<table>
<thead>
<tr>
<th>Title</th>
<th>Some Problems in Formal Language Theory Known as Decidable are Proved EXPTIME Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Kasai, Takumi; Iwata, Shigeki</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1992), 796: 8-21</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1992-07</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/82765">http://hdl.handle.net/2433/82765</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
<tr>
<td>Institution</td>
<td>Kyoto University</td>
</tr>
</tbody>
</table>
Some Problems in Formal Language Theory Known as Decidable are Proved EXPTIME Complete

Takumi Kasai* and Shigeki Iwata†

Abstract
Some problems in formal language theory are considered and shown deterministic exponential time complete. They include the problems for a given context-free language $L$, a regular set $R$, a deterministic context-free language $L_D$, to determine whether $L \subset R$, and to determine whether $L_D \subset R$.

1 INTRODUCTION

A number of complete problems for deterministic exponential time have been presented. Since Chandra and Stockmeyer [1] established the notion of alternation in 1976, many authors have shown complete problems for deterministic exponential time by using of alternation. Most of these problems were related to combinatorial games. [2, 5, 6, 7, 8]

We consider in this paper several problems in the formal language theory and show that the problems are deterministic exponential time complete. They were already known as decidable. Let $L$ be a context-free language, $R$ a regular set, $L_D$ a deterministic context-free language. The problems we consider include the ones to determine whether $L \subset R$, and whether $L_D \subset R$.

In order to prove that the concerned problems are deterministic exponential time-hard, we use the pebble game problem [5], which was already shown complete for deterministic exponential time, and we establish the polynomial-time reduction from the pebble game problem.

We write $\lambda$ to denote the empty string, and $|x|$ to denote the length of a string $x$. Let $\Sigma_k$ denote the set $\{[1], [2], \ldots, [k]\}$. See [4] for definitions of deterministic finite automata (dfa) $M = (Q, \Sigma, \delta, q_0, F)$ except that the transition function $\delta$ is given by a partial function from $Q \times \Sigma$ to $Q$. See also [4] for definitions of nondeterministic finite automata (nfa), regular set, context-free grammar (cfg), context-free language (cfl), deterministic context-free language (dcfl), deterministic pushdown automaton (dpda), Turing machine, polynomial time, and polynomial-time reducibility.

The Dyck language $D_k$ of $k$ balanced parenthesis is the one generated by the cfg $G = (\{S\}, \Sigma_k, P, S)$, where $P$ is the set of productions of the forms

$$S \rightarrow SS \mid \lambda \mid [i, S]; \ (1 \leq i \leq k).$$

*Department of Computer Science, University of Electro-Communications, Chofu, Japan
†Information Science Laboratory, Tokai University, Hiratsuka, Japan
For a cfg $G$, let $L(G)$ denote the language generated by $G$, and for an automaton or a machine $M$, $L(M)$ denote the language accepted by $M$. Whenever we say “given a cfl $L$, …”, we assume that a cfg $G$, $L(G) = L$, is given, and in particular when we say “given a dcl $L$, …”, a dpda $M$, $L(M) = L$ is assumed. When we say “given a regular set $R$, …”, it always means that an nfa $M$, $L(M) = R$ is given.

EXPTIME is the class of sets accepted by $2^n^k$ time bounded deterministic Turing machines for some $k$. A language $L$ is called EXPTIME complete if $L$ is in EXPTIME, and $L'$ is polynomial-time reducible to $L$ for any $L'$ in EXPTIME.

A pebble game [5] is a quadruple $G = (X, R, S, t)$ where:

1. $X$ is a finite set of nodes,
2. $R \subseteq \{ (x_a, x_b, x_c) | x_a, x_b, x_c \in X, x_a \neq x_b, x_b \neq x_c, x_c \neq x_a \}$ is called a set of rules,
3. $S$ is a subset of $X$, and
4. $t \in X$ is called the terminal node.

At the beginning of a pebble game, pebbles are placed on all nodes of $S$, and we call the placement the initial pebble-placement. A move of the game is as follows: if pebbles are placed on $x_a$, $x_b$, but not on $x_c$, and $(x_a, x_b, x_c) \in R$, then a player can move a pebble from $x_a$ to $x_c$ in his turn. The game is played by two players, and each player alternately applies one of the rules of $G$ to move a pebble. The winner is the player who can first put a pebble on the terminal node, or who can make the other player unable to move.

The first player has a forced win (or winning strategy) from a pebble-placement in $G$ if there is a winning game-tree for the first player, whose root is labeled with the pebble-placement. The winning game-tree of $G$ for the first player (game-tree for short) is the tree, nodes of which are labeled with pebble-placements, or WIN, where WIN means that the second player is already unable to move, thus the first player wins the game. We sometimes confuse a node of the game-tree with its label. A level of a node in the tree is the length of the path from the root to the node. The level of the root is zero. A depth of the game-tree is the maximum level among the nodes of the tree. Any node $v$ of the even level in the tree is labeled with a pebble-placement for the first player’s turn to move, and has exactly one child $v$, where $v$ is obtained by an application of a rule of the game to $v$. Any non-leaf of the odd level is labeled with a pebble-placement for the second player’s turn, and has exactly $m$ children, where $m$ is the number of the rules of the game. For $1 \leq j \leq m$, the $j$-th child of $v$ is labeled with a pebble-placement obtained by an application of the $j$-th rule $r_j$ of the game to $v$ if $r_j$ is applicable; and with WIN if $r_j$ is not applicable to $v$. Every leaf of the game-tree is labeled either with WIN or with a pebble-placement in which the first player wins.

