数理解析研究所講究録
第 731 巻 1990 年 73-83

語彙機能文法と計算量
Lexical-Functional Grammars and Computational Complexity

西野哲朗
Tetsuro Nishino

東京電機大学理工学部情報科学科
Department of Information Sciences, Tokyo Denki University

Abstract

Lexical-functional grammars (LFGs) have been widely used to formally specify the syntax of natural languages. In this paper, we show the followings : (1) the emptiness problem for LFGs is undecidable, and (2) the membership problem for LFGs with at least one ε-production is $EXPTIME$-hard.

1 Lexical-Functional Grammars

In this section, we describe the overview of LFG necessary for understanding the material in this paper. For details, see [3]. We first illustrate the LFG machinery by a linguistic example, then describe the formal definition of LFG.

In a sentence description of LFG, there are two types of componets, a *constituent structure* (*c-structure*) and a *functional structure* (*f-structure*). Using these two structures, a LFG system specify a set of grammatical sentences. A c-structure is a standard parsing tree of a context-free grammar, which represents the superficial arrangements of words and phrases in the sentence. While an f-structure is a hierarchical structure constructed by the pairs of names of grammatical functions and their unique values. The f-structure mainly represents the configuration of the surface grammatical functions.

An LFG system is specified by a set of *annotated phrase structure rules.* An annotated phrase structure rule is a context-free rule associated with functional equations. Fig. 1 illustrates a simple example of LFG. We deal with this LFG throughout the examples in this paper. A c-structure is generated using annotated phrase structure rules ignoring the functional equations appearing in them. Fig. 2 illustrates an example of a c-structue genarated by the LFG in Fig. 1 (The meaning of variables x_i, $1 \leq i \leq 12$ in Fig. 2 will be explained later.).

Each node in a c-structure is assumed to have an associated f-structure. The metavariable $\downarrow$ attached to the node n in a c-structure represents the f-structure associated to n. While the metavariable $\uparrow$ attached to n represents the f-structure associated to the parent of n. Therefore a defining equation of the form $\uparrow=\downarrow$ attached to n represents that the f-structure associated to n is equal to the f-structure associated to the parent of n. Furthermore a defining equation of the form ($\uparrow$ OBJ) $=\downarrow$ attached to n represents that the f-structure associated to n is equal to the value of the OBJ of the f-structure associated to the parent of n.

Given a c-structure t, the f-structure corresponding to t is determined by the following procedure :

```
SE →       NP          VP
     (↑ SUBJ)=↓   ↑=↓

NP → DET   N
     ↑=↓   ↑=↓

VP →  V       NP           NP
     ↑=↓  (↑ OBJ)=↓  (↑ OBJ2)=↓
```

the:	DET,	(↑ SPEC)=THE
boy:	N,	(↑ NUM)=SG (↑ PRED)='BOY'
handed:	V,	(↑ TENSE)=PAST (↑ PRED)='HAND<(↑ SUBJ)(↑ OBJ)(↑ OBJ2)>'
girl:	N,	(↑ NUM)=SG (↑ PRED)='GIRL'
a:	DET,	(↑ SPEC)=A (↑ NUM)=SG
candy:	N,	(↑ NUM)=SG (↑ PRED)='CANDY'

Figure 1: An example of LFG.

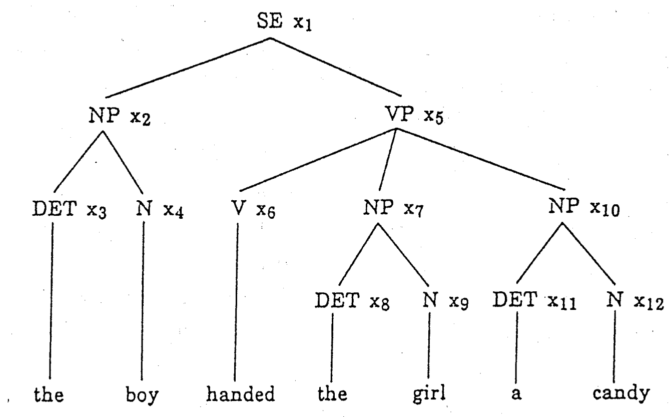

Figure 2: A c-structure genarated by the LFG in Fig. 1.

