KYBERNETIKA — VOLUME *30* (1994), NUMBER 1, PAGES 53 – 62

TRANSFORMATIONS OF TRANSLATION GRAMMARS

Bořivoj Melichar

A one-pass translation algorithm may be constructed by an extension of $LR(k)$ parser for some R-translation grammars. An LR parser is possible to extend in such a way that output of output symbols is performed during both basic operations of the parser – shift and reduce. Some transformations are studied which enable to transform some class of translation grammars on R-translation grammars. The most important transformations that can be used for this purpose are those called shaking down and postponing.

1. INTRODUCTION

Translation grammars are one of formal systems for the description of syntax-directed translations. It is possible for an arbitrary translation grammar with $LL(k)$ input grammar to create one-pass translation algorithm by a simple extension of an $LL(k)$ parser [2]. Similar approach is not possible for a translation grammar with an $LR(k)$ input grammar. One-pass translation algorithm may be constructed by an extension of $LR(k)$ parser only for some R-translation grammars [4]. There is a possibility to make an extension of an LR parser in which the output of output symbols can be performed during both basic operations of the parser – shift and reduce. The only condition for the construction of one-pass translation algorithm for a R-translation grammar is that during each operation shift, the string of output symbol can be unambiguously selected.

In this paper transformations are studied that enable to transform some translation grammars to R-translation grammars. The most important transformations that can be used fot this purpose are those called shaking down and postponing.

2. NOTATION

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

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*, and 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 α, $\beta \in (N \cup T)^*$, $\alpha \Rightarrow \beta$, 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 the *k-power*, and for the *transitive*, and *transitive and reflexive* closures of $\Rightarrow$, respectively. The symbol $\Rightarrow_{\mathrm{rm}}$ is reserved for the *rightmost derivation*, i. e., $\gamma_1 A \gamma_2 \Rightarrow_{\mathrm{rm}} \gamma_1 \alpha \gamma_2$, if $\gamma_2 \in T^*$. A *sentential form* is a string α which can be derived from S, $S \Rightarrow^* \alpha$. A sentential form α in $S \Rightarrow^*_{\mathrm{rm}} \alpha$ is called a *right sentential form.* The formal language generated by a grammar $G = (N, T, P, S)$ is the set of strings $L(G) = \{w : S \Rightarrow^* w,\, w \in T^*\}$. Two grammars G_1 and G_2 are *equivalent*, if $L(G_1) = L(G_2)$.

A nonterminal A is *recursive*, if there is a derivation $A \Rightarrow^+ \alpha A \beta$, for some α, $\beta \in (N \cup T)^*$. A nonterminal A is *left-recursive*, if $\alpha = \varepsilon$. If for some $A \in N$, there is a derivation $A_0 \Rightarrow A_1 \alpha_1 \Rightarrow \cdots \Rightarrow A_n \alpha_n \cdots \alpha_1$, $n \geq 1$, with $A_0 = A_n = A$, then the rules $A_i \to A_{i+1} \alpha_{i+1}$ $(0 \leq i < n)$ are called *left-recursive rules* of the grammar.

A *formal translation* Z is a relation $Z \subset A \times B$, where 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 terminal symbols is divided into two disjoint subsets, the set of input symbols and the set of output symbols, respectively.

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 $A \in N$, $\alpha \in (N \cup T \cup D)^*$, and S is the start 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 as follows:

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

For $h \in \{h_i^{TG}, h_o^{TG}\}$ holds $h(\varepsilon) = \varepsilon$, $h(aw) = h(a)\, h(w)$, where $a \in (N \cup T \cup D)$, $w \in (N \cup T \cup D)^*$

The derivation in a translation grammar TG is denoted by $\Rightarrow$, and called the *translation derivation.* The *formal translation* defined by a 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 a translation grammar TG is the context-free grammar

$$G_i = (N, T, R_i, S), \text{ where } R_i = \{A \to h_i^{TG}(\alpha) : A \to \alpha \in R\}.$$

The *output grammar* of a translation grammar TG is the context-free grammar

$$G_o = (N, D, R_o, S), \text{ where } R_o = \{A \to h_o^{TG}(\alpha) : A \to \alpha \in R\}.$$

Note: The superscript TG is omitted when no confusion arises.

A translation grammar TG is called a *postfix translation grammar*, if the strings of output symbols appear only at the ends of right-hand sides of the rules.

