Kybernetika

Jan Vinař

A syntax directed translation algorithm for ALGOL-like languages

*Kybernetika*, Vol. 2 (1966), No. 5, (389)--396

Persistent URL: http://dml.cz/dmlcz/124960

**Terms of use:**

© Institute of Information Theory and Automation AS CR, 1966

Institute of Mathematics of the Academy of Sciences of the Czech Republic provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.

This paper has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://project.dml.cz

KYBERNETIKA ČÍSLO 5, ROČNÍK 2/1966

# A Syntax Directed Translation Algorithm for ALGOL-like Languages

JAN VINAŘ

This paper deals with the translation of ALGOL-like languages using well-translatable grammars in the sense of [1], [2]. The notions and symbolism of these papers will be used without explicit reference. A translation algorithm is presented which, while giving essentially the same results as the translation algorithms of [1], [2], is simpler and better suited for computer usage. An example is used to clarify its function.

## DEFINITIONS AND NOTATION

We shall be concerned with a pair of grammars $G = \langle V_t, V_n, \mathfrak{R} \rangle$ and $G^* = \langle V_t^*, V_n^*, \mathfrak{R}^* \rangle$ generating the languages $L \langle E, S \rangle$ and $L^* = \langle E^*, S^* \rangle$ respectively. We suppose that $G$ is well-translatable into $G^*$, i.e. that there exist two mappings $\tau : V_p \cup V_n \rightarrow V_p^* \cup V_n^*$ and $\Phi : \mathfrak{R} \rightarrow \mathfrak{R}^*$ such that:

(1) $\tau(V_p) \subset V_p^*$, $\tau(V_n) \subset V_n^*$,

(2) if $a_0 :: = b_0 a_1 b_1 \ldots a_k b_k$ is the standard form of a rule $\mathfrak{r} \in \mathfrak{R}$, then the rule $\Phi(\mathfrak{r})$ has the standard form $c_0 :: = d_0 c_1 d_1 \ldots c_k d_k$ where $c_0 :: = \tau(a_0)$ and there exists a permutation $\pi$ of the set $\{1, 2, \ldots, k\}$ such that $c_{\pi(i)} = \tau(a_i)$ for $i = 1, 2, \ldots, k$,

(3) if $x_i \in G(a_i)$, where $a_i \in V_n$ and $y_i \in G^*(c_i)$ where $c_i \in V_n^*$ and $\bar{S}(x_i) = \bar{S}^*(y_i)$, then

$$S(b_0 x_1 b_1 \ldots x_k b_k) = S^*(d_0 y_1 d_1 \ldots y_k d_k) .$$

## THE CANONICAL REDUCTION SEQUENCE [3], [4]

Suppose that $e_0$ is a string, $e_0 \in E$. Then there exists a sequence $(e_0, e_1, \ldots, e_m)$ of strings with the following property: for $i = 1, 2, \ldots, k$ there exists a rule $\mathfrak{r} = (x :: = y) \in \mathfrak{R}$ and two strings $v_i$, $w_i$ (possibly empty) such that $e_{i-1} = v_i y w_i$,

 $e_i = v_i x w_i$. Moreover, $e_m \in V_n$. This sequence is a *derivation* of $e_0$ in $G$). The operation which produces $e_i$ from $e_{i-1}$ is called the *reduction of $e_{i-1}$ to $e_i$ by the rule* $\mathfrak{r}$. If there exists a rule $\mathfrak{r}$ by which we can reduce the string $c$ to string $d$, we say that $c$ is *immediately reducible* to $d$ ($c\varrho_0 d$). The relation $\varrho$ (*reducibility*) is the transitive closure of $\varrho_0$.

A sequence of reductions which produces a derivation of $e_0$ in $G$ is a *reduction sequence* of $e_0$. A pair of reductions of $e_{i-1}$ to $e_i$ by the rule $\mathfrak{r} = x_1 :: = y_1$ and of $e_i$ to $e_{i+1}$ by rule $\mathfrak{r} = x_2 :: = y_2$ is termed *canonical* if $e_{i-1} = v_1 y_1 w_1$, $e_i = v_1 x_1 w_1 = v_2 y_2 w_2$, $e_{i+1} = v_2 x_2 w_2$, $l(v_2) \geqq l(v_1)$.