$(x_1\ \mathrm{SUBJ})=x_2,$
$x_1=x_5,$
$x_2=x_3,$
$x_2=x_4,$
$(x_3\ \mathrm{SPEC})=\mathrm{THE},$
$(x_4\ \mathrm{NUM})=\mathrm{SG},$
$(x_4\ \mathrm{PRED})=\text{'BOY'},$
$x_5=x_6,$
$(x_5\ \mathrm{OBJ})=x_7,$
$(x_5\ \mathrm{OBJ2})=x_{10},$
$(x_6\ \mathrm{TENSE})=\mathrm{PAST},$
$(x_6\ \mathrm{PRED})=\text{'HAND}<(\uparrow \mathrm{SUBJ})(\uparrow \mathrm{OBJ})(\uparrow \mathrm{OBJ2})>\text{'},$
$x_7=x_8,$
$x_7=x_9,$
$(x_8\ \mathrm{SPEC})=\mathrm{THE},$
$(x_9\ \mathrm{NUM})=\mathrm{SG},$
$(x_9\ \mathrm{PRED})=\text{'GIRL'},$
$x_{10}=x_{11},$
$x_{10}=x_{12},$
$(x_{11}\ \mathrm{SPEC})=\mathrm{A},$
$(x_{11}\ \mathrm{NUM})=\mathrm{SG},$
$(x_{12}\ \mathrm{NUM})=\mathrm{SG},$
$(x_{12}\ \mathrm{PRED})=\text{'CANDY'}$

(a)

$$\left[\begin{array}{ll}
\mathrm{SUBJ} & \left[\begin{array}{ll}\mathrm{SPEC} & \mathrm{THE}\\ \mathrm{NUM} & \mathrm{SG}\\ \mathrm{PRED} & \text{'BOY'}\end{array}\right]\\
\mathrm{TENSE} & \mathrm{PAST}\\
\mathrm{PRED} & \text{'HAND}<(\uparrow \mathrm{SUBJ})(\uparrow \mathrm{OBJ})(\uparrow \mathrm{OBJ2})>\text{'}\\
\mathrm{OBJ} & \left[\begin{array}{ll}\mathrm{SPEC} & \mathrm{THE}\\ \mathrm{NUM} & \mathrm{SG}\\ \mathrm{PRED} & \text{'GIRL'}\end{array}\right]\\
\mathrm{OBJ2} & \left[\begin{array}{ll}\mathrm{SPEC} & \mathrm{A}\\ \mathrm{NUM} & \mathrm{SG}\\ \mathrm{PRED} & \text{'CANDY'}\end{array}\right]
\end{array}\right]$$

(b)

Figure 3: (a) An f-description and (b) an f-structure corresponding to the c-structure in Fig. 2.

1. Assign a unique variable to each internal node of t which represents the f-structure associated to the node.

2. Using the variables of (a), *instantiate* (i.e. have the appropriate variables filled in for the arrows) all the equations associated to the nodes in t. The set of equations thus produced is called the *f-description* of t. (The f-description produced from the c-structure in Fig. 2 is shown in Fig. 3 (a).)

3. Solve the f-description of (b) algebraically to obtaine the value in the f-description's unique minimal solution of the $\downarrow$-variable of the SE node. (This value is called the f-srtucture *assigned to the string.* The obtained value for x_1 of the f-description in Fig. 3 (a) is the f-structure shown in Fig. 3 (b). This f-structure is the one assigned to the string *the boy handed the girl a candy.*) For details of this solution algorithm, see [3].

