{ET0L}} \notin \text{NSPACE}(n^{1-\epsilon}) \) for any \( \epsilon > 0 \).

Proof  Let \( Z = (K, \Sigma, \Gamma, \#, \delta, q_0, \{ q_f \}) \) be an arbitrary 1 tape Turing machine which operates in space \( n \) (# is an end marker). For any \( x = a_1 \ldots a_n \), construct the \( \text{ET0L} \) system \( G_x = (V_n, \mathcal{T}_n, w_x, \{ 0 \}) \) where

\[
V_n = \{ g, 0 \} \cup \{ A^i \mid A \in \Gamma \text{ and } 0 \leq i \leq n+1 \} \cup K
\]

\[
w_x = p \neq 0 \quad a_1a_2 \ldots a_n \#^{n+1}
\]

for each \( (p, a) \in (K - \{ q_f \}) \times \Gamma \) there will be a table \( T_{p, a} \) in \( \mathcal{T}_n \) defined as follows:

If \( \delta(p, a) = (q, b, R) \) then

\[
T_{p, a} = \{ p \rightarrow q, a^0 \rightarrow b^{n+1} \} \cup \{ c^i \rightarrow c^{i-1} \mid c \in \Gamma, \ 0 < i \leq n+1 \} \cup G_{p, a}
\]

where \( G_{p, a} \) contains \( d \rightarrow g \) for every \( d \in V_n \) other than \( p, a^0 \) or \( c^i \) for \( c \in \Gamma, \ 0 < i \leq n+1 \).

If \( \delta(p, a) = (q, b, C) \) then

\[
T_{p, a} = \{ p \rightarrow q, a^0 \rightarrow b^0 \} \cup \{ c^i \rightarrow c^i \mid c \in \Gamma, \ 0 < i \leq n+1 \} \cup G_{p, a}
\]

If \( \delta(p, a) = (q, b, L) \) then

\[
T_{p, a} = \{ p \rightarrow q, a^0 \rightarrow b^1 \} \cup \{ c^i \rightarrow c^{i+1} \mid c \in \Gamma, \ 0 < i \leq n \}
\]

\[
\quad \quad \quad \quad \cup \{ c^{n+1} \rightarrow c^0 \mid c \in \Gamma \} \cup G_{p, a}.
\]
In addition, \( T_f \) contains the table

\[
T_f = \{ q_f \rightarrow 0 \} \cup \{ c^i \rightarrow 0 | c \in \Sigma, 0 \leq i \leq n+1 \} \cup \{ a \rightarrow g | a \in \Sigma \cup \{ g, 0 \} - \{ q_f \} \}.
\]

It is easily verified that \( Z \) yields an l.d. \( \alpha = b_0 \ldots b_{i-1} p b_i \ldots b_{n+1} \) iff \( G \) derives the string \( p b_{i-1+n+2} \ldots b_i b_{i+1+n+1} b_{i+1} b_i \ldots b_{i+1+n+1} \). Consequently, \( L(G) = \{ 0^n \} \) if \( Z \) accepts \( x \), and \( L(G) = \emptyset \) if \( Z \) does not accept \( x \).

Further, \( |G| = O(n \log n) \). In the usual way this implies

\[
\text{MEMBER}^{E0L} \notin \text{NSPACE}(n^{1-\epsilon}), \text{ for any } \epsilon > 0.
\]

**Corollary**

\( \text{MEMBER}^{E0L} \) is complete for polynomial space.

**Theorem** There is a deterministic E0L language \( L \) such that if \( L \) is in \( \text{DSPACE}(S(n)) \), then

\[
\sup_{n \to \infty} \frac{S(n)}{\log n} > 0.
\]

**Proof** \( L = \{ ab^n cd^n | n \geq 0 \} \) is clearly a deterministic E0L language. By Alt and Mehlhorn [1], \( L \) in \( \text{DSPACE}(S(n)) \) implies that \( S \) must satisfy the condition above. \( \square \)
REFERENCES