Among all reduction sequences of $e_0$ there exists at least one in which every two adjacent reductions form a canonical pair. This is a *canonical reduction sequence* of $e_0$.

*Note*: a) The canonical property of the reduction sequence has the following simple meaning: every reduction is applied to the leftmost substring of $e_0$ that can be reduced, i.e. every initial substring is reduced as far as possible before proceeding to the next symbol.

b) A string $e_0$ which has more than one canonical reduction sequence is an *ambiguity* of $L$.

## THE TRANSLATION ALGORITHM

The translation algorithm consists of three parts:

1. *Table construction algorithm* which produces a *table of correspondence* used by the actual translation algorithm. This table is produced once for ever, then the other two parts are used independently.
2. *Syntactic analysis algorithm.*
3. *Actual translation algorithm.*

### Table construction algorithm

a) Let the rules $\mathfrak{r} \in \mathfrak{R}$ and $\Phi(\mathfrak{r}) \in \mathfrak{R}^*$ have the standard forms described in (2). We define the following operation of *bracketing*:

(5) We number the symbols of the right part of $\mathfrak{r}$ (including auxiliary symbols) by the numbers 1, 2, ... *from right to left*, starting with the last symbol, which is thus numbered by 1.

(6) To *bracket* the rule $\Phi(\mathfrak{r})$ we substitute for $c_{\pi(i)}$ the bracket $\{C_n\}$ where $n$ is the number assigned to $a_i$ in (5).

b) The *correspondence table* has two columns and $r$ rows ($r$ being the number of rules in $\mathfrak{R}$). In the left column we place the rules $\mathfrak{r}_1, \mathfrak{r}_2, \ldots, r_r$ in any given order; in the right column, the bracketed rules $\Phi(\mathfrak{r}_1), \ldots, \Phi(\mathfrak{r}_r)$ are placed in the same order.

c) The rules of $\mathfrak{R}$ are divided into three groups:

(I) The rules whose right part consists of one terminal symbol while the right part of $\Phi(\mathfrak{r})$ contains no symbols from $V_a^*$.

(II) The rules $\mathfrak{r}$ such that the right part of the bracketed rule $\Phi(\mathfrak{r})$ contains only brackets $\{C_i\}$ in the same order as the corresponding symbols of the right part of $\mathfrak{r}$.

(III) The rules $\mathfrak{r}$ such that in the right part of the bracketed rule $\Phi(\mathfrak{r})$ either the order of brackets is different from that of the corresponding symbols in the right part of $\mathfrak{r}$ (i.e., the permutation $\pi$ mentioned in (2) is not identical), or symbols from $V_a^*$ are introduced.

**Syntactic analysis algorithm**

We will assume the existence of an *analysis algorithm* with the following properties:

a) for every $e_0 \in E$ it produces a canonical reduction sequence of $e_0$ and the corresponding derivation $(e_0, e_1, \ldots, e_m)$

b) this reduction sequence is produced in exactly $m$ steps, i.e., no steps need be retraced.

(The analysis algorithm of [2], working as it does on the trial – and – error principle, does not satisfy condition b), but it can form a basis for the necessary algorithm. Namely, the reduction sequence can first be obtained in the usual way and stored, then supplied step by step.)

Let us now consider the $i$-th step of the algorithm, in which $e_{i-1}$ is reduced to $e_i$ by the rule $\mathfrak{r}_{p_i}$. We define $q_i = l(v_i) + 1$, $m_i$ as the length of the right part of $\mathfrak{r}_{pi}$.