As shown in Fig. 3 (b), an f-structure is a hierarchical structure constructed by the ordered pairs each of which consists of a function name and its unique value. The f-description solution algorithm in [3] constructs one solution (f-structure) for a properly instantiated f-description if it is possible.

Grammatical functions whose values can serve as arguments to semantic predicates are called *governable grammatical functions.* The *well-formedness conditions* on f-structures are defined as follows :

Well-formedness conditions on f-structures

1. (*uniqueness*) In a given f-structure, each function name may have at most one value.

2. (*completeness*) An f-structure is locally complete iff it contains all the goveranable grammatical functions that serve as arguments to its predicate. An f-structure is *complete* iff it and its subsidiary f-structure are locally complete.

3. (*coherency*) An f-structure is locally coherent iff all governable grammatical functions that it contains serve as arguments to a local predicate. An f-structure is *coherent* iff it and all its subsidiary f-structures are locally coherent.

A string is *grammatical* only if it has a valid c-structure and it is assigned a well-formed f-structure. A language *generated by LFG G*, denoted by $L(G)$, is a set of grammatical strings of G. The class of languages generated by lexical functional grammars is denoted by $\mathcal{L}_{LFG}$. From the definition of LFG, it is obvious that the class of languages generated by LFGs includes the class of context-free languages.

In this paper, we deal with a subclass of LFGs which is called restricted LFGs. We now describe a formal definition of restricted LFGs. The following definition is based on the one of Pinker [4].

Definition 1 A *restricted lexical functional grammar* (RLFG for short) is a 7-tuple $G = (\Sigma,\ \Gamma,\ S,\ FN,\ FV,\ PR,\ AR)$ consists of 1-7 as follows :

1. Σ is a finite *terminal alphabet.*

2. Γ is a finite *nonterminal alphabet.*

3. S is a *start symbol.*

4. FN is a finite set of *function names.*

5. FV is a finite set of *function values.*

6. PR is a finite set of *predicates.*

7. AR is a finite set of *annotated phrase structure rules.* An annotated phrase structure rule is of the form
$$A \ \rightarrow\ (B_1, E_1)\ (B_2, E_2)\ \dots\ (B_m, E_m),$$
where $A \in \Gamma$, $B_1, B_2, \dots, B_m \in \Sigma \cup \Gamma$ and E_i is a functional equation set, with the constraint that for all i, j such that $i \neq j$, $(B_i, E_i) \neq (B_j, E_j)$. A *functional equation set* is a set of statements of the form
$$M_1F_1F_2 \ =\ M_2F_3F_4,$$
where $M_1, M_2 \in \{\uparrow, \downarrow\}$ ($\uparrow$ and $\downarrow$ are called *metavariables*), and $F_i \in FN \cup FV$ for each i, $1 \leq i \leq 4$. In the above equation, any symbol except $=$ may be null, but the left-hand side of the equation must be one of the following forms : $M_1F_1F_2$, M_1F_1, M_1 or F_1, and this is the same for the right-hand side of the equation. An annotated phrase structure rule of the form $A \ \rightarrow\ (B_1, E_1)$, where $B_1 \in \Sigma$ and E_1 is a functional equation set, is especially called a *lexical insertion rule* or a *lexical entry.*

□

Example 1 The LFG in Fig. 1 is indeed an RLFG defined as follows:
$G = (\Sigma,\ \Gamma,\ SE,\ FN,\ FV,\ PR,\ AR)$, where
$\Sigma = \{the,\ boy,\ handed,\ girl,\ a,\ candy\}$,
$\Gamma = \{SE,\ NP,\ VP,\ DET,\ N,\ V\}$,
$FN = \{SUBJ,\ OBJ,\ OBJ2,\ SPEC,\ NUM,\ PRED,\ TENCE\}$,
$FV = \{A,\ THE,\ SG,\ PAST\}$
$PR = \{'BOY',\ 'GIRL',\ 'CANDY',\ 'HAND < (\uparrow\ SUBJ)(\uparrow\ OBJ)(\uparrow\ OBJ2) >'\}$

