Incremental Code Generation in a Distributed Integrated Programming Environment

Michael James McCarthy

Thesis submitted for the degree of Doctor of Philosophy in The University of Adelaide (Department of Computer Science)

December 1995
# Table of Contents

1. Introduction ........................................................................................................... 1
   1.1. Contributions of this thesis .......................................................................... 3
   1.2. Outline of this thesis ...................................................................................... 5

2. Incremental compilation ....................................................................................... 7
   2.1. Basic definitions .............................................................................................. 8
   2.2. Goals and issues ............................................................................................ 10
   2.3. Incremental code generation .......................................................................... 14
   2.4. Previous systems ........................................................................................... 29
   2.5. Conclusions ..................................................................................................... 45

3. A distributed architecture for an integrated programming environment .......... 48
   3.1. Integrated programming environments ...................................................... 49
   3.2. A client-server architecture .......................................................................... 59
   3.3. Conclusions ..................................................................................................... 69

4. Abstract syntax trees .......................................................................................... 71
   4.1. Abstract syntax ............................................................................................... 72
   4.2. Tree domains and tree addresses ................................................................. 74
   4.3. Tree patterns .................................................................................................. 77
   4.4. Attribution of abstract syntax trees ............................................................... 78
   4.5. A simple model of editing .............................................................................. 82
   4.6. A language specification formalism .............................................................. 83
   4.7. Conclusions ..................................................................................................... 85

5. Incremental instruction selection ....................................................................... 86
   5.1. Retargetable code generation ........................................................................ 87
   5.2. BURS-based incremental instruction selection ........................................... 107
   5.3. Transforming abstract syntax trees .............................................................. 114
   5.4. The incremental instruction selection algorithm ....................................... 125
   5.5. Variations ........................................................................................................ 142
   5.6. Conclusions ..................................................................................................... 144
Abstract

An incremental compiler recompiles a program, after an edit, in time proportional to the size of the edit; this incremental recompilation will include incremental generation of object code for the program. This thesis presents a new method for performing incremental code generation in a distributed integrated programming environment. A prototype implementation of such an incremental code generator is also described.

Retargetable instruction selectors are generated from a formal specification of the architecture of the target computer. This thesis derives a new retargetable incremental instruction selection algorithm from a non-incremental instruction selection technique in the framework of a precise model of the underlying program representation. The resulting algorithm incrementally regenerates locally optimal object code after the replacement of a subtree in an abstract syntax tree program representation.

A greedy incremental code generator recompiles the updated program immediately after an edit. The impact on the response time of the environment is reduced if the incremental code generator executes in parallel with editing. An efficient greedy parallel incremental code generation algorithm is derived from the retargetable incremental instruction selection algorithm; the derivation of the parallel algorithm is also based on an analysis of the propagation of static semantic information after the replacement of an arbitrary subtree in the abstract syntax tree.

The integration of a greedy incremental code generator into the MultiView distributed integrated programming environment is demonstrated via a prototype implementation. An instance of this environment for a particular programming language is generated from a formal specification of the language and the target computer architecture on which the programs in the language are to be executed. Experimental data gathered from the prototype verifies that the recompilation time after a program update depends approximately linearly on the number of recompiled nodes.