KYBERNETIKA — VOLUME *28* (1992), NUMBER 1, PAGES 50 – 61

FORMAL TRANSLATION DIRECTED BY LR PARSING

Bořivoj Melichar

The notion of the syntax-directed translation was a highly influential idea in theory of the formal translation. Models for the description of the formal translations are syntax-directed translation schemes. The special case of syntax-directed translation schemes are simple syntax-directed translation schemes, which can be written in the form of translation grammars. It is possible for an arbitrary translation described by a translation grammar with $LL(k)$ input grammar to create one-pass translation algorithm by a simple extension of the algorithm of a syntax analysis for $LL(k)$ grammars. Similar approach for an $LR(k)$ grammar led to the result that it is possible to perform an one-pass formal translation during $LR(k)$ analysis only in that case when the translation grammar has a postfix property. In this paper the construction of the algorithm is studied, which can, for a particular class of translation grammars (called $LR(k)$ R-translation grammars), perform one pass formal translation. The basic idea discussed in this paper is the following: It is possible to make an extension of the algorithm of the syntax analysis for $LR(k)$ grammars in such a way, that the output of output symbols can be performed not only as a part of the operation reduction but also as a part of the operation shift.

1. INTRODUCTION

The notion of the syntax-directed translation introduced by Irons ([5], [6]) was a highly influential idea in theory of the formal translation. Mathematical models of the syntax-directed translation have been developed and studied in [1], [2], [4], [8], [10] and [12]. Models for the description of the formal translations are syntax-directed translation schemes. The special case of syntax-directed translation schemes are simple syntax-directed translation schemes, which can be written in the form of translation grammars.

Parallel to the development of methods of the formal description of the translation, principles for implementation of algorithms of the syntax-directed translation were researched. Already in 1968, Lewis and Stearns [8] have shown that it is possible for an arbitrary translation described by a translation grammar with $LL(k)$ input grammar to create one-pass translation algorithm by a simple extension of the algorithm of a syntax analysis for $LL(k)$ grammars.

Similar approach for an $LR(k)$ grammar led to the result that it is possible to perform an one-pass formal translation during $LR(k)$ analysis only in that case when the translation grammar has a postfix property, which means that output symbols are placed only at the ends of the right-hand sides of the grammar rules. It means that the output of output symbols is made only if the end of the rule

is discovered. This means, from the point of view of the syntax analyzer, that the output is performed as a part of the operation reduction of the syntax analyzer. The restriction of the translation grammar rules mentioned has led to a development of various transformations of translation grammars into grammars having postfix property (cf. [8], [9] and [12]) and to a creation of the four pass model of the formal translator (cf. [1]). Others (cf. [7]) remarked that almost all bottom up syntax analyzers contain elements of the top down methods, which can be used in the process of extension of the syntax analyzer to the algorithm of the formal translation.

In this paper the construction of the algorithm is studied, which can, for a particular class of translation grammars (called $LR(k)$ R-translation grammars), perform one pass formal translation. The class of $LR(k)$ R-translation grammars is a superset of $LR(k)$ postfix translation grammars.

The basic idea discussed in this paper is the following: It is possible to make an extension of the algorithm of the syntax analysis for $LR(k)$ grammars in such way that the output of output symbols can be performed not only as a part of the operation reduction but also as a part of the operation shift.

2. NOTATION

Alphabet is a finite nonempty set of symbols. The set of strings of symbols from the alphabet A including empty string (e) is denoted by A^*. A formal language L over an alphabet A is a subset of A^*, $L \subset A^*$.

A context-free grammar is a quadruple $G = (N, T, P, S)$, where N is a finite set of nonterminal symbols, T is a finite set of terminal symbols, $T \cap N = \emptyset$, S is the start symbol, P is a finite set of rules of the form $A \to \alpha$, $A \in N$, $\alpha \in (N \cup T)^*$. The symbol $\Rightarrow$ is used for the derivation relation. For any $\alpha, \beta \in (N \cup T)^*$, $\alpha \Rightarrow \beta$ if $\alpha = \gamma_1 A \gamma_2, \beta =$
$= \gamma_1 \gamma_0 \gamma_2$ and $A \to \gamma_0 \in P$, where $A \in N$ and $\gamma_0, \gamma_1, \gamma_2 \in (N \cup T)^*$. Symbols $\Rightarrow^k$, $\Rightarrow^+$, $\Rightarrow^*$ are used for k-power, transitive, transitive and reflexive closure of $\Rightarrow$, respectively. The symbol $\Rightarrow_{rm}$ is reserved for the rightmost derivation, e. g. $\gamma_1 A \gamma_2 \Rightarrow_{rm} \gamma_1 \alpha \gamma_2$ if $\gamma_2 \in T^*$. The sentential form α is a string which can be derived from S, $S \Rightarrow^* \alpha$. The sentential form α for $S \Rightarrow^*_{rm} \alpha$ is called the right sentential form. The formal language generated by the grammar $G = (N, T, P, S)$ is the set of strings $L(G) = \{w : S \Rightarrow^* w, w \in T^*\}$.