and AR contains the following rules :
$SE \rightarrow (NP,\ \{(\uparrow\ SUBJ) = \downarrow\})(VP,\ \{\uparrow=\downarrow\})$,
$NP \rightarrow (DET,\ \{\uparrow=\downarrow\})(N,\ \{\uparrow=\downarrow\})$,
$VP \rightarrow (V,\ \{\uparrow=\downarrow\})(NP,\ \{(\uparrow\ OBJ) = \downarrow\})(NP,\ \{(\uparrow\ OBJ2) = \downarrow\})$,
$DFT \rightarrow (the,\ \{(\uparrow\ SPEC) = THE\})$,
$N \rightarrow (boy,\ \{(\uparrow\ NUM) = SG,\ (\uparrow\ PRED) =' BOY'\})$, etc.

Remarks.

(1) Let $A \in NV$, $X \in TV$ and E be a functional equation set. As shown in Fig. 1, a lexical insertion rule $A \rightarrow (X, E)$ is usually written as

$$X\ :\ A,\ E.$$

(2) As can be seen from the definition, no more than two function names may appear on either side of functional equation. This constraint is identical to the following *functional locality constraint* of Kaplan and Bresnan [3] : no rule in the grammar may refer to symbols separated by more than a single level of embedding in an f-structure.
(3) As mentioned in [4], entries for predicates taking a number of arguments can be broken down into a set of equations, each one specifying the gramatical function assigned to one argument of the predicate. For example, the lexical entry for the word *handed* in Fig. 1 would be translated into the following annotated phrase structure rule :

$$\begin{array}{ll} V \rightarrow & \textit{handed} \\ & (\uparrow \text{TENSE}) = \text{PAST} \\ & (\uparrow \text{PRED}) = \text{'HAND'} \\ & (\uparrow \text{ARG1}) = (\uparrow \text{SUBJ}) \\ & (\uparrow \text{ARG2}) = (\uparrow \text{OBJ}) \\ & (\uparrow \text{ARG3}) = (\uparrow \text{OBJ2}) \end{array}$$

So, without loosing generality, we can assume that functional equations are of the form described in the above definition. But, for the simplicity sake, we write the lexical entries as in Fig. 1.
(4) Metavariables are left-associative, i.e. $M_1F_1F_2 = ((M_1\ F_1)\ F_2)$. In order to improve the readability, we often write $((M_1\ F_1)\ F_2)$ rather than $M_1F_1F_2$.
(5) In the general LFG machinery, there are two types of metavariables. They are (a) *immediate domination metavariables* (i.e. $\uparrow$ and $\downarrow$) and (b) *bounded domination metavariables* (i.e. $\Uparrow$ and $\Downarrow$). We have already seen immediate domination metavariables in the above example. The bounded domination

metavariables $\Uparrow$ and $\Downarrow$ are used to represent *long distance binding* used for handling *wh* movement in examples such as *Which boy did the girl kiss.*

Theorem 1 $CFL \not\subseteq \mathcal{L}_{RLFG}$

Proof Since any CFG generates c-structures with no functional equations, the class CFL is obviously contained in $\mathcal{L}_{RLFG}$. In order to show that this containment is proper, it is suffice to see that the following RLFG G generates the language $\{a^n b^n c^n \mid n \geq 1\}$ which is a famous example of the language that is known not to be a context-free language. This LFG is originally given in [3].