Definition 2.1. A translation grammar TG is called an*R-translation grammar*, if the strings of output symbols appear at the ends of right-hand sides of the rules and/or immediately in front of input symbols.

Context-free grammar $G = (N, T \cup D, R, S)$ is the *characteristic grammar* of a translation grammar $TG = (N, T, D, R, S)$. Language $L(G)$ is the *characteristic language* of the translation $Z(TG)$. Sentence $w \in L(G)$ is the *characteristic sentence* of a pair $(x, y) \in Z(TG)$, where $x = h_i(w)$, and $y = h_o(w)$. A derivation tree for some string generated by characteristic grammar G of a translation grammar TG is a *translation tree* for a pair $(h_i(w),\ h_o(w))$ in $Z(TG)$.

3. EQUIVALENCE OF TRANSLATION GRAMMARS

Definition 3.1. Translation grammars TG_1 and TG_2 are equivalent iff $Z(TG_1) = Z(TG_2)$.

Lemma 3.2. Let $TG_1 = (N_1, T_1, D_1, R_1, S_1)$ and $TG_2 = (N_2, T_2, D_2, R_2, S_2)$ be translation grammars with equivalent characteristic context-free grammars $G_1 = (N_1, T_1 \cup D_1, R_1, S_1)$ and $G_2 = (N_2, T_2 \cup D_2, R_2, S_2)$ and $T_1 = T_2$, $D_1 = D_2$. Then translation grammars TG_1 and TG_2 are equivalent.

Proof. It holds for equivalent context-free grammars G_1 and G_2 that $L(G_1) = L(G_2)$, i.e. for each $w \in L(G_1) \Leftrightarrow w \in L(G_2)$. From $T_1 = T_2$ and $D_1 = D_2$ follows $h_i^{TG_1}(w) = h_i^{TG_2}(w) = x$ and $h_o^{TG_1}(w) = h_o^{TG_2}(w) = y$. Therefore $(x, y) \in Z(TG_1) \Leftrightarrow (x, y) \in Z(TG_2)$. □

Lemma 3.2 facilitates the use of transformations known for context-free grammars also for translation grammars. Let us mention for example the "substitution" (see Lemma 2.14 in [1]).

Lemma 3.3. Input and output grammars of equivalent translation grammars are equivalent.

Proof. If for translation grammars TG_1 and TG_2 holds

$$\begin{aligned}
&Z = Z(TG_1) = Z(TG_2) \text{ then}\\
&L(TG_{1i}) = \{x : (x, y) \in Z\} = L(TG_{2i}) \text{ and}\\
&L(TG_{1o}) = \{y : (x, y) \in Z\} = L(TG_{2o})
\end{aligned}$$

□

Example 3.4. The characteristic context-free grammars of equivalent translation grammars need not be equivalent:

$$\begin{aligned}
&TG_1 = (\{S\}, \{a\}, \{x\}, \{S \to xa\}, S) \text{ and}\\
&TG_2 = (\{S\}, \{a\}, \{x\}, \{S \to ax\}, S)
\end{aligned}$$

are equivalent translation grammars because

$$Z(TG_1) = \{(a, x)\} = Z(TG_2).$$

Nevertheless, the characteristic grammars

$G_1 = (\{S\}, \{a, x\}, \{S \to ax\}, S)$ and

$G_2 = (\{S\}, \{a, x\}, \{S \to xa\}, S)$ are not equivalent because

$L(G_1) = \{ax\} \neq L(G_2) = \{xa\}$.

Example 3.5. The reverse of Lemma 3.3 does not hold. For instance for translation grammars

$TG_1 = (\{S\}, \{a, b\}, \{x, y\}, \{S \to ax, S \to by\}, S)$ and

$TG_2 = (\{S\}, \{a, b\}, \{x, y\}, \{S \to ay, S \to bx\}, S)$

holds

$L(TG_{1i}) = L(TG_{2i}) = \{a, b\}$,

$L(TG_{1o}) = L(TG_{2o}) = \{x, y\}$, but

$Z(TG_1) = \{(a, x), (b, y)\} \neq Z(TG_2) = \{(a, y), (b, x)\}$.

4. BASIC TRANSFORMATIONS OF TRANSLATION GRAMMARS

The simplest transformation specific for translation grammars consists of the exchange of adjacent input and output symbols on the right hand side of a rule.