A derivation tree may be viewed as a graphical representation for a derivation. Each interior node of it is labeled by some nonterminal symbol A and the children of the node are labeled, from left to right, by the symbols in the right hand side of the rule by which this A was replaced in the derivation. The leaves of the derivation tree are labeled by empty strings or terminal symbols and, if read from left to right, they constitute a string derived by the grammar. The derivation tree will be treated as an expression of the syntactic structure of the derived string.

By T^{*k} we shall denote the set $T^{*k} = \{x : x \in T^*,\ |x| \leq k,\ k > 0\}$, where the length of string $x \in T^*$ is denoted by $|x|$. We define the sets $\mathrm{FIRST}_k(\alpha)$ for $\alpha \in (N \cup T)^*$ and $\mathrm{FOLLOW}_k(A)$ for $A \in N$, as follows.
$\mathrm{FIRST}_k(\alpha) = \{x \in T^* : \alpha \Rightarrow^* x\beta$ and $|x| = k$, or $\alpha \Rightarrow^* x$ and $|x| < k\}$,
$\mathrm{FOLLOW}_k(A) = = \{x \in T^* : S \Rightarrow^*_{rm} \alpha A\beta$ and $x \in \mathrm{FIRST}_k(\beta)\}$.

3. TRANSLATION GRAMMARS

A formal translation Z is a relation $Z \in A \times B$, where A and B are sets of strings. A and B are sets of input and output strings, respectively.

A context-free translation grammar is a context-free grammar, in which the set of the terminal symbols is divided into two disjoint subsets, the set of input symbols and the set of output symbols.

Definition 1. A context-free translation grammar is a 5-tuple $TG = (N, T, D, R, S)$, where

N is the set of nonterminal symbols,
T is the set of input symbols,
D is the set of output symbols,
R is the set of rules of the form $A \to \alpha$, where $A \in N$, $\alpha \in (N \cup T \cup D)^*$,
S is the starting symbol.

The input homomorphism h_i^{TG} and the output homomorphism h_o^{TG} from $(N \cup T \cup D)^*$ to $(N \cup T \cup D)^*$ are defined in the following way:

$$h_i^{TG}(a) = \begin{cases} a & \text{for } a \in T \cup N \\ e & \text{for } a \in D \end{cases} \qquad h_0^{TG}(a) = \begin{cases} a & \text{for } a \in T \\ e & \text{for } a \in D \cup N \end{cases}$$

The derivation in the translation grammar TG is denoted by $\Rightarrow$ and called the translation derivation. The formal translation defined by the translation grammar TG is the set $Z(TG) = \{(h_i^{TG}(w), h_o^{TG}(w)) : S \Rightarrow^* w,\ w \in (T \cup D)^*\}$.

The input grammar of the translation grammar TG is the context-free grammar $G =$
$= (N, T, R_i, S)$, where $R_i = \{A \to h_i^{TG}(\alpha) : A \to \alpha \in R\}$.

Note. The upper index TG is omitted if no confusion arises.

4. R–TRANSLATION GRAMMARS

As stated above, it is possible to extend the LR parser to perform an output of a symbol as a part of the operation reduce. The basic idea described below is a possibility to extend the LR parser in order to perform the output of symbols as a part of the operation shift as well. Let us consider a simple case when a rule of the translation grammar has the form $A \to \alpha x a \beta$, where x is the string of output symbols, a is the input symbol, α, β are strings of input, output and nonterminal symbols.

In such a case, it is possible to add the string x to the output string during the shift of the symbol a.