$G = (\Sigma,\ \Gamma,\ S,\ FN,\ FV,\ PR,\ AR)$, where
$\Sigma = \{a, b, c, \}$, $\Gamma = \{S, A, B, C\}$, $FN = \{N\}$, $FV = \{0\}$, $PR = \emptyset$ and AR contains the following rules :
$S \rightarrow (A, \{\uparrow=\downarrow\})(B, \{\uparrow=\downarrow\})(C, \{\uparrow=\downarrow\})$,
$A \rightarrow (a, \emptyset)(A, \{(\uparrow\ N) =\downarrow\})$,
$B \rightarrow (b, \emptyset)(A, \{(\uparrow\ N) =\downarrow\})$,
$C \rightarrow (c, \emptyset)(A, \{(\uparrow\ N) =\downarrow\})$,
$A \rightarrow (a, \{(\uparrow\ N) = 0\})$,
$B \rightarrow (b, \{(\uparrow\ N) = 0\})$,
$C \rightarrow (c, \{(\uparrow\ N) = 0\})$. □

We denote the class of context-sensitive languages by CSL. The following theorem is known.

Theorem 2 [3] $\mathcal{L}_{LFG} \subseteq CSL$ □

Corollary 3 $\mathcal{L}_{RLFG} \subseteq CSL$ □

2 The Emptiness Problem for LFGs

The *emptiness problem* for LFGs is as follows : given an LFG G, decide whether $L(G) = \emptyset$. In this section, we first show that the emptiness problem for LFGs is undecidable. We reduce Post's Corresponding Problem to the emptiness problem. An instance of *Post's Corresponding Problem* (*PCP* for short) consists of two lists, $A = (x_1, x_2, ..., x_k)$ and $B = (y_1, y_2, ..., y_k)$ of strings over some alphabet Σ. An instance of PCP is said to *have a solution* if there exists a sequence of integers $i_1, i_2, ..., i_m$ $(m \geq 1)$ such that

$$x_{i_1}\ x_{i_2}\ \dots\ x_{i_m} = y_{i_1}\ y_{i_2}\ \dots\ y_{i_m}.$$

In this case, the sequence $i_1, i_2, ..., i_m$ is called a *solution* to this instance of PCP, and the strings $x_{i_1}\ x_{i_2}\ \dots\ x_{i_m}$ ($= y_{i_1}\ y_{i_2}\ \dots\ y_{i_m}$) is called a *value* of this instance. The following theorem is well known (see for example [2]).

Theorem 4 PCP is undecidable. □

Example 2 Let $\Sigma = \{0, 1\}$, $A = (01, 11)$ and $B = (1011, 1)$. This instance of PCP has a solution $i_1 = 2$, $i_2 = 1$ and $i_3 = 2$ ($m = 3$). Then $x_2\ x_1\ x_2 = y_2\ y_1\ y_2 = 110111$.

Theorem 5 The emptiness problem for RLFGs is undecidable.

Proof Sketch. We reduce PCP to the emptiness problem. Let $\Sigma = \{0, 1\}$. Then let $A =$

$(x_1, x_2, ..., x_k)$ and $B = (y_1, y_2, ..., y_k)$ be an instance of PCP, where for each i $(1 \leq i \leq k)$, $x_i = a_1\ a_2\ ...\ a_{m(i)}$ with $a_j \in \Sigma$ for each j, $1 \leq j \leq m(i)$, and $y_i = b_1\ b_2\ ...\ b_{n(i)}$ with $b_j \in \Sigma$ for each j, $1 \leq j \leq n(i)$. From the lists A and B, we construct the following RLFG $G(A, B) = (\Sigma, \Gamma, S, FN, FV, PR, AR)$:

$\Gamma = \{S, I, A, B\} \cup \{A_j^i \mid 1 \leq i \leq k, 1 \leq j \leq m(i)\} \cup$
$\{B_j^i \mid 1 \leq i \leq k, 1 \leq j \leq n(i)\}$,
$FN = \{TAPE, LIST, REST, H, T, L\}$,
$FV = \{0, 1, 2, ..., k, \$\}$, $PR = \emptyset$ and

AR contains the following annotated phrase structure rules :