The reduction sequence produced by the algorithm can thus be characterized by the sequence $(e_0, e_1, \ldots, e_m)$ and/or by the sequence of number pairs $(p_1, q_1)$, $(p_2, q_2), \ldots, (p_n, q_m)$. There is, however, no need to store all of these. (This is one of the differences between our algorithm and the algorithm of [2]). In fact, at any given time only one string $e_j$ and one pair $(p_j, q_j)$ will be stored. Let us consider the $i$-th reduction of the canonical reduction sequence. This reduction reduces $e_{i-1}$ to $e_i$ by the rule $\mathfrak{r}_{p_i}$. For reasons of convenience we will divide this step into two substeps marked $i_a$ and $i_b$. *In step* $i_a$ the string $e_{i-1}$ is searched and the numbers $p_i$, $q_i$ and $m_i$ determined. *In step* $i_b$ the string $e_i$ is formed and *supplants the string* $e_{i-1}$. Thus, after the step $i_a$, the string $e_{i-1}$ and the numbers $p_i$, $q_i$ are available; after step $i_b$, there are available the string $e_i$ and the pair $p_i$, $q_i$.

The string $e_i = x_1x_2 \ldots x_n$ (for $j = 1, 2, \ldots, n$, $x_j \in V$) will be termed *the input string* for the translation algorithm. Since, as we have shown, only one such string is available at any time, there is no danger of confusion.

**The actual translation algorithm**

This algorithm is actuated alternately with the analysis algorithm. It performs mainly the following two functions:

a) *The marking of the input string*, i.e., assigning to each symbol $x_k$ ($k = 1, 2, \ldots, n$) a superscript $(r_k, s_k)$ where $r_k$ and $s_k$ are either both natural numbers or both zeroes. The result of this operation is the *marked input string* $x_1^{(r_1,s_1)} \ldots$ $\ldots x_k^{(r_k,s_k)}$.

b) Introducing and rearranging symbols in the *output string*. This is a sequence $y_1 y_2 y_3 \ldots$ of symbols from $V_t^*$ produced and manipulated by the algorithm. To be precise, we shall regard the output string alternately as a sequence of symbols or as a sequence of "empty places" to put symbols into.

*Putting a string x into the output string b* means filling the necessary number of empty places $y_f y_{f+1} \ldots$ with the symbols of this string ($f$ is the number of the first unfilled place) and updating $f$. Any substring of the output string will be also termed an output string. All of these conventions will hold also for the *temporary storage string* $z_1 z_2 z_3 \ldots$ used by the algorithm.

Two other notions will serve to simplify the description of this algorithm. Let $e_{i-1} = vbw$, $e_i = vaw$ (i.e., $\mathfrak{r}_{p_i} = (a :: = b)$). Suppose that in the marked input string, the symbol $a$ has been assigned the superscript $(r, s)$. Then $y_r \ldots y_s$ is

1. the output string *assigned* to $a$ $(A(a))$,
2. the output string *corresponding* to $b$ $(C(b))$. We shall now give a detailed description of the translation algorithm:

1. $e_0$ is the input string, $l(e_0) = n$. Put $r_1 = s_1 = \ldots = r_n = s_n = 0$. Put $i = 1$.
2. Perform step $i_a$ of the analysis algorithm.
3. If the rule $\mathfrak{r}_{p_i}$ belongs to group (I), put $\tau(x_{q_i})$ into the output string. Perform step $i_b$ of the analysis algorithm and put $r_{q_i} = s_{q_i} = f - 1$. Go to step 6.
4. If the rule $\mathfrak{r}_{p_i}$ belongs to group (II), find

$$a = \min\left(r_{q_i}, \ldots, r_{q_i+m_i-1}\right), \quad b = \max\left(s_{q_i}, \ldots, s_{q_i+m_i-1}\right).$$

   Perform step $i_b$ of the analysis algorithm and put $r_{q_i} = a$, $s_{q_i} = b$. Go to step 6.
5. If the rule $\mathfrak{r}_{p_i}$ belongs to group (III), then the bracketed rule $\Phi(\mathfrak{r}_{p_i})$ has the form

$$\tau(a_0) :: = d_0\{C_{t_1}\}\, d_1 \ldots \{C_{t_k}\}\, d_k\,.$$

   Perform the following operations:

   a) Put $d_0$ into temporary storage.
   b) For $j = 1, 2, \ldots, k$ do the following:
      ba) Put $A(x_{q_i+m_i-t_j})$ into temporary storage.
      bb) Put $d_j$ into temporary storage.
   c) Put $f = a = \min\left(r_{q_i+m_i+t_1}, \ldots, r_{q_i+m_i-t_k}\right)$. Put the temporary storage string into the output string.
   d) Perform step $i_b$ of the analysis algorithm and put

$$r_{q_i} = a, \quad s_{q_i} = f - 1\,.$$

6. If the syntactical analysis has been completed, then terminate. $A(e_m)$ is the required translation. Otherwise raise $i$ by 1 and go to step 2.

