ON ATTRIBUTE GRAMMARS

V.N.Agafonov
Institute of mathematics,
Novosibirsk 90, USSR

A common mechanism for specifying a programming language L can be viewed as consisting of two components. The first one is a context free (CF) grammar G which generates a broader language $L(G) \supset L$. The second one contains more or less formal rules which take into account so called context conditions and select from language L(G) strings of language L. In the recent literature on formal languages much attention is paid to the following way of specifying languages which are not context free. A CF grammar G is supplied with some type of device which controls applications of production rules in the course of derivations. This control device C picks out, from the set of all derivations, a subset called control set. A control device grammar $G'=(G,C)$ defines language $L(G') \subset L(G)$ in such a way that a string $\varphi \in L(G)$ belongs to L(G') iff its derivation is contained in the control set. Such grammars include, among others, matrix, programmed and conditional grammars (see the survey [9]). Restrictions which are imposed on derivations in these grammars are but poorly related to semantics of programming languages and corresponding context conditions. In this paper a method for specifying languages which are not CF is suggested and investigated in which restrictions are computed in such a way that was previously used for defining semantics of programming languages [5, 6]. This method is based on the notion of attribute CF grammar introduced by Knuth [6] for specifying semantics of CF languages. In [7] Knuth's notion was transformed into the concept of attributed translation (AT) grammar adapted for specifying translations of CF languages whose terminal strings are supplied with attributes.

I. An attribute generative (AG) grammar G is a pair (G',C) where G' is a CF grammar (called the base of the grammar G) and C is a control device. The grammar $G'= (N,T,R,S)$ is supposed to be reduced, the start nonterminal $S \in N$ does not appear on the right-hand side of any production rule $r \in R$, and for any nonterminal $A \in N$ there does not exist a nontrivial derivation $A \overset{+}{\Rightarrow} A$. The control device $C = (A,V,F,\sigma)$ is defined as follows. I) A is a function which associates a finite set $A(X)$ of attributes to each nonterminal $X \in N$. Each $A(X)$ is partitioned into two disjoint sets, the synthesized attributes $A_{syn}(X)$ and the inherited attributes $A_{inh}(X)$. It is supposed that $A_{inh}(S) = \emptyset$. A set $A' = \bigcup_{X \in N} A(X)$ will be called attribute alphabet. 2) V is a function which associates a set of values to each attribute $\alpha \in A$. Let the set $\bigcup_{\alpha \in A'} V(\alpha)$ be set of strings in a finite alphabet. 3) F is a finite set of control functions which is defined in the following way. Let the r-th production of the set R ($I \leq r \leq m$) be $X_0 \rightarrow \varphi_0 X_I \varphi_I X_2 \ldots X_{m_r} \varphi_{m_r}$ where $\varphi_i \in T$, $X_i \in N$. Then a control function set $F_r = \{ f_{rj\alpha} \mid I \leq j \leq m_r$ and $\alpha \in A_{syn}(X_j)$ if $j = 0$, $\alpha \in A_{inh}(X_j)$ if $j > 0 \}$ corresponds to this production and $F = \bigcup_{I \leq r \leq m} F_r$. The function $f_{rj\alpha}$ is a recursive mapping of $V(\alpha_I) \times \ldots \times V(\alpha_t)$ into $V(\alpha)$, for some $t = t(r,j,\alpha) \geq 0$, where each $\alpha_i = \alpha_i(r,j,\alpha)$ is an attribute of some X_k , for $0 \leq k_i = k_i(r,j,\alpha) \leq m_r$, $I \leq i \leq t$. 4) $\sigma \in A_{syn}(S)$ is a special attribute for which $V(\sigma) = \{0,I\}$ and $\sigma \notin \bigcup_{X \neq S} A(X)$. A language $L(G) \subseteq L(G')$ generated by the AG grammar G is defined as follows. Let $\varphi \in L(G')$ and t be a derivation tree of φ in the CF grammar G'. If each node of the tree labeled by a nonterminal X is labeled also by attributes of the set $A(X)$ then we obtain the corresponding attributed tree. Now, by means of control functions $f_{rj\alpha}$ the computation of the value of each attribute in the tree should be attempted. If this can be done and the value of the attribute σ is I then the tree t is accepted. The string φ belongs to $L(G)$ iff φ has an accepted tree.

In the sequel AG grammars are supposed to be well defined (G is well defined [6] if for any attributed tree values of all atributes at all nodes can be defined).

The difference between AG grammars and Knuth grammars is that in an AG grammar I) terminals have no attributes (or if you like the terminal itself is its only own attribute) and 2) there is a special attribute whose value at the root of a given tree tells us whether this tree is accepted or not. These differences reflect different viewpoints. An AG grammar is intended for generating a language which is not CF and so satisfies some context conditions whereas a Knuth grammar is intended for assigning values (meanings) to strings of a CF language. It should also to be noted that in the definition of AG grammar values of attributes are only strings in a finite alphabet.