1. $S \rightarrow (I, E_1)\ (A, E_2)\ (B, E_3)$, where
   $E_1 = \{(\uparrow\ TAPE) = \downarrow\}$ and
   $E_2 = E_3 = \{(\downarrow\ TAPE) = (\uparrow\ TAPE), (\uparrow\ LIST) = (\downarrow\ LIST), (\downarrow\ L) = \$\}$.
2. $I \rightarrow (I, E_1)\ (0, E_2)$, $I \rightarrow (I, E_3)\ (1, E_4)$, where
   $E_1 = E_3 = \{((\uparrow\ TAPE)\ T) = \downarrow\}$,
   $E_2 = \{((\uparrow\ TAPE)\ H) = 0\}$ and
   $E_4 = \{((\uparrow\ TAPE)\ H) = 1\}$.
3. $I \rightarrow (0, E_1)$, $I \rightarrow (1, E_2)$, where
   $E_1 = \{((\uparrow\ TAPE)\ H) = 0, ((\uparrow\ TAPE)\ T) = \$\}$ and
   $E_2 = \{((\uparrow\ TAPE)\ H) = 1, ((\uparrow\ TAPE)\ T) = \$\}$.
4. $A \rightarrow (A, E_1)\ (A_i^1, E_2)$, $B \rightarrow (B, E_3)\ (B_i^1, E_4)$ $(\ 1 \leq i \leq k\ )$, where
   $E_1 = E_3 = \{(\downarrow\ TAPE) = (\uparrow\ REST), ((\downarrow\ L)\ H) = i, ((\downarrow\ L)\ T) = (\uparrow\ L)\}$ and
   $E_2 = E_4 = \{(\downarrow\ TAPE) = (\uparrow\ TAPE), (\uparrow\ REST) = (\downarrow\ REST)\}$.
5. $A_i^j \rightarrow (A_i^{j+1}, E_1)\ (a_j, E_2)$, $B_i^l \rightarrow (B_i^{l+1}, E_3)\ (b_l, E_4)$
   $(1 \leq i \leq k,\ 1 \leq j \leq m(i) - 1,\ 1 \leq l \leq n(i) - 1)$, where
   $E_1 = E_3 = \{(\downarrow\ TAPE) = ((\uparrow\ TAPE)\ T), (\uparrow\ REST) = (\downarrow\ REST)\}$,
   $E_2 = \{((\uparrow\ TAPE)\ H) = a_j\}$ and
   $E_4 = \{((\uparrow\ TAPE)\ H) = b_l\}$.
6. $A_i^{m(i)} \rightarrow (a_{m(i)}, E_1)$, $B_i^{n(i)} \rightarrow (b_{n(i)}, E_2)$ $(1 \leq i \leq k)$, where
   $E_1 = \{((\uparrow\ TAPE)\ H) = a_{m(i)}, (\uparrow\ REST) = ((\uparrow\ TAPE)\ T)\}$ and
   $E_2 = \{((\uparrow\ TAPE)\ H) = b_{n(i)}, (\uparrow\ REST) = ((\uparrow\ TAPE)\ T)\}$.
7. $A \rightarrow (A_i^1, E_1)$, $B \rightarrow (B_i^1, E_2)$ $(1 \leq i \leq k)$, where
   $E_1 = E_2 = \{(\downarrow\ TAPE) = (\uparrow\ TAPE), ((\downarrow\ L)\ H) = i,$
   $((\downarrow\ L)\ T) = (\uparrow\ L), (\uparrow\ LIST) = (\downarrow\ L), (\downarrow\ REST) = \$\}$.

From the construction, it is easy to see that an instance (A, B) of PCP has a value y (i.e. has a solution) iff $(y^R)^3 \in L(G(A, B))$ iff $L(G(A, B)) \neq \emptyset$, where y^R denotes the reverse of y. □

Corollary 6 The emptiness problem for LFGs is undecidable. □

3 Lower Bounds on the Membership Problem

In this section, we show that the membership problem for LFGs which have at least one ε-production is $EXPTIME$-hard. We first briefly describe the basic concepts in computational complexity. For details, see [1, 2].