The pebble game problem is, given a pebble game $G$, to determine whether there is a winning strategy for the first player from the initial pebble-placement in $G$.

**Theorem 1.1** [5] The pebble game problem is EXPTIME complete.

**Example 1.1** Consider the following pebble game $G = (X, R, S, t_5)$, where $X = \{x_1, x_2, x_3, x_4, x_5\}$, $S = \{x_1, x_2, x_3\}$, $R = \{r_1, r_2, r_3, r_4\}$, and $r_1 = (x_1, x_2, x_4), r_2 = (x_2, x_1, x_4), r_3 = (x_3, x_4, x_2), r_4 = (x_2, x_4, x_5)$. 
begin
1) Let $G, L = L(G)$ be a cfg, and let $M, R = L(M)$ be an nfa.
2) Construct a dfa $M'$ such that $L(M') = \Sigma^* - L(M)$.
3) Construct the cfg $G'$ as in Lemma 2.1 such that $L(G') = L(G) \cap L(M')$.
4) Use polynomial time algorithm to determine whether $L(G') = \phi$.
5) If $L(G') = \phi$ then $L \subset R$ else $L \not\subset R$.
end

Fig. 2.1 Algorithm to determine whether $L \subset R$

If the first player applies $r_1$ to move a pebble from $x_1$ to $x_4$, the second player then applies $r_4$ to move a pebble from $x_2$ to $x_5$ and the second player wins the game. Suppose that the first player first applies $r_2$ to move a pebble from $x_2$ to $x_4$. Then the only rule for the second player to apply is $r_3$ to move a pebble from $x_3$ to $x_2$. Then the first player applies $r_4$ to move a pebble from $x_2$ to $x_5$ and wins the game. Thus the first player has a forced win in $G$.

2 COMPLETE PROBLEM

Lemma 2.1 For a cfg $G$ and an nfa $M$, we can construct a cfg $G'$ such that $L(G') = L(G) \cap L(M)$ within polynomial time.

Proof. Let $G = (V, \Sigma, P, S)$ and $M = (Q, \Sigma', \delta, \{q_0\}, F)$. Without loss of generality, we assume that $\Sigma = \Sigma'$, and that $G$ is in Chomsky normal form. Let $G' = (V', \Sigma', P', S')$, $V' = \{[q, X, p] \mid q, p \in Q, X \in V \} \cup \{S\}$. $P'$ contains

$$
\begin{cases}
[q, X, p] \rightarrow a & \text{if } X \rightarrow a \in P \text{ and } p \in \delta(q, a), \\
[q, X, p] \rightarrow [q, A, q'][q', B, p] & \text{if } X \rightarrow AB \in P \text{ for } q, q', p \in Q,
\end{cases}
$$

and $S' \rightarrow [q_0, S, q_f]$ for $q_f \in F$.

By induction, we can prove that for $q, p \in Q, X \in V, w \in \Sigma^*$

$$
[q, X, p] \xrightarrow{\gamma} w \text{ if and only if } X \xrightarrow{\gamma} w \text{ and } p \in \delta(q, w).
$$

Thus

$$
S' \xrightarrow{\gamma} [q_0, S, q_f] \xrightarrow{\gamma} w \text{ if and only if } S \xrightarrow{\gamma} w \text{ and } q_f \in \delta(q_0, w).
$$

The number of productions in $G'$ is polynomial to the length of $G$ and $M$. Thus the construction of $G'$ can be performed within polynomial time. □

Next we present an algorithm in Fig.2.1 to determine whether $L \subset R$ for a given cfl $L$ and a regular set $R$.

Lemma 2.2 Given a cfl $L$ and a regular set $R$, the algorithm shown in Fig.2.1 determines whether $L \subset R$ within exponential time.

Proof. In line (2), apply an usual algorithm, for example p.22 of [4], to obtain a dfa $M_1$ such that $L(M) = L(M_1)$, and exchange the accepting states and the non-accepting states of $M_1$ to obtain $M''$, which accepts the complement of $R$. Note that the time
for the construction of $M'$ needs exponential time, since the number of states of $M_1$ is exponential compared with that of $M$.

In line (4), apply the CYK algorithm [9] for example.
In total, our algorithm runs in exponential time to determine whether $L \subset R$.

Consider the following problem $P_1$:

Given: a cfl $L$, and a regular set $R$.
To determine whether: $L \subset R$.

**Theorem 2.1** $P_1$ is EXPTIME complete.

**Proof.** Since EXPTIME is closed under complementation, it is sufficient to show that the problem $P_1'$:

Given: a cfl $L$, and a regular set $R$.
To determine whether: $L \not\subset R$.

is EXPTIME complete. By Lemma 2.2, $P_1'$ is solvable within exponential time.

To show that $P_1'$ is EXPTIME hard, we establish that the pebble game problem is polynomial-time reducible to $P_1'$. Let $G = (X, \tilde{R}, S, z_n)$ be a pebble game. We construct a cfg $G$ and an nfa $M$ within polynomial time such that there is a forced win for the first player in $G$ if and only if $L(G) \not\subset L(M)$.

Prior to the construction of $M$, we construct dfa's $M_1, M_2, \cdots, M_n$, where $n$ is the number of the nodes of $G$, such that there is a winning strategy for the first player in $G$ if and only if $L(G) \cap \bigcap_{i=1}^{n} L(M_i) \neq \emptyset$. Then we construct an nfa $M$, which accepts the complement of $\bigcap_{i=1}^{n} L(M_i)$. Thus $L(G) \cap \bigcap_{i=1}^{n} L(M_i) \neq \emptyset$ is equivalent to $L(G) \not\subset L(M)$.