Lemma 4.1. Let $TG = (N, T, D, R, S)$ be a translation grammar which contains the rule $A \to \alpha C x \beta$, where $\alpha, \beta \in (N \cup T \cup D)^*$, $C \in T \cup N$, $x \in D^+$ and if $C \in N$ then C generates strings of input symbols only. Then translation grammar $TG' = (N, T, D, R', S)$, where $R' = (R - \{A \to \alpha C x \beta\}) \cup \{A \to \alpha x C \beta\}$, is equivalent to grammar TG.

P r o o f. First we prove that $Z(TG) \subseteq Z(TG')$. We show, using induction by n, that for any translation derivation of the form $B \Rightarrow^n w$ with length $n > 0$, where $w \in (T \cup D)^*$, in the translation grammar TG, exists translation derivation $B \Rightarrow^* w'$, $w' \in (T \cup D)^*$, in the grammar TG' and it holds $h_i^{TG}(w) = h_i^{TG'}(w')$ and $h_o^{TG}(w) = h_o^{TG'}(w')$ holds.

Let us suppose that $n = 1$. Two cases can occur. In the first case, the same rule is used in both derivations $B \Rightarrow w$ in TG and $B \Rightarrow w'$ in TG'. Therefore $w = w'$ and the assertion holds. In the second case, the rule $A \to \alpha C x \beta$ is used in the derivation of w and $C \in T$. Then the rule $A \to \alpha x C \beta$ is used in the derivation of w' in the grammar TG'. It holds that $h_i(w) = h_i(w') = h_i(\alpha)\, C h_i(\beta)$ and $h_o(w) = h_o(w') = h_o(\alpha)\, x h_o(\beta)$.

Let us suppose that the assertion holds for all derivations shorter than n. Let us have the derivation $B \Rightarrow^n w$ with the length n in the grammar TG. Two cases occur again. Let us treat the first case when $B = A$ and there is following derivation in the grammar TG: $A \Rightarrow \alpha C x \beta \Rightarrow^{n-1} w_1 y x w_2 = w$, $y \in T^*$, $C \Rightarrow^* y$.
The corresponding derivation in the grammar TG' is $A \Rightarrow \alpha x C \beta \Rightarrow^{n-1} w_1' x y w_2' =$

w'. Because derivations $\alpha \Rightarrow^* w_1',\ \beta \Rightarrow^* w_2'$ in the grammar TG' are shorter than n, it holds

$$h_i(w_1yxw_2) = h_i(w_1'xyw_2') = h_i(w_1)yh_i(w_2),$$
$$h_o(w_1yxw_2) = h_o(w_1'xyw_2') = h_o(w_1)xh_o(w_2).$$

If the same rule is used in the first step in the derivation $B \Rightarrow^n w$ in both grammars TG and TG', then the assertion also holds.

The special case is $B = S$. It follows from this that $Z(TG) \subseteq Z(TG')$.

The reverse inclusion $Z(TG') \subseteq Z(TG)$ may be proved in the similar way. From this follows $Z(TG) = Z(TG')$. □

Note. The transformation given by Lemma 4.1 we shall call *postponing* of input symbol. A reverse transformation to the one given by Lemma 4.1 called *advancing* is given by Lemma 4.2.

Lemma 4.2. Let $TG = (N, T, D, R, S)$ be a translation grammar which contains rule $A \to \alpha x C \beta$, where $\alpha, \beta \in (N \cup T \cup D)^*$, $C \in T \cup N$, $x \in D^+$ and if $C \in N$ then C generates strings of input symbols only. Then translation grammar $TG' = (N, T, D, R', S)$, where $R' = (R - \{A \to \alpha x C \beta\}) \cup \{A \to \alpha C x \beta\}$, is equivalent to grammar TG.

The proof of this Lemma is similar to the proof of Lemma 4.1. □

During a transformation of a translation grammar to a postfix translation grammar, we must solve the situation in which a string of output symbols appears inside the right-hand side of a rule. Similarly, during the transformation of a translation grammar on a R-translation grammar the situation is to solve, in which some string of output symbols inside the right-hand side of a rule is followed by a nonterminal symbol which generates at least one string containing output symbols. If such a string of output symbols is preceded at least by one input symbol, it is possible to perform such a transformation by exchange of input and output symbols according to Lemma 4.1. In other cases specific transformation called left and right absorption may be used. Let us first introduce a more general Lemma.