The operations described are really simple but rather hard to visualize from the formal description. The following example will serve to illustrate the process of translation.

The languages $L$ — the usual arithmetical expressions with dyadic numbers (containing only "+" and "×") and $L^*$ — the corresponding expressions in the Lukasiewicz notation, but read from right to left — are generated, respectively, by the grammars $G$ and $G^*$ defined as follows [2]:

$$V_p = \{0, 1, +, \times\} \qquad V_p^* = V_p$$
$$V_a = \{[, ]\} \qquad V_a^* = \{;\}$$
$$V_n = \{p, q, r, s\} \qquad V_n^* = \{a, b, c, d\}$$

| | |
|---|---|
| 1. p :: = 0 | a :: = 0 |
| 2. p :: = 1 | a :: = 1 |
| 3. q :: = p | b :: = a |
| 4. q :: = qp | b :: = ab |
| 5. r :: = + | c :: = + |
| 6. r :: = × | c :: = × |
| 7. s :: = q | d :: = b |
| 8. s :: = [srs] | d :: = d; dc |

It can be shown (cf. [2]) that $G$ is well — translatable into $G^*$. The *correspondence table* (table 1) needs no explanations. Now table 2 describes the whole process of translation of the string [10 × [110 + 1]]. In row 0 we see the input string $e_0$ marked in accordance with the first step of the translation algorithm. $e_0$ is reduced to $e_1$ by the group (I) rule $\mathfrak{r}_2 = (\text{p} :: = 1)$. In accordance with step 3, $\tau(1) = 1$ is put into the output string and the symbol p in $e_1$ is marked accordingly. Next, $e_1$ is reduced to $e_2$ by the group (II) rule q :: = p. No change results in the output string (see step 4), and the symbol q in $e_2$ will have the same superscript as p in $e_1$. In fact, the only changes produced by group (II) rules take place when $m_i > 1$. Then the new symbol replaces more than one symbol; the strings assigned to them are immediately adjacent and, by definition, their order need not be changed. Thus the string assigned to the new symbol is

**Table 1.**

| Number | Group | $\mathfrak{r}$ | Bracketed $\Phi(\mathfrak{r})$ |
|---|---|---|---|
| 1 | I | p : : = 0 | a : : = $\{C_1\}$ |
| 2 | I | p : : = 1 | a : : = $\{C_1\}$ |
| 3 | II | q : : = p | b : : = $\{C_1\}$ |
| 4 | III | q : : = qp | b : : = $\{C_1\}\ \{C_2\}$ |
| 5 | I | r : : = + | c : : = $\{C_1\}$ |
| 6 | I | r : : = × | c : : = $\{C_1\}$ |
| 7 | II | s : : = q | d : : = $\{C_1\}$ |
| 8 | III | s : : = [srs] | d : : = $\{C_2\}; \{C_4\}\ \{C_3\}$ |

