ON THE DEFINITION OF AN ATTRIBUTE GRAMMAR

Martti Tienari
Department of Computer Science
University of Helsinki
Tukholmankatu 2
SF-00250 HELSINKI 25

ABSTRACT

The definition of an attribute grammar is analyzed in order to simplify and clarify certain points. An easy-to-test property termed "balance" is shown to be useful in analysis. The classification of attributes into synthesized and inherited attributes is shown to be a property derivable as a logical consequence of the "balance".

1. Introduction

Attribute grammars were defined in [3] in 1968. Since then they have been employed for many purposes, often to give a theoretical framework for semantic analysis in compiler writing. The basic definition of attribute grammar varies with different authors [1], [2], [3], [4] and, in terms of the fine details of the definition, no generally accepted standard definition seems to have emerged yet.

The aim of this article is to scrutinize the definition of an attribute grammar. We analyze the concept "well-defined" attribute grammar and demonstrate that a related simple concept "balanced" attribute grammar is also useful in the logical development of the theory. As an interesting new result we show that the property of a semantic attribute to be either "synthesized" or "inherited" is owned by any attribute in a balanced attribute grammar. Thus the concepts of synthesized and inherited attributes are not necessary in the basic definition of an attribute grammar. These can be considered as properties derivable as a logical consequence of the basic definition.

2. Basic definition of an attribute grammar

An attribute grammar is constructed as an extension of a context-free grammar. Let the vocabulary of the context-free grammar be $V = V_N \cup V_T$, $V_N \cap V_T = \phi$, partitioned into nonterminals V_N and terminals V_T. The productions of this grammar are expressed in the form "$A \rightarrow \alpha$" where $A \in V_N$ is a nonterminal and $\alpha \in V^*$ is any string over V.

We say the attribute grammar is "reduced" if the context-free grammar has no useless productions. Any string in a context-free language has a "derivation tree" or "parse tree" describing the way the string can be derived by the grammar rules.

We associate with the symbols in the vocabulary of a context-free grammar attribute symbols or attribute names from a finite set of attribute symbols A. Each symbol $X \in V$ in the context-free grammar is associated with a set of attribute symbols $A(X) \subset A$. Further, each attribute symbol $a \in A$ is associated with a range of values or type. The nodes of a derivation tree of a sentence in the context-free language are labeled with the symbols $X \in V$ in the vocabulary of the context-free grammar. We attach to a node labeled with X a set $\{a(X) \mid a \in A(X)\}$ of attribute instances [4] (called "attribute occurrence" in [1]). An instance a(X) has the same range of values as the corresponding attribute symbol $a \in A$. Each attribute symbol usually has an instance associated with several grammar symbols and several nodes in a parse tree.

Each attribute instance in any parse tree of the grammar has an associated evaluation rule which eventually gives a value to the instance. The evaluation rule is either external or internal. An external evaluation rule gives a value to an attribute instance at the root node or a leaf node of the derivation tree. These evaluation rules are not specified as part of the attribute grammar formalism; they are defined in the environment in which the attribute grammar is embedded. An internal evaluation rule gives the value of an attribute instance as a function of other attribute instances located at neighboring nodes of the parse tree. To be more specific: When a branch of the parse tree is associated with the production $X_0 \rightarrow X_1X_2\ldots X_n$, $n \geq 0$, an internal evaluation rule of the attribute instance $a(X_k)$ is of the form $a(X_k) := f(y_1,\ldots,y_r)$, $r \geq 0$, where f is an evaluation function of $a(X_k)$ and the arguments $y_1,\ldots,y_r$ are attribute instances of the set $\bigcup_{i=0}^{n}\{a(X_i) \mid a \in A(X_i)\}$. For each instance of the production $X_0 \rightarrow X_1\ldots X_n$ in the parse tree the same set of evaluation rules must be used. Thus, a finite collection of evaluation rules is associated with each production of the context-free grammar.

The evaluation rules are used to give values to the attribute instances at the nodes of the derivation tree. Initially the values of all attribute instances are undefined. The external evaluation rules are first applied. Then internal evaluation rules are applied in any order subject to the constraint that the values of the arguments in the evaluation function are defined before the corresponding rule is applied. The derivation tree with evaluated attribute instances at its nodes is the end result of the analysis of a sentence. This derivation tree with attribute values is useful in the semantic analysis and interpretation of the sentence at hand.