Lemma 4.3. Let $TG = (N, T, D, R, S)$ be a translation grammar, where set R contains rule $A \to \alpha\beta\gamma$ where $\alpha, \beta, \gamma \in (N \cup T \cup D)^*$. Then translation grammar $TG' = (N \cup \{A'\}, T, D, R', S,)$ where $A' \notin N$ and $R' = (R - \{A \to \alpha\beta\gamma\}) \cup \{A \to \alpha A' \gamma,\ A' \to \beta\}$, is equivalent to grammar TG.

Proof. Because a transformation called substitution may be used also for translation grammars (see note after Lemma 3.2), we can substitute β for A' in grammar TG' and we obtain grammar TG. □

Using Lemma 4.3 we define two transformations called left and right absorption.

The left absorption of the output string is the following transformation:
Let $TG = (N, T, D, R, S)$ be a translation grammar, where R contains rule $A \to$

$\alpha\beta x\gamma$, $\alpha, \beta, \gamma \in (N \cup T \cup D)^*$, $x \in D^+$. Then we obtain by the left absorption of the output string x the equivalent translation grammar $TG' = (N \cup \{A'\}, T, D, R', S)$, where $A' \notin N$ and $R' = (R - \{A \to \alpha\beta x\gamma\}) \cup \{A \to \alpha A'\gamma, A' \to \beta x\}$.

The right absorption of the output string can be defined similarly:
Let $TG = (N, T, D, R, S)$ be a translation grammar, where R contains rule $A \to \alpha x\beta\gamma$, $\alpha, \beta, \gamma \in (N \cup T \cup D)^*$, $x \in D^+$. Then we obtain by the right absorption of the output string x the equivalent translation grammar $TG' = (N \cup \{A'\}, T, D, R', S)$, where $A' \notin N$ and $R' = (R - \{A \to \alpha x\beta\gamma\}) \cup \{A \to \alpha A'\gamma, A' \to x\beta\}$.

The transformations of left and right absorption have one common case, the situation in which string β is an empty string. The rule for nonterminal symbol A' has then the form $A' \to x$, where x is a string of output symbols. The rule of input grammar corresponding to this rule has the form $A' \to \varepsilon$. This variant of the transformations in question is called the ε-rule insertion. Such transformation was studied in detail by Purdom and Brown [5]. The main result of this study is the stipulation of conditions when the insertion of ε-rules either destroy or does not destroy the $LR(k)$ property of the input grammar. The special case of left absorption is mentioned by Lewis, Rosenkrantz and Stearns in [3]. In this case string α is an empty string and such transformation is used to transform a translation grammar to the postfix one.

5. THE TRANSFORMATION SHAKING DOWN

Let us discuss a special case of the right absorption. Let string β to which an output string will be absorbed is a single nonterminal symbol. For the clarity of explanation, let us denote the nonterminal symbol created for nonterminal symbol B to which output string x will be absorbed by $[xB]$.

Let us have rule $A \to \alpha x B\gamma$ in translation grammar TG. We obtain following rules after the right absorption of string x into nonterminal symbol B:

$$A \to \alpha[xB]\gamma$$
$$[xB] \to xB.$$

Let there are the following rules for nonterminal symbol B in TG:

$$B \to \delta_1|\delta_2|\cdots|\delta_n.$$

We can substitute the right hand sides of these rules into the rule $[xB] \to xB$ for symbol B and the final rules are:

$$A \to \alpha[xB]\gamma$$
$$[xB] \to x\delta_1|x\delta_2|\cdots|x\delta_n.$$

In the Figure 1 are depicted parts of the translation trees in original grammar TG and in the transformed grammar.

We can see from this picture that string x of output symbols is in the tree for the transformed grammar one level below in comparison with its position in the tree for the original grammar. Therefore we shall call this transformation the "*shaking down*" transformation.

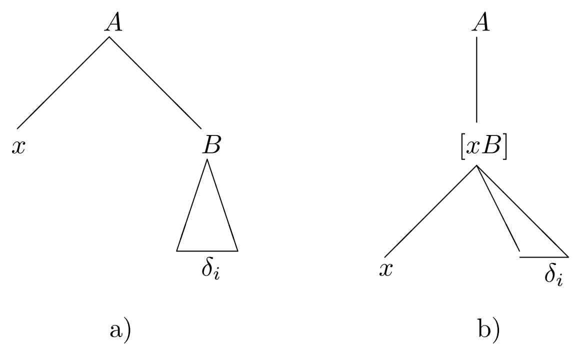

Fig. 1. Parts of translation trees, a) in original grammar, b) in transformed grammar.

