Incremental Attribute Evaluation
for Multiple Subtree Replacements in Structure-Oriented Environments

An Feng, Tohru Kikuno and Koji Torii

Department of Information and Computer Sciences
Faculty of Engineering Science, Osaka University
Toyonaka, Osaka, Japan, 560

Abstract
In this paper, we discuss the development of a structure-oriented environment (that is, an interactive tool) by applying attribute grammars. First, we define a new model \( MR \) in which replacement of multiple subtrees of a given attributed tree \( T \) is allowed. Then, we introduce a new scheme \((\text{Simp}(T), \text{Copy}(T), \text{Comp}(T))\), where \(\text{Simp}(T)\), \(\text{Copy}(T)\) and \(\text{Comp}(T)\) are the simplified attributed tree, copy tree and compressed tree, respectively. Based on the scheme, we propose an efficient algorithm that updates the values of all inconsistent attribute instances in the tree \( T \).

1. Introduction
The structure-oriented environments start with a syntax-directed editor, then provide single-user programming environments that support interactive semantic analysis, program execution and debugging, and are currently continued to support programming-in-the-many[1][5].

In this paper, we discuss the development of a structure-oriented environment (that is, an interactive tool) by applying attribute grammars, in which a program is represented as an attributed tree and the modification of the program is represented as the replacements of subtrees in the attributed tree.

Concerning the evaluation of the values of the attribute instances in a given attribute tree, there exist some well-known interesting problems: Bypassing of copy chains and computation of (topological) ordering among inconsistent attribute instances. Many researchers have already tried to solve these two problems[2][3][4][5][9]. Especially, Reps et al.[9] has proposed a new data structure "compressed trees" and solved the ordering problem for a subclass of noncircular attribute grammars (NC-AG's)[6][9].

In this paper, we will introduce three kinds of trees: copy trees, a compressed tree and a simplified attributed tree. Based on this, we can reduce the memory size required to store information in the attributed tree as well as the amount of the time required to solve the problems mentioned above. Especially, the problem of computing the ordering
among \( y \) inconsistent attribute instances can be solved in \( O(y \log u) \) time for the class of NC-AG's, where \( u \) is the number of attribute instances in the attributed tree.

2. Motivation and Previous Works

2.1 Attribute grammars

An attribute grammar\[6\] \( G \) is a context-free grammar \( G_0 \) that has been extended by associating a set \( A(X) \) of attributes with each nonterminal \( X \) of \( G_0 \). \( A(X) \) is divided into two mutually disjoint subsets: a set of inherited attributes and a set of synthesized attributes. Associated with each production \( p \) of \( G_0 \) is a set of semantic rules. Each semantic rule defines a value of an attribute \( attr \) of one of the nonterminals in \( p \) in terms of a semantic function applied to other attributes in \( p \).

Example 1 Figure 1 shows an example of attribute grammar, which generates a simple language with declaration and use statements.

A vertex labeled \( X \) in a derivation tree defines a set of attribute instances corresponding to the attributes of \( X \). An attributed tree \( T \) is a derivation tree together with an assignment of a value to each attribute instance of the tree. Let \( \text{val}(attr) \) denote the value assigned to attributed instance \( attr \) in the tree \( T \). An attribute instance \( attr \) is inconsistent if \( \text{val}(attr) \neq f(\text{val}(attr_1), ..., \text{val}(attr_k)) \), where \( f(attr_1, ..., attr_k) \) is the semantic function of \( attr \). In other cases, we say \( attr \) is consistent.

The definitions and notations for sets of attribute instances in tree \( T \) are summarized in Table 1. An attributed tree \( T \) is consistent if \( \mathcal{A}(T) \) is empty.

The functional dependencies among attribute instances in \( \mathcal{A}(T) \) are represented by a compound dependency graph, denoted \( \mathcal{R}(T) = (V, E) \), where \( V = \mathcal{A}(T) \), \( E = \{(a, b) \mid a \) is an argument of the semantic function of \( b \)\}. 

Example 2 An attributed tree \( T \) for "\( \text{dec } x \text{ use } x \text{ dec } y \text{ use } y \)" is shown in Figure 2. Attribute instances are shown in boxes, and their values are denoted as italic letters. Attribute instances \( v_1.err \) and \( v_3.err \) are inconsistent. The dependency graph \( \mathcal{R}(T) \)