Definition 2. A translation grammar TG is called R-translation grammar if the strings of output symbols appear at the ends of the right-hand sides of the rules and/or immediately in front of input symbols.

5. $LR(k)$ R–TRANSLATION GRAMMARS

Now we can demonstrate that an extended LR parser can perform the translation, if it is possible for every shift operation to determine unambiguously the string of output symbols, which may be added to the output string.

Definition 3. A translation $LR(k)$ item for the translation grammar $TG = (N, T, D, R, S)$ is the object of the form $[A \to \alpha \bullet \beta, x, w]$ where $A \to \alpha\beta$ is a rule of the input grammar for the translation grammar TG, $x \in D^*$, $w \in T^{*k}$, $k \geq 0$.
For $k = 0$ an $LR(0)$ translation item will be written in the form $[A \to \alpha \bullet \beta, x]$.

The following algorithm constructs the collection of sets of the translation $LR(k)$ items for given translation grammar TG.

Algorithm 1. Construction of the collection of sets of $LR(k)$ translation items.

Input: R-translation grammar $TG = (N, T, D, R, S)$, $k \geq 0$.

Output: Collection P of sets of $LR(k)$ translation items for the translation grammar TG.

Method:

Step 1. Construct an augmented grammar $TG' = (N \cup \{S'\}, T, D, R \cup \{S' \to S\}, S')$.

Step 2. Construct the initial set of $LR(k)$ translation items in the following way:

(a) $\# := \{[S' \to \bullet S, e, e]\}$.

(b) If $[A \to \bullet B\beta, e, u] \in \#$, $B \in N$ and $B \to \gamma \in R$, then $\# := \# \cup \{[B \to \bullet h_i(\gamma), y, v] : y \in D^*$ is the longest prefix of γ containing output symbols only, $v \in \mathrm{FIRST}_k(h_i(\beta)u)\}$.

(c) Repeat the step (b) while new items can be inserted into the set $\#$.

(d) $P := \{\#\}$, $\#$ is the initial set.

Step 3. If the set M_i of $LR(k)$ translation items has been constructed, construct for each symbol $X \in (N \cup T)$, which is in some $LR(k)$ item in M_i just behind the dot, a new set of $LR(k)$ translation items X_j, where $j = \max(k) + 1$ for $X_k \in P$ or $j = 1$ for $X_k \notin P$, in the following way:

(a) $X_j := \{[A \to \alpha X \bullet \beta, y, u] : [A \to \alpha \bullet X\beta, x, u] \in M_i$, $y \in D^*$ is the string of output symbols from the right hand side of the translation grammar rule corresponding to the rule $A \to \alpha X\beta$ between symbol X and string β.$\}$,

(b) If $[A \to \alpha \bullet B\beta, e, u] \in X_j$, $B \in N$ and $B \to \gamma \in R$, then $X_j := X_j \cup \{[B \to \bullet h_i(\gamma), y, v]$, $y \in D^*$ is the longest prefix of γ containing only output symbols, $v \in \mathrm{FIRST}_k(h_i(\beta)u)\}$.

(c) Repeat the step (b) while new items can be inserted into the set X_j.

(d) $P := P \cup \{X_j\}$.

Step 4. Repeat Step 3 for all sets M_i, while new sets can be added into the collection P.

This algorithm constructs the collection of sets of $LR(k)$ translation items for a given translation grammar. This collection differs from the collection of sets of $LR(k)$ items for the input grammar. Each of its items contains a string of output symbols.

There is a string of output symbols y in the item with the dot at the end of the right hand side of the rule. The string y is a string of output symbols from the end of the rule in question. Such a situation means that the operation reduce will be performed during the translation and the string y will be added to the output string.

There is also a string x of output symbols in the item with the dot just in front of an input symbol. In this case the string x is the string of output symbols from the rule in question placed in front of the input symbol behind the dot. This means that for the rule of the translation grammar of the form $A \to \alpha x a \beta$ the constructed item for some $u \in T^{*k}$ is $[A \to h_i(\alpha) \bullet a h_i(\beta), x, u]$ where $x \in D^*$, $a \in T$, $\alpha, \beta \in (N \cup T \cup D)^*$ and α does not end with the output symbol.