Lemma 5.1. Let $TG = (N, T, D, R, S)$ be a translation grammar, where R contains rule $A \to \alpha x B \gamma, \alpha, \gamma \in (N \cup T \cup D)^*$, $B \in N$, and $B \to \delta_1 | \delta_2 | \cdots | \delta_n$ are the only rules in R with nonterminal symbol B on the left-hand sides. Let $TG' = (N \cup \{[xB]\}, T, D, R', S)$, where $R' = (R - \{A \to \alpha x B \gamma\}) \cup \{[xB] \to x\delta_1 | x\delta_2 | \cdots | x\delta_n\}$. Then $Z(TG) = Z(TG')$.

This Lemma follows immediately from Lemmas 3.2, 4.3, and Theorem on substitution [1].

6. TRANSFORMATION OF TRANSLATION GRAMMAR TO R–TRANSLATION GRAMMAR

The repeated shaking down transformation may be used for the transformation of a translation grammar to R-translation grammar.

Theorem 6.1. Let $TG = (N, T, D, R, S)$ be a translation grammar in which no output symbol appears in front of a left recursive nonterminal symbol in any left recursive rule. Then by repeated shaking down transformation is possible to transform translation grammar TG to R-translation grammar TG' such that $Z(TG) = Z(TG')$.

P r o o f. If we transform translation grammar TG to translation grammar TG' using repeated shaking down transformation and the number of shaking down operations is finite, then grammars TG and TG' are equivalent. Therefore, we must prove that the necessary number of shaking down operations is finite.

Let us have rule $A \to \alpha x C_0 \beta$ in the translation grammar TG, where $A, C_0 \in N$, $x \in D^+$ is the longest string of output symbols, α, $\beta \in (N \cup T \cup D)^*$. If for symbol C_0 exists derivation

$$(*) \quad C_0 \Rightarrow y_1 C_1 \gamma_1 \Rightarrow y_1 y_2 C_2 \gamma_2 \gamma_1 \Rightarrow \cdots$$
$$\Rightarrow y_1 y_2 \cdots y_n C_n \gamma_n \cdots \gamma_2 \gamma_1 \Rightarrow$$

$$\Rightarrow y_1 y_2 \cdots y_n z\delta\gamma_n \cdots \gamma_2\gamma_1,$$

where $C_1, C_2, \ldots, C_n \in N$, $y_1, y_2, \ldots, y_n$, $z \in D^*$, z is the longest string of output symbols, $\gamma_1, \gamma_2, \ldots, \gamma_n$, $\delta \in (N \cup T \cup D)^*$, and none of nonterminal symbols $C_0, C_1, \ldots, C_n$ is left recursive, then all these nonterminal symbols are different and there exist following rules in grammar TG :

$$\begin{aligned}
&C_0 \to y_1 C_1 \gamma_1\\
&C_1 \to y_2 C_2 \gamma_2\\
&C_2 \to y_3 C_3 \gamma_3\\
&\quad\vdots\\
&C_{n-1} \to y_n C_n \gamma_n\\
&C_n \to z\delta.
\end{aligned}$$

We obtain the following rules of grammar TG' using the operation shaking down $n + 2$ times:

$$\begin{aligned}
&A \to \alpha[xC_0]\beta\\
&[xC_0] \to [xy_1C_1]\gamma_1\\
&[xy_1C_1] \to [xy_1y_2C_2]\gamma_2\\
&[xy_1y_2C_2] \to [xy_1y_2y_3C_3]\gamma_3\\
&\quad\vdots\\
&[xy_1y_2 \cdots y_{n-1}C_{n-1}] \to [xy_1y_2 \cdots y_{n-1}y_nC_n]\gamma_n\\
&[xy_1y_2 \cdots y_{n-1}y_nC_n] \to xy_1y_2 \cdots y_{n-1}y_nz\delta,
\end{aligned}$$

in which no output symbols are in front of nonterminal symbols $[xC_0]$ and $[xy_1 \cdots y_kC_k]$ for $k = 1, 2, \ldots, n$. Only the last rule contains output symbols either at the end of its right-hand side if δ is empty string or in front of input symbol if δ is not empty string. It means, that strings x, y_1, $y_2, \ldots, y_n$ have been shaken down to the rule which satisfies the definition of R-translation grammar.