We will explain briefly how the simulation of $G$ works in $G$ and $M_i$'s. The derivation of $G$ guesses a game-tree of $G$, that is, what rules of $G$ the first player applies in order to win the game. For the first player's turn to move in the game-tree, a derivation of $G$ guesses a rule which the first player applies to the pebble-placement, while for the second player's turn, derivations in $G$ guess for each rule whether the rule is applicable to the corresponding pebble-placement. The purpose of $M_i$'s is to examine whether the above guesses by $G$ are correct, and whether the derivation is the one for the first player to win the game.

Assume that $X = \{x_1, x_2, \cdots, x_n\}$, and that $\tilde{R} = \{r_1, r_2, \cdots, r_m\}$. We write $
abla_{4m} = \{r_j, \overline{r_j}, a_j, \overline{a_j}, b_j, \overline{b_j}, c_j, \overline{c_j} | 1 \leq j \leq m\}$, where a symbol without bar and the symbol with bar are intended to form a pair of balanced parenthesis in $\nabla_{4m}$. Let $G = (\{U, W, V_1, V_2, \cdots, V_m\}, \nabla_{4m}, P, U)$, where $P$ contains

(1) $W \to V_1 V_2 \cdots V_m$,

and for each rule $r_j = (x_{j1}, x_{j2}, x_{j3}), 1 \leq j \leq m$ of $\tilde{R}$,

(2) $\begin{cases} U \to r_j W \overline{r_j} \quad (j3 \neq n) \\ U \to r_j \overline{r_j} \quad (j3 = n) \end{cases}$,

(3) $V_j \to a_j \overline{a_j} | b_j \overline{b_j} | c_j \overline{c_j}$, and

(4) $V_j \to r_j U \overline{r_j} \quad (j3 \neq n)$.
The nonterminal \( U \) is associated with a pebble-placement for the first player's turn to move, while \( W \) is for the second player. \( V_j \), \( 1 \leq j \leq m \), means in the simulation to guess an application of a rule \( r_j \in \mathcal{G} \) to the pebble-placement associated with \( W \). The production rules in (2) are for the simulation of the first player in \( \mathcal{G} \) to select \( r_j \) to move a pebble from \( x_j1 \) to \( x_j3 \). The production \( U \rightarrow r_j W \overline{r_j} \) is the one to denote that the first player applies \( r_j \) and the next turn is the second player, while \( U \rightarrow r_j \overline{r_j} \) denotes for the first player to apply \( r_j \) and wins to put a pebble on \( x_n \). The productions (1),(3),(4) are for the second player's move. (1) is to try every rule \( r_1, r_2, \ldots, r_m \) as the second player's move. (3) is to indicate that \( r_j \) is not a proper rule to make: if a pebble is not on \( x_j1 \) (is not on \( x_j2 \), is on \( x_j3 \)), then \( V_j \rightarrow a_j \overline{a_j} \) \((V_j \rightarrow b_j \overline{b_j}, V_j \rightarrow c_j \overline{c_j})\), respectively) can be applied. (4) is to select \( r_j \) to move. \( V_j \rightarrow r_j \overline{r_j} \) is to apply \( r_j \) and the next turn is the first player.

For \( 1 \leq i \leq n, M_i \) keeps track of the existence of a pebble on \( x_i \) in \( \mathcal{G} \). If the state of \( M_i \) is in \( x_i \) \((\overline{x_i})\) then it means that there is \((\text{there is}, \text{respectively})\) a pebble on \( x_i \) in \( \mathcal{G} \). Let \( M_i = (\{x_i, \overline{x_i}\}, \Sigma_{4m}, \delta, q_i, \{q_i\}) \), and \( q_i = x_i \) for \( x_i \in S \), and \( q_i = \overline{x_i} \) for \( x_i \not\in S \). For each \( i \) \((1 \leq i \leq n) \) and \( j \) \((1 \leq j \leq m) \), let \( \delta_i(p_i, x_j) \), \( p_i \in \{x_i, \overline{x_i}\}, x_j \in \{r_j, \overline{r_j}, a_j, \overline{a_j}, b_j, \overline{b_j}, c_j, \overline{c_j}\} \), be the following transition. Assume that \( r_j = (x_j1, x_j2, x_j3) \) is a rule in \( \mathcal{G} \).

If \( i = j1 \) then \( \delta_i(p_i, x_j) \) is the transitions shown in Fig.2.2. If \( i = j2 \) then it is shown in Fig.2.3, and if \( i = j3 \) then it is in Fig.2.4. If \( i \not\in \{j1, j2, j3\} \) then \( \delta_i(p_i, x_j) = p_i \) for each \( p_i \in \{x_i, \overline{x_i}\} \), and \( x_j \in \{r_j, \overline{r_j}, a_j, \overline{a_j}, b_j, \overline{b_j}, c_j, \overline{c_j}\} \). Note that \( \delta_1(x_j1, a_j), \delta_2(x_j2, b_j), \) and \( \delta_3(x_j3, c_j) \) are undefined. (See Fig's 2.2, 2.3, and 2.4.)

