

Preface

This thesis concerns the design of interactive, language-based programming environments that use knowledge of a programming language to provide functions based on the structure and meaning of programs. The goal of the research is a system-constructor to enable editors for different languages to be created easily.

The most challenging aspect of such a system is the design of the semantic component, because a language-based editor performs static semantic analysis when a program is altered in order to detect erroneous constructions or to prevent illegal modifications. For efficiency, this should be performed incrementally, re-using as much old information as possible; therefore, a major focus of my research concerned a model of editing for which it is possible to perform incremental semantic analysis efficiently.

In this model, a program is represented as an attributed tree in which all attributes have consistent values; programs are modified by tree operations such as pruning, grafting, and deriving. After each modification, some of the attributes require new values; incremental semantic analysis is performed by updating attribute values to again make them all consistent. The thesis presents several algorithms for this process that are asymptotically optimal in time.

The chief disadvantage of attribute grammars is that they use large amounts of storage. In the thesis, I discuss three aspects of utilizing storage efficiently in such systems. One way to reduce the amount of storage used is to reduce the number of attribute values retained at any stage of attribute evaluation. Two results are established concerning this idea: I present one algorithm for evaluating an n -attribute tree that never saves more than $O(\sqrt{n})$ attribute values, and a second algorithm that never saves more than $O(\log n)$ attribute values. A second method for reducing the amount of storage is to share the space used for storing attributes whose values are complex data structures; in the thesis I present a very general method for such sharing that can be applied to attributes of many types. Finally, I describe how, by restricting the class of attribute grammars, it is possible to reduce the amount of storage overhead required for updating trees in optimal time.