3. Properties of attribute grammars

A useful attribute grammar should have the following properties: (1) all attribute instances at all nodes of all derivation trees should be given a value in the evaluation process, (2) the value of an attribute instance should be independent of the order in which the evaluation rules are executed. When the condition (1) is fulfilled, the attribute grammar is said to be well-defined [3] or well-formed [4]. Condition (2) expresses the fact that the set of attribute evaluation rules is declarative i.e. the outcome of applying these rules is independent of the order in which this is carried out. A slightly restricting but natural way to guarantee that this condition is fulfilled is to demand that any attribute instance at any node of any derivation tree should have exactly one evaluation rule, no more, no less. If this condition is fulfilled we say the attribute grammar is balanced.

If there were an attribute instance without an evaluation rule, the grammar could not be well-defined because this particular attribute instance could not get a value. On the other hand, if there were an attribute instance with two or more evaluation rules, the outcome of the evaluation process would in general be dependent on the evaluation order unless all the rules of this attribute instance always yielded the same value. One could ask, in any case, why two or more evaluation rules for an attribute instance were needed.

The property of an attribute grammar of being well-defined can be shown to be equivalent to the absence of circular definitions. The attribute grammar is defined to be "circular" if the evaluation dependency graph of attribute instances in a derivation tree contains an oriented cycle; otherwise the grammar is "noncircular". To be more precise, the following theorem [3] is true.

Theorem 1: A balanced attribute grammar is well-defined if and only if the grammar is noncircular.

Proof: Although this result is proved in [3] it is worthwhile going through the argument in detail in order to see the role of balence in the proof.

If the grammar is noncircular it is easy to see that the evaluation dependency relation among attribute instances in any derivation tree is a partial order. Thus the attribute instances in any derivation tree can be ordered linearly and evaluated in linear order. Note that this argument is valid for a balanced grammar because in such a grammar there is an evaluation rule available for each attribute instance.

If the grammar were circular we could find a derivation tree with an evaluation cycle consisting of the attribute instances $a_1(X_1),\ldots,a_h(X_h)$, $h > 1$, $a_h(X_h) = a_1(X_1)$, where evaluation of the attribute instance $a_i(X_i)$ is possible only if the value of $a_{i+1}(X_{i+1})$ is available. Due to balance, no more than one evaluation rule is

available for each attribute instance. Thus there is no way the interdependent attribute instances in the cycle could be evaluated. The value of the attribute instance $a_1(X_1)$ is ultimately dependent on the value of $a_1(X_1)$ itself. Thus a circular, balanced attribute grammar cannot be well-defined. □

4. Synthesized and inherited attributes

A balanced attribute grammar has a property which makes it essentially easier to deal with. The attribute symbols A(X) associated with any grammar symbol X can be partitioned into two disjoint sets: the set of "synthesized" attributes $A_S(X)$ and the set of "inherited" attributes $A_I(X)$, $A(X) = A_S(X) \cup A_I(X)$, $A_S(X) \cap A_I(X) = \phi$.

An internal evaluation rule associated with the production $X_0 \rightarrow X_1 \ldots X_n$ is called either synthesizing, if it evaluates an attribute instance $a(X_0)$ at a node labeled with the left-hand side nonterminal X_0, or inheriting, if it evaluates an attribute instance $a(X_k)$, $k \geq 1$, at a node labeled with a symbol X_k on the right-hand side of the production. Correspondingly, an external evaluation rule is classified as "synthesizing" or "inheriting"according to whether it evaluates an attribute instance at a leaf node or the root node of a parse tree.

An attribute $a \in A(X)$ is called synthesized resp. inherited if an attribute instance a(X) is evaluated with a synthesizing resp. inheriting evaluation rule.

The definitions of "synthesized" and "inherited" attributes, based on classifying the evaluation rules into synthesizing and inheriting ones, classify the attributes of A(X) into two possibly overlapping sets $A_S(X) \subset A(X)$ and $A_I(X) \subset A(X)$. For any balanced attribute grammar the sets of synthesized and inherited attributes $A_S(X)$ and $A_I(X)$ do not overlap.

Theorem 2 : For any balanced and reduced attribute grammar the sets of synthesized and inherited attributes $A_S(X)$ and $A_I(X)$, $X \in V$, do not overlap but give a partition $A(X) = A_S(X) \cup A_I(X)$, $A_S(X) \cap A_I(X) = \phi$.