The object of the construction of \( M_1, M_2, \ldots, M_n \) is to define a "product dfa" \( N \) of \( M_1, M_2, \ldots, M_n \), which is defined below. We consider \( N \) as a tool for the proof of the theorem, and we do not actually construct \( N \) in the simulation.
Fig. 2.4 transition $\delta_{j3}(p_{j3}, \sigma_{j})$

Now we define $N = (Q, \Sigma_{4m}, \delta, S, \{S\})$, where

\[Q = \{x_1, \overline{x}_1\} \times \{x_2, \overline{x}_2\} \times \cdots \times \{x_n, \overline{x}_n\},\]
\[S = (q_1, q_2, \ldots, q_n),\]
\[\delta((p_1, p_2, \ldots, p_n), \sigma) = (\delta_1(p_1, \sigma), \delta_2(p_2, \sigma), \ldots, \delta_n(p_n, \sigma)), p_i \in \{x_i, \overline{x}_i\},\]

and $\delta((p_1, p_2, \ldots, p_n), \sigma)$ is undefined if $\delta_i(p_i, \sigma)$ is undefined for some $i$.

We use a state $(p_1, p_2, \ldots, p_n)$ of $N$ and a pebble-placement $P$ of the game-tree in the same meaning: for each $i (1 \leq i \leq n)$, $p_i = x_i$ if and only if there is a pebble on $x_i$ in $P$, and $p_i = \overline{x}_i$ if and only if there is not a pebble on $x_i$ in $P$.

Then by the definition of $N$, we have the following lemmas 2.3 and 2.4:

**Lemma 2.3** Let $P$ be a pebble-placement and let $r_j$ be a rule of $G$. If $r_j$ is applicable to $P$ and if $P'$ is the resultant pebble-placement then

\[\delta(P, r_j) = P' \text{ and } \delta(P', \overline{r_j}) = P.\]

If $r_j$ is not applicable to $P$, then $\delta(P, r_j)$ is undefined.

**Proof.** Let $P = (p_1, p_2, \ldots, p_n)$ and let $r_j = (x_{j1}, x_{j2}, x_{j3})$. Suppose that $r_j$ is not applicable to $P$. Then either $p_{j1} = x_{j1}$ (there is not a pebble on $x_{j1}$), $p_{j2} = \overline{x}_{j2}$ (a pebble is not on $x_{j2}$), or $p_{j3} = x_{j3}$ (a pebble is on $x_{j3}$) holds. If $p_{j1} = x_{j1}$ then $\delta_{j1}(p_{j1}, r_j)$ is undefined (see Fig.2.2), if $p_{j2} = \overline{x}_{j2}$ then $\delta_{j2}(p_{j2}, r_j)$ is undefined (see Fig.2.3), and if $p_{j3} = x_{j3}$ then $\delta_{j3}(p_{j3}, r_j)$ is undefined (see Fig.2.4). Thus $\delta(P, r_j)$ is undefined.

Suppose that $r_j$ is applicable to $P$. Then $p_{j1} = x_{j1}$, $p_{j2} = x_{j2}$, and $p_{j3} = \overline{x}_{j3}$. Thus

\[\delta(P, r_j) = (p_1', p_2', \ldots, p_n'),\]
\[p_{j1}' = \overline{x}_{j1}, p_{j2}' = \overline{x}_{j2}, p_{j3}' = x_{j3}, \text{ and } p_i' = p_i, i \notin \{j1, j2, j3\}.\]

Further we have $\delta((p_1', p_2', \ldots, p_n'), \overline{r_j}) = P. \square$

**Lemma 2.4** For any pebble-placement $P$ and any symbol $\sigma \in \{a_j, \overline{a}_j, b_j, \overline{b}_j, c_j, \overline{c}_j | 1 \leq j \leq m\}$,

\[\delta(P, \sigma) = P \text{ or it is undefined.}\]

Further, $r_j$ is not applicable to $P$ if and only if there is $w_j \in \{a_j, \overline{a}_j, b_j, \overline{b}_j, c_j, \overline{c}_j\}$ such that $\delta(P, w_j) = P$. 

\[\delta_{j3}(p_{j3}, \sigma_{j})\]
Proof. For any $p_i \in \{x_i, \overline{x_i}\}, 1 \leq i \leq n$, and $\sigma \in \{a_j, b_j, c_j\}, 1 \leq j \leq m$, we have $\delta_i(p_i, \sigma) = p_i$. (See Fig's. 2.2, 2.3, and 2.4.) For any $\sigma \in \{a_j, b_j, c_j\}$, either $\delta_i(p_i, \sigma) = p_i$ or $\delta_i(p_i, \sigma)$ is undefined.

The necessary and sufficient condition that $\delta_i(p_i, a_j)$ is undefined is that $i = j1$ and $p_i = x_{j1}$, that is, there is a pebble on $x_{j1}$ in $P$. Likewise, the necessary and sufficient condition for $\delta_i(p_i, b_j)$ to be undefined is that $i = j2$ and $p_i = x_{j2}$, that is, a pebble is on $x_{j2}$ in $P$, and the necessary and sufficient condition for $\delta_i(p_i, c_j)$ to be undefined is that $i = j3$ and $p_i = x_{j3}$, that is, a pebble is not on $x_{j3}$ in $P$. Thus, $r_j$ is applicable to $P$ if and only if none of $\delta(P, a_j), \delta(P, b_j), \delta(P, c_j)$ are defined. \hfill \Box

Note that $L(G)$ is a subset of $D_{4m}$. Further we can obtain the following lemma:

**Lemma 2.5** For any $\alpha \in D_{4m}$ and a pebble-placement $P$,

$$\delta(P, \alpha) = P \text{ or it is undefined.}$$

*Proof. We can show the lemma by induction on $|\alpha|$. \hfill \Box*

**Lemma 2.6** The first player has a winning strategy from a pebble-placement $P$ if and only if there is $w \in \Sigma_{4m}^*$ such that

$$U \Rightarrow w \text{ and } \delta(P, w) = P.$$ 

**Example 2.1** Before we prove the lemma, consider the pebble game $G$ of Example 1.1. The cfg $G$ guesses the following derivation:

$$U \Rightarrow r_2Wr_2 \Rightarrow r_2V_1V_2V_3V_4r_2 \Rightarrow r_2b_1a_2\overline{a_2}r_3U\overline{r_3}a_4\overline{a_4}r_2 \Rightarrow r_2b_1a_2\overline{a_2}r_3U\overline{r_3}a_4\overline{a_4}r_2.$$ 

Let $P_0 = (x_1, x_2, x_3, \overline{x_4}, \overline{x_5})$. $P_0$ is the initial pebble-placement of $G$. Then

$$\delta(P_0, r_2) = (x_1, \overline{x_2}, x_3, x_4, \overline{x_5}) = P_1.$$ 

$P_1$ is the resultant pebble-placement after an application of $r_2$ to $P_0$.

Since there is not a pebble on the second component $x_2$ of $r_1$, $r_1$ is not applicable to $P_1$, and $\delta(P_1, b_1\overline{b_1}) = P_1$. Similarly, $r_2$ and $r_4$ are not applicable to $P_1$, since there is not a pebble on the first component $x_2$ of $r_2$ and $r_4$. Thus $\delta(P_1, a_2\overline{a_2}) = P_1$, and $\delta(P_1, a_4\overline{a_4}) = P_1$. Further

$$\delta(P_1, r_3) = (x_1, x_2, \overline{x_3}, x_4, \overline{x_5}) = P_2,$$

and

$$\delta(P_2, r_4) = (x_1, \overline{x_2}, \overline{x_3}, x_4, x_5) = P_3.$$ 

$P_2$ is the pebble-placement after the second player applies $r_3$ to $P_1$, and $P_3$ is the pebble-placement after the first player applies $r_4$ to $P_2$. The symbols $r_4$, $\overline{r_3}$, $\overline{r_2}$ are for backtracking procedures. Thus we have

$$\delta(P_3, r_4) = P_2, \delta(P_2, \overline{r_3}) = P_1, \text{ and } \delta(P_1, \overline{r_2}) = P_0.$$ 

Therefore, there is $w \in \Sigma_{4m}^*$ such that $U \Rightarrow w$, and $\delta(P_0, w) = P_0$. 

Proof. (Only if): There is a game-tree, the root of which is $P$. We will prove the “only if” part by induction on the depth of the game-tree. Assume that the depth of the tree is one. That is, the first player applies $r_j = (x_{j1}, x_{j2}, x_{j3})$ to put a pebble on $x_n$, and $j3 = n$. Then $U \Rightarrow r_j \overline{r_j}$, and if $P'$ is the resultant pebble-placement after the application of $r_j$ to $P$, then

$$\delta(P, r_j \overline{r_j}) = \delta(P', \overline{r_j}) = P$$

by Lemma 2.3. Thus the “only if” part holds for the basis of the induction.

Assume that the depth of the tree is greater than one, that $r_j = (x_{j1}, x_{j2}, x_{j3})$ is the first player's rule to apply to $P$ and that $P'$ is the resultant pebble-placement. Prior to show the inductive step, we will show that

for each $j (1 \leq j \leq m)$, there is $w_j \in D_{4m}$ such that

$$(*) \quad V_j \Rightarrow w_j, \delta(P', w_j) = P'.$$

If $r_j$ is not applicable to $P'$ then there is $w_j \in \{a_j \overline{a_j}, b_j \overline{b_j}, c_j \overline{c_j}\}$ which satisfies $(*)$ by Lemma 2.4.

Suppose that $r_j$ is applicable to $P'$, and that $P'_j$ is the pebble-placement after the application of $r_j$ to $P'$. Since the first player has a winning strategy from $P'_j$, there is $v_j \in \Sigma_{4m}$ such that

$$U \Rightarrow v_j, \delta(P'_j, v_j) = P'_j$$

by the inductive hypothesis. If we put $w_j = r_j v_j \overline{r_j}$ then

$$V_j \Rightarrow r_j U \overline{r_j} \Rightarrow r_j v_j \overline{r_j} = w_j,$$

$$\delta(P', w_j) = \delta(P'_j, v_j) = \delta(P'_j, \overline{r_j}) = P'.$$

Thus $(*)$ holds in the inductive step. We have shown $(*)$.

Therefore we have

$$U \Rightarrow r_j W \overline{r_j} \Rightarrow r_j V_1 \cdots V_m \overline{r_j} \Rightarrow r_j w_1 \cdots w_m \overline{r_j},$$

and

$$\delta(P, r_j \cdots w_m \overline{r_j}) = \delta(P', w_1 \cdots w_m \overline{r_j}) = \delta(P', \overline{r_j}) = P.$$

(If): We use induction on the number of steps of the derivation $U \Rightarrow w$. Assume that the number of the steps is one, that is, $U \Rightarrow r_j \overline{r_j} = w$. Obviously the first player has a winning strategy from $P$.