The existence of such an item in some set of $LR(k)$ translation items means that the operation shift will be performed during the translation and the string x will be added to the output string. In order to select the output string x unambiguously, there must not be, in the same set of $LR(k)$ translation items, two different items with different output strings, with the same input symbol behind the dot, and with the same lookahead strings from $\mathrm{FIRST}_k(a h_i(\beta) u)$.

Definition 4. We say that in the collection P of $LR(k)$ translation items there is a translation conflict, if in some set of P two items are of the form

$$[A \to \alpha \bullet a\beta, x, u]$$
$$[B \to \gamma \bullet b\delta, y, v]$$

for $x \neq y$ and $\mathrm{FIRST}_k(a\beta u) \cap \mathrm{FIRST}_k(b\delta v) \neq \emptyset$.

Definition 5. An R-translation grammar TG is called an $LR(k)$ R-translation grammar, if the input grammar of TG is an $LR(k)$ grammar and there is no translation conflict in any set of $LR(k)$ translation items of the collection P for TG.

6. ALGORITHM OF THE FORMAL TRANSLATION

For the $LR(k)$ R-translation grammar translation can be performed using the algorithm, which is obtained by the following modification of the LR parser.

Step 1. During the operation reduce, add the string of output symbols to output string from the $LR(k)$ item corresponding to the reduce operation performed.

Step 2. During the operation shift, add the string of output symbols to output string from the $LR(k)$ item corresponding to the shift operation performed.

Strings of output symbols can be inserted into the corresponding items of the action table of the LR parser. The resulting table will be called the translation table.

Algorithm 2. Construction of the translation table for a $LR(k)$ R-translation grammar.

Input: $LR(k)$ R-translation grammar $TG = (N, T, D, R, S)$ and a collection P of sets of $LR(k)$ translation items for $LR(k)$ R-translation grammar TG.

Output: Translation table p for the translation grammar TG.

Method: Translation table has rows denoted by the sets of items from P, columns are denoted by the elements of the set T^{*k}.

Step 1. $p(M_i, u) = shift(x)$, if $[A \to \alpha \bullet \beta, x, v] \in M_i$, $\beta \in T(N \cup T)^*$, $u \in \mathrm{FIRST}_k(\beta v)$, $x \in D^*$,
Step 2. $p(M_i, u) = reduce\ j(x)$, if $j \geq 1$ and $[A \to h_i(\alpha) \bullet, x, u] \in M_i$, $A \to \alpha$ is jth rule in R, $u \in T^{*k}$, $x \in D^*$,
Step 3. $p(M_i, e) = accept$, if $[S' \to S \bullet, e, e] \in M_i$,
Step 4. $p(M_i, u) = error$ in all other cases.

Note. The goto table may be constructed in the same way as the one for the LR parser (see [3]).

Algorithm 3. Formal translation for $LR(k)$ R-translation grammar.

Input: The translation table p and the goto table g for the translation grammar $TG =$
$= (N, T, D, R, S)$, input string $x \in T^*$, $k \geq 0$.

Output: Output string y in case that for $x \in L(G_i)$, $(x, y) \in Z(TG)$, otherwise error signalisation.

Method: The symbol # is an initial symbol in the pushdown store. Repeat Steps 1, 2 and 3 until accept or error appears. Symbol Y is on the top of the pushdown store.

Step 1. Fix the string of first k symbols from the unused part of the input string and denote it by u.
Step 2. (a) If $p(X, u) = shift(x)$, read one input symbol, add the string x to the output string and go to Step 3.
(b) If $p(X, u) = reduce\ i(x)$, pop from the pushdown store the same number of symbols as is the number of input and nonterminal symbols at the right-hand side of the ith rule $(i)A \to \alpha$ and add string x to the output string. Go to Step 3.
(c) If $p(X, u) = accept$, finish the translation; then the output string is the translation of the input string, provided that the input string is read completely, otherwise finish the translation by an error signalisation.

(d) If $p(X,u) = error$, finish the translation by an error signalisation.

Step 3. If W is a symbol which may be pushed to the pushdown store (the read symbol in 2(a) or the left hand side of the rule used for the reduction in 2(b)) and Y is the symbol at the top of the pushdown store, then:

(a) If $g(Y,W) = M$, then push M at the top of the pushdown store and repeat the algorithm from the step 1.

(b) If $g(Y,W) = error$, finish the translation by an error signalisation.