Proof: We note first that in a balanced attribute grammar any attribute instance has an evaluation rule; thus each attribute $a \in A(X)$ can be given the property of being either synthesized or inherited or, possibly both synthesized and inherited. Thus the sets $A_S(X)$ and $A_I(X)$ subdivide, possibly in an overlapping way, the set of attributes A(X), $A(X) = A_S(X) \cup A_I(X)$.

Let us assume that an attribute $a \in A(X)$ is both synthesized and inherited. Then there will be a production $Y_0 \rightarrow Y_1 \ldots Y_n$ associated with a synthesizing evaluation rule to evaluate an instance a(X), $X = Y_0$, and also a production $Z_0 \rightarrow Z_1 \ldots Z_m$ associated with an inheriting evaluation rule to evaluate an instance a(X), $X = Z_k$, $k \geq 1$. In a derivation tree where below production $Z_0 \rightarrow Z_1 \ldots Z_m$ the production

$Y_0 \to Y_1 \ldots Y_n$, $Y_0 = X$, is applied at the immediate descendant node $Z_k = X$, $k \geq 1$, of Z_0, we would get two separate evaluation rules for the attribute instance $a(X) = a(Y_0) = a(Z_k)$. In a balanced attribute grammar this is not possible. Thus the sets $A_S(X)$ and $A_I(X)$ do not overlap. □

We have shown that the attributes A(X) associated with any grammar symbol can be partitioned into disjoint sets $A_S(X)$ and $A_I(X)$. In practical applications of attribute grammars it would be helpful if the set A of all attribute symbols could be partitioned in the similar way: $A = A_I \cup A_S$, $A_I \cap A_S = \phi$. This is actually possible if we eventually rename some attributes as follows: if the attribute $a \in A_S(X)$ when $X \in V' \subset V$ and $a \in A_I(X)$ when $X \in V'' \subset V$, $V' \cap V'' = \phi$, we introduce a new unused attribute name "b" and rename "a" with "b" in the latter cases. The attribute grammar remains essentially the same in this renaming process; the number of attribute symbols is only increased by one. After the renaming process $a \in A_S$ and $b \in A_I$.

We can thus assume or require without sacrificing the descriptive power of the attribute grammar formalism that in the attribute grammar we always have $A_S = \bigcup_{X \in V} A_S(X)$, $A_I = \bigcup_{X \in V} A_I(X)$, $A_S \cap A_I = \phi$. The properties expressed by the adjectives "synthesized" and "inherited" can be considered as properties associated with the attribute symbols, not only properties of an attribute associated with the grammar symbol. This feature makes it easier to grasp intuitively the workings of an attribute grammar. It also simplifies algorithmic handling of attribute grammars in a computer.

5. Discussion

We generalized slightly the definition of an attribute grammar in a way which, to our taste, is a simplification and permits a pedagogically more pleasing introduction to the concept. We are able to postpone the classification of attributes as synthesized and inherited ones to a stage where the attribute instances attached to the parse tree and the attribute evaluation rules have already been defined.

Our novel approach to the definition of an attribute grammar is intended to be a variation in the style of the formal development of the theory. We do not, herewith, advocate necessarily any change in the practical usage of an attribute grammar e.g. in the input requirements of a compiler generating system.

In our generalized definition we must require any useful attribute grammar to be balanced. In our slightly modified terminology a "balanced attribute grammar" corresponds to what the authors of the references [1], [2] and [3] have defined to be an "attribute grammar". In [4, p. 18] an equivalent condition of "validity" is explicitly formulated for the collection of attribute evaluation rules associated with a grammar production.

The balance of an attribute grammar can be tested quite straightforwardly as follows:

(1) First , check the evaluation rules; if the attributes A(X) associated with the symbol X, $X \in V$, cannot be partitioned into the set $A_S(X)$ of synthesized attributes and the set $A_I(X)$ of inherited attributes, $A(X) = A_S(X) \cup A_I(X)$, $A_S(X) \cap A_I(X) = \phi$, the grammar is not balanced. When the copy rules of attributes are to be generated automatically we should require at least one semantic rule for each attribute name. This would enable us to classify the attributes as synthesized or inherited. The default copy rules should be generated before the next step (2) is undertaken.