Assume that

$$U \Rightarrow r_j W \overline{r_j} \Rightarrow r_j V_1 \cdots V_m \overline{r_j} \Rightarrow r_j w_1 \cdots w_m \overline{r_j} = w,$$

$$V_j \Rightarrow w_j, (1 \leq j \leq m).$$

Since $\delta(P, w) = P$, $\delta(P, r_j)$ is defined. If $\delta(P, r_j) = P'$, then $P'$ is the pebble-placement after the application of $r_j$ to $P$, and $\delta(P', \overline{r_j}) = P$. By Lemma 2.5 and by $\delta(P', w_1 \cdots w_m) = P'$, we have

$$\delta(P', w_j) = P'.$$
for every \( j \) (\( 1 \leq j \leq m \)). If \( w_j \in \{a_j\overline{a_j}, b_j\overline{b_j}, c_j\overline{c_j}\} \), then \( r_j \) is not applicable to \( P' \) by Lemma 2.4. If \( w_j \notin \{a_j\overline{a_j}, b_j\overline{b_j}, c_j\overline{c_j}\} \), then \( r_j \) is applicable to \( P' \) and \( w_j \) is of the form \( r_jv_j\overline{r_j}, v_j \in D_{4m} \). Thus

\[
V_j \Rightarrow r_jU\overline{r_j} \Rightarrow r_jv_j\overline{r_j} = w_j, \text{ and } U \Rightarrow v_j.
\]

If \( \delta(P', r_j) = P'_j \) then \( P'_j \) is the pebble-placement after the application of \( r_j \) to \( P' \), and \( \delta(P'_j, v_j) = P'_j \). By the inductive hypothesis, \( U \Rightarrow v_j \) and \( \delta(P'_j, v_j) = P'_j \) imply that the first player has a winning strategy from \( P'_j \). Thus the first player can win the game no matter what rule \( r_j \) the second player may apply to \( P' \).

Therefore the lemma is proved.

By Lemma 2.6, the necessary and sufficient condition for the first player to have a winning strategy from the initial pebble-placement in \( G \) is that there is \( w \in \Sigma_{4m}^* \) such that \( w \in L(G) \cap L(N) \), and the condition is also that \( L(G) \cap \bigcap_{i=1}^{n} L(M_i) \neq \phi \).

To complete the proof of the theorem, we have to construct \( M \). It is clear that we can easily construct the dfa \( M'_i \) from \( M_i \) which accepts \( \Sigma_{4m}^* - L(M_i) \), the complement of \( L(M_i) \). Now we consider an nfa \( M \) such that \( M \) accepts the complement of \( \bigcap_{i=1}^{n} L(M_i) \). Since

\[
\Sigma_{4m}^* - \bigcap_{i=1}^{n} L(M_i) = \bigcup_{i=1}^{n} (\Sigma_{4m}^* - L(M_i)) = \bigcup_{i=1}^{n} L(M'_i) = L(M),
\]

we can construct an nfa \( M \) as the collection of \( M'_1, M'_2, \ldots, M'_n \) together with the initial state \( q_0 \) of \( M \) by simply adding \( \lambda \)-moves from \( q_0 \) to each initial state of \( M'_1, M'_2, \ldots, M'_n \). The set of the accepting states of \( M \) is the union of the ones of \( M'_1, M'_2, \ldots, M'_n \).

Therefore, there is a winning strategy for the first player from the initial pebble-placement in \( G \) if and only if \( L(G) \not\subset L(M) \). The constructions of \( G \) and \( M \) can be performed within polynomial time. We note that \( M \) can be constructed within polynomial time since \( M \) is nondeterministic. Thus both \( P'_1 \) and \( P_2 \) are complete for \( \text{EXPTIME} \). \( \square \)

3 Problems on dcfl's

We consider in this section some problems concerning dcfl's.

**Theorem 3.1** The problem \( P_2 \):

Given: a regular set \( R \subset \Sigma_2^* \).
To determine whether: \( D_2 \subset R \).

is \( \text{EXPTIME} \) complete.

**Proof.** To prove the theorem, it suffices to show that the following \( P'_2 \):

Given: a regular set \( R \subset \Sigma_2^* \).
To determine whether: \( D_2 \not\subset R \).
is EXPTIME complete. By Lemma 2.2, $P_2'$ is solvable within exponential time. We show that the pebble game problem is polynomial time reducible to $P_2'$. The proof proceeds similarly as in the one of Theorem 2.1.

Let $G = (X, R, S, x_n)$ be a pebble game, $X = \{x_1, x_2, \cdots, x_n\}$, $|R| = m$. Let $G$ be the cfg, let $M_1, M_2, \cdots, M_n$ be the dfa’s, and let $M$ be the nfa constructed in the proof of Theorem 2.1. We have shown in the preceding proof that the necessary and sufficient condition for the first player having a forced win from the initial pebble-placement in $G$ is $L(G) \not\subset L(M)$, hence $L(G) \cap \cap_{j=1}^{n} L(M_j) \neq \phi$. We will construct a dfa $M_0$ such that $L(G) = D_{4m} \cap L(M_0)$.