The configuration of the algorithm is the triple (α, x, y), where
α is the content of the pushdown store,
x is the unused part of the input string,
y is the part of the output string already created.
The initial configuration is a triple $(\#, x, e)$, the accepting configuration is a triple $(\#M_i, e, y)$, where M_i is the symbol at the top of the pushdown store, and it holds for M_i that $p(M_i, e) = accept$.

Example. Let us have translation grammar
$TG = (\{A,B\},\{a,b\},\{x,y\},R,A)$, where R contains the rules:

(1) $A \to aAy$ (2) $A \to B$
(3) $B \to xbB$ (4) $B \to x$

This grammar describes the translation $Z(TG) = \{(a^i b^j, x^{j+1} y^i) : i,j \geq 0\}$. Let us construct the collection of sets of $LR(1)$ translation items for the grammar TG.

$$
\begin{aligned}
\# &= \{[A' \to \bullet A,\ e,\ e], [A \to \bullet aA,\ e,\ e], [A \to \bullet B,\ e,\ e], [B \to \bullet bB,\ x,\ e], [B \to \bullet,\ x,\ e]\} \\
A_1 &= \{[A' \to A\bullet,\ e,\ e]\} \\
a_1 &= \{[A \to a\bullet A,\ e,\ e], [A \to \bullet aA,\ e,\ e], [A \to \bullet B,\ e,\ e], [B \to \bullet B,\ x,\ e], [B \to \bullet,\ x,\ e]\} \\
B_1 &= \{[A \to B\bullet,\ e,\ e]\} \\
b_1 &= \{[B \to b\bullet B,\ e,\ e],\ [B \to \bullet bB,\ x,\ e],\ [B \to \bullet,\ x,\ e]\} \\
A_2 &= \{[A \to aA\bullet,\ y,\ e]\} \\
B_2 &= \{[B \to bB\bullet,\ e,\ e]\}
\end{aligned}
$$

The following table is the translation and goto table. Symbols S and A stand for operations shift and accept, respectively. The reduction by the rule number (i) is denoted by R_i.

	a	b	e	A	B	a	b
#	S	$S(x)$	$R_4(x)$	A_1	B_1	a_1	b_1
A_1			A				
a_1	S	$S(x)$	$R_4(x)$	A_2	B_1	a_1	b_1
B_1			R_2				
b_1		$S(x)$	$R_4(x)$		B_2		b_1
A_2			$R_1(y)$				
B_2			R_3				

Algorithm 3 performs the translation of the input string aab in the following way:

$$\begin{array}{lll}
(\#, aab, e) & \vdash & (\#a_1, & ab, e) \\
& \vdash & (\#a_1\,a_1, & b, e) \\
& \vdash & (\#a_1\,a_1\,b_1, & e, x) \\
& \vdash & (\#a_1\,a_1\,b_1\,B_2, & e, xx) \\
& \vdash & (\#a_1\,a_1\,B_1, & e, xx) \\
& \vdash & (\#a_1\,a_1\,A_2, & e, xx) \\
& \vdash & (\#a_1\,A_2, & e, xxy) \\
& \vdash & (\#A_1, & e, xxyy)
\end{array}$$

Main theorem. Algorithm 3 of the formal translation for $LR(k)$ R-translation grammar TG creates, for each input string $x \in L(G_i)$, where G_i is the input grammar of translation grammar TG, an output string y such that $(x, y) \in Z(TG)$.

P r o o f. Algorithm 3 is an extension of an LR parser, which means that it constructs the reverse of the rightmost derivation of the input string x and, if this derivation does not exist, it produces an error signalisation. Therefore we have to prove the fact, that for an input string $x \in L(G_i)$ the output string y is produced such that $(x, y) \in Z(TG)$.

The proof will be made by the induction on the length of the rightmost derivation of the input string.

First the following claim has to be proved:

($\star$) If for some $A \in N$ a derivation $A \Rightarrow^n w$ exists in TG such that $x = h_i(w)$, $y = h_o(w)$, then Algorithm 3 performs the sequence of moves $(\alpha, x, \beta) \vdash^* (\alpha A', e, \beta y)$ for some string α of pushdown symbols, $\beta \in D^*$, where A' is the pushdown symbol corresponding to A.