**Table 2.**

| | Marked input string | | | | | | | | | | | | $q_i$ | $p_i$ | Group | Output string | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | | | | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | $y_8$ | $y_9$ | $y_{10}$ |
| 0 | $[^{(0,0)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $\times^{(0,)0}$ | $[^{(0.0)}$ | $1^{(0,0)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | | | | | | | | | | | |
| 1 | $[^{(0,)0}$ | $p^{(1,1)}$ | $0^{(0,0)}$ | $\times^{(0,0)}$ | $[^{)0,0)}$ | $1^{(1,1)}$ | $1^{(0,0)}$ | $(0^{0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | 2 | 2 | I | 1 | | | | | | | | | |
| 2 | $[^{(0,0)}$ | $q^{(1,1)}$ | $0^{(0,0)}$ | $\times^{(0,0)}$ | $[^{)0,0)}$ | $1^{(0,0)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,)0}$ | 2 | 3 | II | 1 | | | | | | | | | |
| 3 | $[^{(0,0)}$ | $q^{(1,1)}$ | $p^{(2,2)}$ | $\times^{(0,0)}$ | $[^{(0,0)}$ | $1^{(0,0)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{\ 0,0)}$ | $]^{(0,0)}$ | 3 | 1 | I | 1 | 0 | | | | | | | | |
| 4 | $[^{(0,0)}$ | $q^{(1,2)}$ | $\times^{(0,0)}$ | $[^{(0,0)}$ | $1^{(0,0)}$ | $1^{(0,0)}$ | $0^{0,0)(}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | 2 | 4 | III | 0 | 1 | | | | | | | | |
| 5 | $[^{(0,0)}$ | $s^{(1,2)}$ | $\times^{(0,0)}$ | $[^{(0,0)}$ | $1^{(0,0)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | 2 | 7 | II | 0 | 1 | | | | | | | | |
| 6 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $1^{(0,0)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | 3 | 6 | I | 0 | 1 | $\times$ | | | | | | | |
| 7 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $p^{(4,4)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | 5 | 2 | I | 0 | 1 | $\times$ | 1 | | | | | | |
| 8 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $q^{(4,4)}$ | $1^{(0,0)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | 5 | 3 | II | 0 | 1 | $\times$ | 1 | | | | | | |
| 9 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $q^{(4,4)}$ | $p^{(5,5)}$ | $0^{(0,0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | 6 | 2 | I | 0 | 1 | $\times$ | 1 | 1 | | | | | |
| 10 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $q^{(4,5)}$ | $0^{(0.0)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | 5 | 4 | III | 0 | 1 | $\times$ | 1 | 1 | | | | | |
| 11 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $q^{(4,5)}$ | $p^{(6,6)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | 6 | 1 | I | 0 | 1 | $\times$ | 1 | 1 | 0 | | | | |
| 12 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $q^{(4,6)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | | 5 | 4 | III | 0 | 1 | $\times$ | 0 | 1 | 1 | | | | |
| 13 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $s^{(4,6)}$ | $+^{(0,0)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | | 5 | 7 | III | 0 | 1 | $\times$ | 0 | 1 | 1 | | | | |
| 14 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $s^{(4,6)}$ | $r^{(7,7)}$ | $1^{(0,0)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | | 6 | 5 | I | 0 | 1 | $\times$ | 0 | 1 | 1 | + | | | |
| 15 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $s^{(4,6)}$ | $r^{(7,7)}$ | $p^{(8,8)}$ | $]^{)0,0(}$ | $]^{(0,0)}$ | | | | 7 | 2 | I | 0 | 1 | $\times$ | 0 | 1 | 1 | + | 1 | | |
| 16 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $s^{(4,6)}$ | $r^{(7,7)}$ | $q^{(8,8)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | | 7 | 3 | II | 0 | 1 | $\times$ | 0 | 1 | 1 | + | 1 | | |
| 17 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $[^{(0,0)}$ | $s^{(4,6)}$ | $r^{(7,7)}$ | $s^{(8,8)}$ | $]^{(0,0)}$ | $]^{(0,0)}$ | | | | 7 | 7 | II | 0 | 1 | $\times$ | 0 | 1 | 1 | + | 1 | | |
| 18 | $[^{(0,0)}$ | $s^{(1,2)}$ | $r^{(3,3)}$ | $s^{(4,9)}$ | $]^{(0,0)}$ | | | | | | | | 4 | 8 | III | 0 | 1 | $\times$ | 1 | ; | 0 | 1 | 1 | + | |
| 19 | $s^{(1,10)}$ | | | | | | | | | | | | 2 | 8 | III | 1 | ; | 0 | 1 | 1 | + | ; | 0 | 1 | $\times$ |

formed simply by concatenating these strings and the superscript is formed accordingly. The reductions 2, 5, 8, 13, 16, 17 are all of this type.