Definition 2 A *one-tape alternating Turing machine* (ATM for short) is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F, U)$ where :

1. Q is the finite set of *states.*
2. Σ is the finite *input alphabet.*
3. Γ is the finite *tape alphabet.*
4. δ is the *next move relation* mapping an element of $Q \times \Sigma$ to a subset of $Q \times \Sigma \times D$, where $D = \{L, R\}$.
5. $q_0 \in Q$ is the *initial state.*
6. $F \subseteq Q$ is the set of *accepting states.*
7. $U \subseteq Q$ is the set of *universal states.* $Q - U$ is called the set of *existential states.*

A machine move is represented as follows. Let $\delta(q, x)$ be of the following form.

$$\delta(q,x) = \{(q_1, y_1, d_1), (q_2, y_2, d_2), ..., (q_m, y_m, d_m)\},$$

where $q, q_1, ..., q_m \in Q$, $d_1, d_2, ..., d_m \in D$ and $x, y_1, ..., y_m \in \Sigma$. In state q, scanning symbol x, M takes the following action ACT(i) for some i, $1 \leq i \leq m$ if q is an existential state, and takes ACT(i) for all i, $1 \leq i \leq m$ if q is a universal state.

ACT(i) : Rewrite x as y_i, move the tape head one position in the derection of d_i, and change the state to q_i.

A *configuration* of M consists of the state, head position, and contents of the tapw. Let C be a configuration of M. We denote the set of possible configurations after one move of M by $Next(C)$. A configuration is *existential* (resp. *universal, initial, accepting*) if the state of M in that configuration is an existential (resp. universal, initial, accepting) state. A value $v(C)$ of C is either true or false defined by the following procedure.

```
Procedure  EVAL ( $C$ : a configuration of an ATM );
  begin
    if $C$ is an accepting configuration
      then $v(C) := true$
      else if $C$ is an existential configuration
        then if there is $C' \in Next(C)$ such that $v(C') = true$
          then $v(C) := true$
          else $v(C) := false$
        else  % $C$ is a universal configuration. %
        if for every configuration $C' \in Next(C)$, $v(C') = true$
          then $v(C) := true$
          else $v(C) := false$
end
```

An ATM accepts an input string x iff $v(C_0) = true$, where C_0 is the initial configuration for x.

Let n be the length of an input to a Turing machine. $DTIME(T(n))$ is the class of languages accepted by deterministic Turing machines within $T(n)$ time. We define $EXPTIME = \bigcup_{i \geq 0} DTIME(2^{n^i})$. $ASPACE(S(n))$ is the class of languages accepted by ATMs within space $S(n)$. The following theorem states that the time complexity of the recognition problem for linear space-bounded ATMs is exponential in terms of deterministic Turing machine.

Theorem 7[1] $EXPTIME = \bigcup_{i \geq 0} ASPACE(n^i)$ □

Theorem 8 The membership problem for RLFGs which have at least one ε-production is $EXPTIME$-hard.

Proof Sketch. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F, U)$ be a one-tape linear space ATM. We assume that the length of the tape is exactly n, where n is the length of the input string. Let $w = x_1 x_2 ... x_n$ be an input for M, where $x_i \in \Gamma$ for all $i, 1 \leq i \leq n$. We will construct an RLFG $G(M,w) = (\Sigma, \Gamma, S, FN, FV, PR, AR)$ such that M accepts w iff $w \in L(D(M,w))$. Without loosing generality, we assume that $\Sigma = \{a, b\}$. $G(M,w)$ is constructed as follows.
$\Gamma = \{[q] | q \in Q\} \cup \{S, A_1, A_2, ..., A_{n-1}, B_1, B_2, ..., B_n\}$,
$FN = \{L, R, H, T, X, ACC\}$,
$FV = \{\$, 0, 1\} \cup \Sigma$,
$PR = \emptyset$ and
AR is constructed based on δ as follows :