**Lemma 3.1** There exist a dfa $M_0$ such that $L(G) = D_{4m} \cap L(M_0)$.

**Proof.** Assume that $R_1$ is the set of rules of $G$ to put a pebble not on $x_n$, i.e., $R_1 = \{r_j | r_j = (x_{j1}, x_{j2}, x_{j3}), j3 \neq n\}$, and that $R_2$ is the set of rules to put a pebble on $x_n$, $R_2 = \{r_j | r_j = (x_{j1}, x_{j2}, x_{j3}), j3 = n\}$. Without loss of generality, we may assume that $R_1 = \{r_1, \cdots, r_1\}$ and $R_2 = \{r_{\ell+1}, \cdots, r_m\}$. We construct $M_0$, which is shown in Fig.3.1, where the transition $r_1 + \cdots + r_\ell$ from $U$ to $V_1$ stands for $\ell$ transitions by $r_1, \cdots, r_\ell$ from $U$ to $V_1$ (See Fig.3.2(a)). Transitions by $\overline{r_1} + \cdots + \overline{r_\ell}$ in Fig.3.1 are similar abbreviations. For $1 \leq j \leq m$, let $\mu_j = a_j \overline{a_j} + b_j \overline{b_j} + c_j \overline{c_j}$. The transition by $\mu_j$ from $V_j$ to $V_{j+1}$ implies that either $a_j \overline{a_j}, b_j \overline{b_j}$, or $c_j \overline{c_j}$ causes the transition from $V_j$ to $V_{j+1}$. (See Fig.3.2(b)).

Let $\delta$ be the transition function of $M_0$. Recall that $D_{4m}$ is generated by $G' = (\{S\}, \Sigma_{4m}, P, S)$, where $P$ contains $S \rightarrow SS | \lambda | [i, S]$, for $1 \leq i \leq 4m$. It is clear that $L(G) \subset D_{4m}$ since any derivation in $G$ can be “mapped into” a derivation in $G'$ by replacing $U, W, V_1, \cdots, V_m$ by $S$.

Thus in order to prove the lemma it suffices to show that for $\alpha \in D_{4m}$

$$U \xrightarrow{G} \alpha \text{ if and only if } \delta(U, \alpha) = U',$$
Fig. 3.2 abbreviations in Fig.3.1

for each $j (1 \leq j \leq m)$, $V_j \xrightarrow{r_j} \alpha$ if and only if $\delta(V_j, \alpha) = V_{j+1},$

$W \xrightarrow{\overline{r_j}} \alpha$ if and only if $\delta(V_1, \alpha) = V_{m+1}$.

(Only if): Let us use induction on $|\alpha|$. If $|\alpha| \leq 2$, the cases are trivial. Consider $\alpha, |\alpha| = k > 2$, assuming that the “only if” part holds for each $\beta \in D_{4m}, |\beta| < k$. Suppose $U \xrightarrow{\overline{r_j}} \alpha$. Then the first step of the derivation should be $U \xrightarrow{r_j} W \overline{r_j} G$ for some $j (1 \leq j \leq m)$, and $W \xrightarrow{\overline{r_j}} \beta \in D_{4m}, \alpha = r_j \beta \overline{r_j}, |\beta| < k$. By the inductive hypothesis, we have $\delta(V_1, \beta) = V_{m+1}$. Thus $\delta(U, \alpha) = \delta(U, r_j \beta \overline{r_j}) = \delta(V_1, \beta \overline{r_j}) = \delta(V_{m+1}, \overline{r_j}) = U'$.

The cases that $V_j \xrightarrow{r_j} \alpha$ and $W \xrightarrow{\overline{r_j}} \alpha$ can be similarly proved.

(If): By simple induction on $|\beta|, \beta \in D_{4m} - \{\lambda\}$, we can show that

(i) $\delta(U, \beta) = U'$ or it is undefined, and

(ii) for each $j (1 \leq j \leq m)$, $\delta(V_j, \beta) \in \{V_{j+1}, \cdots, V_{m+1}\}$ or it is undefined.

Again we will use induction on $|\alpha|$ to show the “if” part. If $|\alpha| \leq 2$ the proof is obvious. Consider $\alpha$, $|\alpha| = k > 2$, and assume that the “if” part holds for each $\beta, |\beta| < k$.

Suppose $\delta(U, \alpha) = U'$. If $\alpha = \alpha_1 \alpha_2$ and if $\alpha_1, \alpha_2 \in D_{4m} - \{\lambda\}$, then $\delta(U, \alpha_1) = U'$ by (i). The transition from $U'$ is made only by one of $\overline{r_1}, \cdots, \overline{r_7}$ and $\delta(U', \alpha_2)$ is undefined. Thus $M_0$ does not accept $\alpha_1 \alpha_2$. So $\alpha = r_j \beta \overline{r_j}$ for some $j (1 \leq j \leq \ell)$ and $\beta \in (D_{4m} - \{\lambda\})$. Since $\delta(U, r_j) = V_1$ and $\delta(V_1, \beta \overline{r_j}) = U'$, we obtain $\delta(V_1, \beta) = V_{m+1}$. By the inductive hypothesis we have $W \xrightarrow{\overline{r_j}} \beta$. Thus

$U \xrightarrow{\overline{r_j}} r_j W \overline{r_j} \xrightarrow{\overline{r_j}} r_j \beta \overline{r_j} = \alpha.$

The cases $\delta(V_1, \alpha) = V_{i+1}$ and $\delta(V_1, \alpha) = V_{m+1}$ can be similarly proved.$\square$

We define a homomorphism $h : \Sigma_{4m}^* \rightarrow \Sigma_2^*$ as follows:

$h([i]) = [i \overline{1}^{i}]$  \hspace{1cm} ($1 \leq i \leq 4m$)
Assume that $\Delta = \{h(l_i), h(r_i) \mid 1 \leq i \leq 4m\}$. Then the following lemma holds.