We can construct in grammar TG' derivation:

$$\begin{aligned}
&A \Rightarrow \alpha[xC_0]\beta \Rightarrow \alpha[xy_1C_1]\gamma_1\beta \Rightarrow \cdots\\
&\;\Rightarrow \alpha[xy_1 \cdots y_nC_n]\gamma_n \cdots \gamma_1\beta\\
&\;\Rightarrow \alpha xy_1 \cdots y_nz\delta\gamma_n \cdots \gamma_1\beta.
\end{aligned}$$

The number of derivations of the form $(*)$, which is possible to construct, is finite and therefore it suffices only the finite number of shaking down transformations.

Let us suppose, that in derivation $(*)$ some of nonterminal symbols $C_0, C_1, \ldots, C_n$ is left recursive. It means that $C_i = C_j$ for some $i, j \in \langle 0, n\rangle$, $i \neq j$. Let us suppose that $i < j$. Then there must exist following rules in TG:

$$\begin{aligned}
&C_0 \to y_1 C_1 \gamma_1\\
&\quad\vdots\\
&C_{i-1} \to y_i C_i \gamma_i\\
&C_i \to C_{i+1} \gamma_{i+1}
\end{aligned}$$

$$\begin{aligned}
&\vdots\\
&C_{j-1} \to C_j\gamma_j\\
&C_j \to y_{j+1}C_{j+1}\gamma_{j+1}\\
&\vdots\\
&C_n \to z\delta.
\end{aligned}$$

It means that $y_{i+1}, \ldots, y_j$ from derivation $(*)$ are empty strings. It arises during shaking down transformation for each left recursive rule $C_k \to C_{k+1}\gamma_{k+1}$, where $i \le k < j$ a new rule of the form $[xy_1 \cdots y_iC_k] \to [xy_1 \cdots y_iC_{k+1}]\gamma_{k+1}$. The number of such rules is equal to the number of left recursive rules which is possible to use in a derivation. Their maximal number is given by a product of the number of left recursive rules in the grammar and the number of shaken down strings $xy_1 \cdots y_i$. Because the shaken down strings of the form $xy_1 \cdots y_i$ are generated only by non-left recursive rules, their number is finite. With respect to the finite number of left recursive rules in the grammar, is the number of created rules also finite.

We finish the proof stating that the number of strings of output symbols appearing in front of nonterminal symbols in grammar rules is finite. Therefore it suffices the finite number of shaking down operations for the transformation of translation grammar to R-translation grammar.

Example 6.2. Let us have translation grammar $TG = (\{S, A, B\}, \{a, b\}, \{x, y\}, R, S)$, where R contains rules:

(1) $S \to axAbyBy$

(2) $A \to Aaxy$

(3) $A \to a$

(4) $B \to Bby$

(5) $B \to \varepsilon$.

We obtain, after repeating shaking down and elimination of useless symbols, translation grammar $TG' = (\{S, [xA], [yB]\}, \{a, b\}, \{x, y\}, R', S)$, where $R\prime$ contains rules:

(1) $S \to a[xA]b[yB]y$

(2) $[xA] \to [xA]axy$

(3) $[xA] \to xa$

(4) $[yB] \to [yB]by$

(5) $[yB] \to y$

This grammar is the R-translation grammar.

7. CONCLUSION

We have shown that it is possible to transform each translation grammar to R-translation grammar provided that no output symbols are present in front of left recursive nonterminal symbols. It is possible, after this transformation, to try to implement the formal translation using the algorithm directed by LR parsing [4] and this approach succeeds in such a case when no translation conflict arises. The transformation shaking down and postponing may be incorporated into the process of the construction of LR parser. This approach leads to the construction of translator directed by LR parser [5].

(Received July 19, 1991.)

REFERENCES

[1] 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.

[2] P. M. Lewis and R. E. Stearns: Syntax directed transductions. J. Assoc. Comput. Mach. *15* (1968), 3, 465–488.

[3] P. M. Lewis, D. J. Rozenkrantz and R. E. Stearns: Compiler Design Theory. Addison–Wesley, London 1976.

[4] B. Melichar: Formal translation directed by LR parsing. Kybernetika *28* (1992), 1, 50–61.

[5] B. Melichar: LR Translation Grammars. Research Report No. DC–92–03, Department of Computers, Czech Technical University, Prague 1992.

[6] P. Purdom and C. A. Brown: Semantic routines and $LR(k)$ parsers. Acta Informatica *14* (1980), 4, 229–315.

Doc. Ing. Bořivoj Melichar, CSc., katedra počítačů elektrotechnické fakulty ČVUT (Department of Computers – Czech Technical University), Karlovo nám. 13, 121 35 Praha 2. Czech Republic.