1. Let
$$\delta(q, x) = \{(q_1, y_1, d_1), (q_2, y_2, d_2), ..., (q_m, y_m, d_m)\}.$$
If $q \in Q$ then add the following rule to R :
$$[q] \rightarrow ([q_1], E_1)([q_2], E_2)...([q_m], E_m),$$
else ($q \in Q - U$) add the following rules to R :
$$[q] \rightarrow ([q_1], E_1),\ [q] \rightarrow ([q_2], E_2), ..., [q] \rightarrow ([q_m], E_m).$$
For each i $(1 \leq i \leq m)$, E_i is constructed as follows : if $d_i = R$ then E_i contains the following rules :
   (a) $(\uparrow\ C) = x$
   (b) $((\downarrow\ L)\ H) = y_i$
   (c) $((\downarrow\ L)\ T) = (\uparrow L)$
   (d) $(\downarrow\ C) = ((\uparrow\ R)\ H)$
   (e) $(\downarrow\ R) = ((\uparrow\ R)\ T)$

(f) $(\downarrow\ ACC) = 1$

(g) $(\uparrow\ ACC) = 1$

else ($d_i = L$) then E_i contains the following rules :

(a) $(\uparrow\ C) = x$

(b) $(\downarrow\ L) = ((\uparrow\ L)\ T)$

(c) $(\downarrow\ C) = ((\uparrow\ L)\ H)$

(d) $((\downarrow\ R)\ H) = y_i$

(e) $((\downarrow\ R)\ T) = (\uparrow R)$

(f) $(\downarrow\ ACC) = 1$

(g) $(\uparrow\ ACC) = 1$

2. For $q \in F$, add the following rule to R :

$$[q] \rightarrow (\epsilon, \{(\uparrow\ ACC) = 1\}).$$

3. Add the following rules to R :

(a) $S \rightarrow (A, \{\uparrow=\downarrow\})([q_0], \{\uparrow=\downarrow\})$

(b) $A_{n-1} \rightarrow (B_n, \{((\uparrow\ R)\ H) = (\downarrow\ X),\ ((\uparrow\ R)\ T) = \$\})$
$(B_{n-1}, \{(\uparrow\ C) = (\downarrow\ X),\ (\uparrow\ L) = \$\})$

(c) $A_k \rightarrow (A_{k+1}, \{((\uparrow\ R)\ H) = (\downarrow\ C),\ ((\uparrow\ R)\ T) = (\downarrow\ R)\})$
$(B_k, \{(\uparrow\ C) = (\downarrow\ X),\ (\uparrow\ L) = \$\})$ for all $1 \leq k \leq n-2$

(d) $B_k \rightarrow (x_k,\ \{(\uparrow\ X) = x_k\})$ for all $1 \leq k \leq n$

□

Corollary 9 The membership problem for LFGs which have at least one ε-production is *EXPTIME*-hard. □

4 Conclusion

We summarize the results in this paper in the following table.

	CFG	LFG	CSG
Emptiness	*PTIME* [2]	undecidable	undecidable [2]
Membership	*PTIME* [2]	*EXPTIME*-hard*	*NSPACE*(n) [2]

Remarks * In the case when the LFG has at least one ε-production.

References

[1] Chandra, A. K., Kozen, D. C., and Stockmeyer, L. J. : Alternation, *JACM, 28* (1981), pp.114-133.

[2] Hopcroft, J. E., and Ullman, J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).

[3] Kaplan, R. M., and Bresnan, J. : Lexical Functional Grammar: A Formal system for Grammatical Representation, In : J. Bresnan (Ed.) *The Mental Representation of Grammatical Relations*, MIT Press, Camblidge, Massachusetts (1982).

[4] Pinker, S. : A Theory of the Acquisition of Lexical Interpretive Grammars, In : J. Bresnan (Ed.) *The Mental Representation of Grammatical Relations.* MIT Press, Camblidge, Massachusetts (1982).