1. For $n = 1$ the derivation has the form $A \Rightarrow w$ and in R there is the rule $A \to y_1 a_1 y_2 a_2 \cdots y_k a_k y_{k+1}$, where $k \geq 0$, $y_1, y_2, \cdots, y_k, y_{k+1} \in D^*$, $a_1, a_2, \cdots, a_k \in T$, $h_i(w) = a_1 a_2 \cdots a_k$, $h_o(w) = y_1 y_2 \cdots y_k y_{k+1}$. In this case the collection P of the sets of translation $LR(k)$ items contains sets $b, a'_1, a'_2, \cdots, a'_k$ and these sets contain the following items:

$$\begin{array}{l}
[A \to \bullet\, a_1 a_2 \cdots a_k, y_1, u] \in b, \\
[A \to a_1 \bullet\, a_2 \cdots a_k, y_2, u] \in a'_1, \\
\cdots \\
[A \to a_1 a_2 \cdots \bullet\, a_k, y_k, u] \in a'_{k-1}, \\
[A \to a_1 a_2 \cdots a_k \bullet, y_{k+1}, u] \in a'_k
\end{array}$$

for some lookahead string $u \in T^{*k}$.
Algorithm 3 performs for some string of pushdown symbols α the following sequence

of moves

$$\begin{aligned}(\alpha, a_1 a_2 \dots a_k, \beta) &\vdash (\alpha a'_1, a_2 \dots a_k, \beta y_1)\\ &\vdash (\alpha a'_1 a'_2, a_3 \dots a_k, \beta y_1 y_2)\\ &\vdash \dots\\ &\vdash (\alpha a'_1 a'_2 \dots a'_k, e, \beta y_1 y_2 \dots y_k)\\ &\vdash (\alpha A', e, \beta y_1 y_2 \dots y_k y_{k+1})\end{aligned}$$

Therefore the claim $(\star)$ is true for $n = 1$.

2. Suppose that the claim $(\star)$ is true for all $m < n$. The rightmost derivation of the length n has the form

$$\begin{aligned}A \quad &\Rightarrow z_{1,1}a_{1,1}z_{1,2}a_{1,2}\cdots z_{1,i_1}a_{1,i_1}B_1 z_{2,1}a_{2,1}z_{2,2}a_{2,2}\cdots z_{2,i_2}a_{2,i_2}B_2\cdots\\ &\quad \cdots B_k z_{k+1,1}a_{k+1,1}z_{k+1,2}a_{k+1,2}\cdots z_{k+1,i_{k+1}}a_{k+1,i_{k+1}}v\\ &\Rightarrow^{m_k} z_{1,1}a_{1,1}z_{1,2}a_{1,2}\cdots z_{1,i_1}a_{1,i_1}B_1 z_{2,1}a_{2,1}z_{2,2}a_{2,2}\cdots z_{2,i_2}a_{2,i_2}B_2\cdots\\ &\quad \cdots w_k z_{k+1,1}a_{k+1,1}z_{k+1,2}a_{k+1,2}\cdots z_{k+1,i_{k+1}}a_{k+1,i_{k+1}}v\\ &\Rightarrow^{m_{k-1}} \dots\\ &\Rightarrow^{m_2} z_{1,1}a_{1,1}z_{1,2}a_{1,2}\cdots z_{1,i_1}a_{1,i_1}B_1 z_{2,1}a_{2,1}z_{2,2}a_{2,2}\cdots z_{2,i_2}a_{2,i_2}w_2\cdots\\ &\quad \cdots w_k z_{k+1,1}a_{k+1,1}z_{k+1,2}a_{k+1,2}\cdots z_{k+1,i_{k+1}}a_{k+1,i_{k+1}}v\\ &\Rightarrow^{m_1} z_{1,1}a_{1,1}z_{1,2}a_{1,2}\cdots z_{1,i_1}a_{1,i_1}w_1 z_{2,1}a_{2,1}z_{2,2}a_{2,2}\cdots z_{2,i_2}a_{2,i_2}w_2\cdots\\ &\quad \cdots w_k z_{k+1,1}a_{k+1,1}z_{k+1,2}a_{k+1,2}\cdots z_{k+1,i_{k+1}}a_{k+1,i_{k+1}}v\end{aligned}$$