Let us now consider the reduction 4 using the rule $\mathfrak{r}_4 = (\mathrm{q} ::= \mathrm{qp})$, where the bracketed $\Phi(\mathfrak{r}_4) = (\mathrm{b} ::= \{c_1\}\{c_2\})$. Step 5 of the translation algorithm puts $A(\mathrm{q})$ and $A(\mathrm{p})$ into temporary storage and then puts them into the output string again, but in the same order as the corresponding brackets in the bracketed rule $\Phi(\mathfrak{r}_4)$, i.e., $A(\mathrm{p})\,A(\mathrm{q})$. In other reductions of this type (e.g. 18), symbols from $V_a^*$ are introduced in appropriate places. Finally, $e_{19} \in V_n$. This terminates the translation process and $y_1 \dots y_{10}$ is the required translation.

*Note*: It is clear that the same procedure could be used for group (II) rules; but the process would thereby be unnecessarily complicated.

## JUSTIFICATION

It remains to be shown that

(a) the required operations can always be performed
(b) the result is a translation in the sense of [1], [2].

To justify assertion (a), it is sufficient to show that for a group (III) rule $\mathfrak{r} = (a_0 ::= b_0 a_1 b_1 \dots a_k b_k)$ (with $\Phi(\mathfrak{r}) = (c_0 ::= d_0 c_1 \dots c_k d_k)$ (no parts of the output string except $A(a_1), \dots, A(a_k)$ are erased in forming $A(a_0)$. This follows readily from the canonical property of the reduction sequence produced by the analysis algorithm; for any such strings would have to follow $A(a_1), \dots, A(a_k)$ and thus be the product of subsequent reductions.

*Note*: There are thus only two reasons for demanding that the reduction sequence be canonical: to ensure that the aforementioned operation can be performed and to justify the simplified procedure for group (II). If we modify the algorithm so as to use the full procedure for group (II) and to include a suitable relocation of symbols threatened with erasure, *any* reduction sequence can be used.

To justify (b), let us consider various types of rules used in the reduction sequence. For a group (I) rule $x ::= y$ clearly $C(y) = \tau(y)$ (step 2), so that $S^*(C(y)) = S(y)$. Suppose now that a group (II) or (III) rule has been used in a reduction. First let $a_i \in V_t$ for $i = 1, 2, \dots, k$. Then the conditions of (3) are satisfied and $S^*(C(b_0 a_1 \dots \dots a_k b_k) = S(b_0 a_1 \dots a_k b_k)$. Now let $a_1, \dots, a_k$ be nonterminal symbols once removed from their terminal equivalents. For all of then the requirements of (3) are satisfied (the method of proof being similar to that used for group (I) rules) and therefore

$$S^*(C(b_0 a_1 \dots a_k b_k)) = S^*(A(a_0)) = S(b_0 x_1 \dots x_k b_k)\,,$$

where $x_1, \dots, x_k$ are the said equivalents.

The rest follows by induction on the distance of symbols from their terminal equivalents.

(Received September 22nd, 1965.)

[1] K. Čulík: Semantics and Translation of Grammars and ALGOL-like Languages. Kybernetika *1* (1965), 1, 47–49.
[2] K. Čulík: Well-translatable Grammars and ALGOL-like Languages. Mimeographed.
[3] J. Eickel: Generation of Parsing Algorithms for Chomsky Type-2 Languages. Bericht nr. 6401, Tech. Hochschule München.
[4] J. Eickel et al.: A Syntax Controlled Generator of Formal Language Processors. CACM *6* (1963), 8, 451–455.

VÝTAH

# Syntakticky řízený překladač pro jazyky typu ALGOL

JAN VINAŘ

V práci je vylíčena modifikace Čulíkova překládacího algoritmu z [1], [2], která pracuje místo s grafy s posloupnostmi symbolů, nepoužívá vůbec neterminálních symbolů cílového jazyka a má menší nároky na paměť, čímž se zdá vhodná pro případné strojové použití. Je uveden příklad překladu a je ukázáno, že algoritmus je vždy proveditelný a dává správné výsledky.

*Jan Vinař, prom. mat., katedra matematiky University P. J. Šafárika, nám. Februárového víťazstva 9, Košice.*