(2) Second, for each production test that the collection of evaluation rules associated with the production has exactly one rule for each synthesized attribute of the left-hand side nonterminal as well as for each inherited attribute of the grammar symbols present in the right-hand side.

(3) Third, make sure that there exists an external evaluation rule for each inherited attribute of the root symbol as well as for each synthesized attribute of any terminal symbol.

One might think that any well-defined grammar would be balanced. However, there exist simple counterexamples which show that balancedness cannot be deduced from the well-definedness as the latter property is defined in this paper. Take the context-free grammar: (1) $W \rightarrow Z$, (2) $Z \rightarrow d$. Associate attribute symbols with grammar symbols as follows: $A(W) = \phi$, $A(Z) = \{a,b\}$. Associate with production $W \rightarrow Z$ an evaluation rule $a(Z) := b(Z)$, further associate with production $Z \rightarrow d$ the evaluation rules $a(Z) := b(Z)$ and $b(Z) := 1$. This grammar has only one derivation tree corresponding to the derivation chain $W \Rightarrow Z \Rightarrow d$. The values of the attribute instances at the node labeled with Z are given values $a = b = 1$ in the evaluation process. However, the grammar is not balanced, because the attribute instance a(Z) at the derivation tree is over-defined. It has two evaluation rules, one inheriting rule associated with the production $W \rightarrow Z$, and another synthesizing rule associated with the production $Z \rightarrow d$. This is also an example of an attribute grammar where the attributes cannot be partitioned into synthesized and inherited attributes: The attribute "a" is both synthesized and inherited.

In the literature treating attribute grammars, it is customary to consider the concepts of "well-definedness" and "noncircularity" as synonyms. When our generalized definition of attribute grammar is used the equivalence of these concepts should, however, somehow be qualified as we did in theorem 1. If we permit a grammar to be unbalanced, that grammar can be well-defined and circular at the same time. Consider again the context-free grammar $W \rightarrow Z$, $Z \rightarrow d$ with attribute associations $A(W) = \phi$, $A(Z) = \{a,b\}$. Define evaluation rules as follows: With production $W \rightarrow Z$ evaluate $a(Z) := b(Z)$, with production $Z \rightarrow d$ evaluate $b(Z) := a(Z)$, $b(Z) := 1$. This unbalanced attribute grammar has

an evaluation cycle $b(Z) \rightarrow a(Z) \rightarrow b(Z)$ at node Z of the derivation tree. However, it is well-defined in our generalized terminology, because attributes a(Z) and b(Z) get their values = 1 with help of the rule $b(Z) := 1$. This extra rule breaks the stalemate caused by the cycle in the evaluation dependency graph.

Most authors disallow inherited attributes at the root or synthesized attributes at the leaves of the derivation trees. The literature references analyzed in [4, p. 15] show no consistency in the way attribute associations are restricted. The judgement on which kind of attributes are useful varies with the author. In our experience with a compiler writing systen [5] employing an attribute grammar, external evaluation rules and consequently putting no restrictions on attribute association at the root or leaves of the parse tree appear to constitute a useful descriptive device in defining the interface with the environment of the grammar. It also makes modular working with the subgrammars of a big attribute grammar conceptually cleaner. Further, the policy of allowing any kind of attribute associations and introducing the external evaluation rules neatly fits in the generalized analysis of an attribute grammar.

Acknowledgements: I would like to thank Kai Koskimies, Kari-Jouko Räihä, Herbert Schwetman and Eljas Soisalon-Soininen for their critical comments of the draft paper. The counterexample showing that balancedness cannot be deduced from well-definedness is due to an undergraduate student, Pekka Orponen, from my language semantics class.

References

1. Bochmann, G. V. : Semantic evaluation from left to right,Comm. ACM 19 (1976), 55-62.

2. Jazayeri, M. : On attribute grammars and the semantic specification of programming languages. Report 1159, Jennings Computing Center, Case Western Reserve University, Cleveland, Ohio, October 1974.

3. Knuth D. E. : Semantics of context-free languages, Math. Syst. Th. 2 (1968), 127-145

4. Räihä, K-J. : On attribute grammars and their use in a compiler writing system, Department of Computer Science, University of Helsinki, Report A-1977-4.

5. Räihä, K-J., Saarinen, M., Soisalon-Soininen, E., Tienari, M. : The compiler writing system HLP (Helsinki Language Processor), Department of Computer Science, University of Helsinki, Report A-1978-2.