where $v, z_{j,l} \in D^*$, $a_{j,l} \in T$, $x_j = h_i(w_j), y_j = h_o(w_j)$, $i_k \geq 0$ for $j = 1, 2, \cdots, k+1$, $l = 1, 2, \cdots, i_j$, $k \geq 0$.
In this case the collection P of sets of $LR(k)$ translation items contains sets

$$b, a'_{1,1}, a'_{1,2}, \cdots, a'_{1,i_1}, B'_1, a'_{2,1}, a'_{2,2}, \cdots, a'_{2,i_2}, B'_2, \cdots, B'_k, a'_{k+1,1}, a'_{k+1,2}, \cdots, a'_{k+1,i_{k+1}}$$

and these sets contain the following items:

$$
\begin{aligned}
&\big[A \to \bullet\, a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, z_{1,1}, u\big] \in b\\
&\big[A \to a_{1,1}\bullet\, a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, z_{1,2}, u\big] \in a'_{1,1}\\
&\ldots\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}\bullet\, B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, e, u\big] \in a'_{1,i_1},\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1\bullet\, a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, z_{2,1}, u\big] \in B'_1,\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}\bullet\, a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, z_{2,2}, u\big] \in a'_{2,1},\\
&\ldots\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}\bullet\, B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, e, u\big] \in a'_{2,i_2},\\
&\ldots\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots a_{k,i_k}\bullet\, B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, e, u\big] \in a'_{k,i_k},\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_k\bullet\, a_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, z_{k+1,1}, u\big] \in B'_k,\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}\bullet\, a_{k+1,2}\ldots a_{k+1,i_{k+1}}, z_{k+1,2}, u\big] \in a'_{k+1,1},\\
&\ldots\\
&[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \ldots\bullet\, a_{k+1,i_{k+1}}, z_{k+1,i_{k+1}}, u\big] \in a'_{k+1,i_{k+1}-1},\\
&\big[A \to a_{1,1}a_{1,2}\cdots a_{1,i_1}B_1a_{2,1}a_{2,2}\cdots a_{2,i_2}B_2\cdots B_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}\bullet, v, u\big] \in a'_{k+1,i_{k+1}}
\end{aligned}
$$

for some lookahead string $u \in T^{*k}$.

Algorithm 3 performs for some string of pushdown symbols α the following se-

quence of moves:

$$
\begin{aligned}
&\big(\alpha, a_{1,1}a_{1,2}\cdots a_{1,i_1}x_1a_{2,1}a_{2,2}\cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\cdots a_{k+1,i_{k+1}}, \beta\big)\\
\vdash\ &\big(\alpha a'_{1,1}, a_{1,2}\cdots a_{1,i_1}x_1a_{2,1}a_{2,2}\cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\cdots a_{k+1,i_{k+1}}, \beta z_{1,1}\big)\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}, \cdots a_{1,i_1}x_1a_{2,1}a_{2,2}\cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\cdots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\big)\\
\vdash\ &\cdots\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}, x_1a_{2,1}a_{2,2}\cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}\big)\\
\vdash^{m_1}\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1, a_{2,1}a_{2,2}\cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1\big)\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}, a_{2,2}\cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\cdots\\
&\qquad\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}\big)\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}, \cdots a_{2,i_2}x_2\cdots x_ka_{k+1,1}a_{k+1,2}\cdots\\
&\qquad\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\big)\\
\vdash\ &\cdots\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}\cdots a'_{2,i_2}, x_2\cdots x_ka_{k+1,1}a_{k+1,2}\cdots\\
&\qquad\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}\big)\\
\vdash^{m_2}\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}\cdots a'_{2,i_2}B'_2, \cdots x_ka_{k+1,1}a_{k+1,2}\cdots\\
&\qquad\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}y_2\big)\\
\vdash\ &\cdots\\
\vdash^{m_k}\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}\cdots a'_{2,i_2}B'_2\cdots B'_k, a_{k+1,1}a_{k+1,2}\cdots\\
&\qquad\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}y_2\cdots y_k\big)\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}\cdots a'_{2,i_2}B'_2\cdots B'_ka'_{k+1,1}, a_{k+1,2}\cdots\\
&\qquad\ldots a_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}y_2\cdots y_kz_{k+1,1}\big)\\
\vdash\ &\big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}\cdots a'_{2,i_2}B'_2\cdots B'_ka'_{k+1,1}a'_{k+1,2}, \cdots\\
&\qquad\ldots a'_{k+1,i_{k+1}}, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}y_2\cdots y_kz_{k+1,1}z_{k+1,2}\big)\\
\vdash\ &\cdots\\
\vdash\ &\Big(\alpha a'_{1,1}a'_{1,2}\cdots a'_{1,i_1}B'_1a'_{2,1}a'_{2,2}\cdots a'_{2,i_2}B'_2\cdots B'_ka'_{k+1,1}a'_{k+1,2}a'_{k+1,i_{k+1}}, e, \beta z_{1,1}z_{1,2}\cdots\\
&\qquad\ldots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}y_2\cdots y_kz_{k+1,1}z_{k+1,2}\cdots z_{k+1,i_{k+1}}\big)\\
\vdash\ &\big(\alpha A', e, \beta z_{1,1}z_{1,2}\cdots z_{1,i_1}y_1z_{2,1}z_{2,2}\ldots z_{2,i_2}y_2\cdots y_kz_{k+1,1}z_{k+1,2}\cdots z_{k+1,i_{k+1}}v\big)
\end{aligned}
$$