**Lemma 3.2** $h(D_{4m}) = D_2 \cap \Delta^*$.

**Proof.** By the definition of $h$ and $D_{4m}$, $h(D_{4m})$ is the language, which can be generated by the cfg $(\{S\}, \Sigma_2, P, S)$, where $P$ contains $S \rightarrow SS \mid \lambda \mid [1 \mid \Sigma_2, \Sigma_2 \mid 2]_1$ for $1 \leq i \leq 4m$. Thus the lemma follows.

We will complete the proof of Theorem 3.1. By the definition of $h$, for languages $L, L' \subseteq \Sigma_4^*$, we have that $L = \phi$ if and only if $h(L) = \phi$, and that $h(L \cap L') = h(L) \cap h(L')$. Thus

$$L(G) \not\subset L(M) \quad \text{if and only if} \quad D_{4m} \cap \bigcap_{i=0}^{n} L(M_i) = \phi$$

if and only if

$$h(D_{4m}) \cap \bigcap_{i=0}^{n} h(L(M_i)) = \phi.$$

It is easy to construct a dfa $\overline{M_i}$ such that $h(L(M_i)) = L(\overline{M_i})$ for $0 \leq i \leq n$. Let $\overline{M_n+1}$ be the dfa, which accepts $\Delta^*$. Then,

$$L(G) \not\subset L(M) \quad \text{if and only if} \quad D_2 \cap \Delta^* \cap \bigcap_{i=0}^{n} L(\overline{M_i}) = \phi$$

if and only if

$$D_2 \cap \bigcap_{i=0}^{n+1} L(\overline{M_i}) = \phi.$$ 

We can construct an nfa $\overline{M}$ which accepts the complement of $\bigcap_{i=0}^{n+1} L(\overline{M_i})$ as in the proof of Theorem 2.1, since $\overline{M_0}, \overline{M_1}, \cdots, \overline{M_n+1}$ are deterministic. Thus,

$$L(G) \not\subset L(M) \quad \text{if and only if} \quad D_2 \not\subset L(\overline{M}).$$

The construction of $\overline{M}$ can be performed within polynomial time. Therefore the proof of the theorem is completed.

**Corollary 3.1** For a given regular set $R$ and for each $k \geq 2$, the problem to determine whether $D_k \subset R$ is EXPTIME complete.

**Proof.** The problem can be solved within EXPTIME. Let $R$ be a regular set. We prove that

$$D_2 \subset R \quad \text{if and only if} \quad D_k \subset R \cup (\Sigma_k^* - \Sigma_2^*).$$

Assume that $D_2 \subset R$, and that $w \in D_k$. If $w \in \Sigma_2^*$ then $w \in D_2$. If $w \not\in \Sigma_2^*$ then $w \in \Sigma_k^* - \Sigma_2^*$. Thus $w \in R \cup (\Sigma_k^* - \Sigma_2^*)$ and we obtain that $D_k \subset R \cup (\Sigma_k^* - \Sigma_2^*)$.

Assume that $D_k \subset R \cup (\Sigma_k^* - \Sigma_2^*)$, and $w \in D_2$. Since $w \in R \cup (\Sigma_k^* - \Sigma_2^*)$ and $w \not\in \Sigma_k^* - \Sigma_2^*$, we obtain that $w \in R$. Thus $D_2 \subset R$.

As we can construct the nfa accepting $R \cup (\Sigma_k^* - \Sigma_2^*)$ within polynomial time, the corollary is proved.

**Open problem 1** The complexity of the problem to determine whether $D_1 \subset R$ for a given regular set $R$ is remained open.

Since we can construct a dpda $M$ to accept $D_2$, we obtain the following corollary.
Corollary 3.2 The problem $P_3$:

$Given$: a $dcfl$ $L$, and a regular set $R$.

$To$ $determine$ $whether$: $L \subseteq R$.

is $EXPTIME$ complete.

Corollary 3.3 The problem $P_4$:

$Given$: a $dcfl$ $L \subseteq \Sigma^*$, and a regular set $R \subseteq \Sigma^*$.

$To$ $determine$ $whether$: $L \cup R = \Sigma^*$.

is $EXPTIME$ complete.

Proof. Let $M$ be a dpda which accepts $L$. Since $M$ is deterministic, we can construct a dpda $M'$ such that $M'$ accepts $\Sigma^* - L$. (See [4], p.238, for example.) Then we can construct a cfg $G$, which satisfies $L(G) = L(M')$.

Since $L \cup R = \Sigma^*$ is equivalent to $L(G) \subseteq R$, and $G$ can be constructed within polynomial time, $P_4$ is $EXPTIME$ complete by Corollary 3.2.

Remark The problem to determine whether $R \subset L$ for a given regular set $R$ and a $dcfl$ $L$ is solvable within polynomial time by constructing a cfg $G$ generating the complement of $L$ and by applying the algorithm of Fig.2.1 to determine whether $R \cap L(G) = \phi$, which is equivalent to $R \subset L$.

Open problem 2 Let $L$ be a $dcfl$ and $R$ be a regular set. The following problems are in $EXPTIME$, however, their complexities are open.

1. $R = L$?
2. $L \subsetneq R$?
3. $R \subsetneq L$?

REFERENCES


謝辞 1

岐阜ペンなんて忘れてしまったでかんわ。町田さん来年も開くまわししてちょう。

謝辞 2

くに荘のおばさん、おっかなかったよ。町田さん、研究会しらせてくれてあんがと。またひておれ。