Since $m_j < n$, for $j = 1, 2, \cdots, k$ the claim $(\star)$ is true for all $n > 0$.

Thus, we have proved the claim $(\star)$ for an arbitrary rightmost derivation and it holds therefore:
For the rightmost derivation $S \Rightarrow^* w$ in the translation grammar TG, where $x = h_i(w)$, $y = h_o(w)$, Algorithm 3 performs the sequence of moves $(\#, x, e) \vdash^* (\#S', e, y)$ and therefore $(x, y) \in Z(TG)$. □

7. CONCLUSION

A similar approach as for $LR(k)$ R-translation grammars may be used for the definition of $SLR(k)$ and $LALR(k)$ R-translation grammars. The class of $LR(k)$ R-translation grammars does not contain all translation grammars with the $LR(k)$ input grammar. E. g. no translation grammar with output symbols in front of nonterminal symbols belongs to this class.

(Received November 3, 1989.)

REFERENCES

[1] A. V. Aho and J. D. Ullman: Properties of syntax directed translations. J. Comput. System Sci. *3* (1969), 3, 319 – 334.
[2] A. V. Aho and J. D. Ullman: Translation on a context-free grammar. Inform. and Control *19* (1971), 5, 439 – 475.
[3] A. V. Aho and J. D. Ullman: The Theory of Parsing, Translation and Compiling. Vol. 1. Parsing, Vol. 2. Compiling. Prentice-Hall, New York 1971, 1972.
[4] K. Čulík: Well-translatable grammars and Algol-like languages. In: Formal Language Description Languages for Computer Programming (T. B. Steel, ed.), North-Holland, Amsterdam 1966, pp. 76 – 85.
[5] E. T. Irons: A syntax directed compiler for Algol 60. Comm. ACM *4* (1961), 1, 51 – 55.
[6] E. T. Irons: The structure and use of the syntax-directed compiler. In: Annual Review in Automatic Programming 3 (R. Goodman, ed.), Pergamon Press, New York – London 1963, pp. 207 – 227.
[7] J. Král and J. Demner: Parsing as a subtask of compiling. In: Mathematical Foundation of Computer Science 1975 (J. Bečvář, ed., Lecture Notes in Computer Science 32), Springer-Verlag, Berlin – Heidelberg – New York 1975.
[8] P. M. Lewis and R. E. Stearns: Syntax directed transductions. J. Assoc. Comput. Mach. *15* (1968), 3, 465 – 488.
[9] P. M. Lewis, D. J. Rozenkrantz and R. E. Stearns: Compiler Design Theory. Addison-Wesley, London 1976.
[10] L. Petrone: Syntactic mappings of context-free languages. Proc. IFIP Congress 1965, Part 2, pp. 590 – 591.
[11] P. Purdom and C. A. Brown: Semantic routines and $LR(k)$ parsers. Acta Inform. *14* (1980), 4, 229 – 315.
[12] S. Vere: Translation equation. Comm. ACM *13* (1970), 2, 83 – 89.

Doc. Ing. Bořivoj Melichar, CSc., katedra počítač elektrotechnické fakulty ČVUT (Department of Computers, Faculty of Electrical Engineering – Czech Technical University), Karlovo náměstí 13, 121 35 Praha 2. Czechoslovakia.