Project Description

Problem to be Addressed

The development and verification of compiler software is a complex task. Various tools are available to automate, or semi-automate, some stages in the process of generating compiler software, such as: 'lex', a tool for the generation of lexical analyser software, and 'yacc', a tool for the generation of parser software. Genetic programming (GP) provides a means for the automatic, unattended development or optimisation of complex computer software, given data representative of the desired outcome of program execution. As compiler software attempts to provide a means for program transformation in a predictable and verifiable way, genetic programming may provide the means for the automatic induction of some or all of the functionality of a compiler.

This project will investigate if it is possible to use GP techniques to reliably induce code in an assembly-like language that, when executed, will have the same effect as an input program given in the form of a parse tree. That is, to investigate if GP be used to reliably perform the code generation phase of a compiler.

Proposed Solution

An assembly-like language and virtual machine will be devised, containing instructions and architecture similar to that found in a modern processor. A linear GP system will be implemented, working directly upon strings of instructions written in this assembly-like language. Individuals from the population of linear programs will be selected for recombination and mutation based upon their fitness values. Fitness values will be calculated by automatically observing individuals' behaviour when executed in the virtual machine. Bonus scores will be awarded to programs exhibiting desirable behaviour such as accessing the correct input registers and memory cells, and placing correct values in correct locations. Penalty scores will be awarded to programs exhibiting destructive or counter-productive behaviour, such as accessing or overwriting unrelated or invalid memory cells and entering infinite loops.

It is hoped that improved programs will be produced through recombination and mutation of programs that are observed to be well-performing, and that this process can be done with little or no intervention from a human operator. This project will investigate methods for identifying 'well-performing' programs within this context, and how the standard GP operations of crossover and mutation can be adapted and extended to expedite the evolution of appropriate programs. Additional measures to promote optimised programs, such as the use of parsimony pressure, will be investigated.

What Will be Produced

Over the course of the project I will produce a complete, functional and well-documented GP facility, with an accompanying series of predefined code generation suites capable of automatically reproducing solutions to the identified problem cases. These cases can then be reapplied to the GP facility to observe the evolution of solutions comparable to those identified in the report.

Modification to Existing Proposal

The specification of this project closely matches the existing proposal, "Using GP to evolve code in a compiler", put forward by David Jackson. The original proposal has been augmented by the definition of a subset of compiler functionality that will be examined in detail.

Conduct of the Project

Background Research

In preparation for the project, research will be undertaken into the theoretical underpinnings of compiler software in order to identify methods for code generation that may be applied to the GP system to assist in the evolution of appropriate programs, such as identifying redundant sections of code. The state of the art of GP will be examined, with a focus on research in producing GP systems directly manipulating machine code or processor microcode to solve problems.

Data Required
A series of test data in the form of input parse trees and their respective expected effects upon the virtual machine will be devised to train the GP population. This data will be sufficiently representative of the range of possible data that could be applied to the problem and incorporate a range of complexity in both length and instruction variety to allow us to be reasonable sure that the generated GP facility is sufficiently generalisable.

New Skills
I have very little experience in the specification, implementation and verification of compiler software; it will be necessary to become familiar with how compilers are specified and tested for correctness in order to ensure that the software produced during the project is suitable. In addition, I will need to gain experience in the implementation, creation and use of GP utilities.

Research Methodology
It will be necessary to experiment with the parameters of the linear GP system and the implementations of the operations upon the individuals in the population, based on methods within the body existing research and consideration of the identified problem.

Software Used
It is expected that the new GP system and its associated virtual machine will be authored in the C++ programming language. This language was chosen as it is compiled, rather than interpreted, and will therefore allow highly efficient execution necessary to meet the significant processing requirements anticipated for this project. Compilers for the language are readily available for many platforms, which may assist the expedient reapplication of the produced software to other hardware platforms in the future.

Statement of Deliverables

Anticipated Documentation
My report will include a detailed account of the design and creation of the software and a comprehensive guide to its operation. Each of the cases identified within the test data will be examined in detail, and the derivation of appropriate GP system parameters for the automatic generation of software performing the desired transformation will be discussed.

Anticipated Software
Over the course of the project I will produce a complete, functional and well-documented GP facility, with an accompanying series of predefined code generation suites capable of automatically reproducing solutions to the identified problem cases. These cases can then be reapplied to the GP facility to observe the evolution of solutions comparable to those identified in the report.

Anticipated Experiments
GP is a stochastic process and therefore experimentation will be required to determine the appropriate parameters for the GP run; these parameters include the population size, operator rates, the implementation of the operations upon the individuals in the population and the termination criterion. In addition, the linear GP model may be extended with additional experimental operations or parameters as yet unattempted.

Methods of Evaluation
The specified GP system and generation suites will be evaluated on their ability to automatically generate appropriate software to handle the identified problem cases, and the ability of the system to generalise to larger or related problems with little interaction from a human operator.

Project Timetable

Week 1, 2	Background Reading and Literature Review Underway
Week 3, 4	Project Specification Project Specification Submission Date: End of Week 4
Week 5, 6, 7, 8	Project Design Design Document and Presentation: During Week 9
Week 9, 10, 11, 12	Software Implementation and Testing
Week 12, 13, 14	Software Experimentation and Analysis of Results
Week 14, 15, 16	Write-up of Dissertation
	Friday September 24th Submission of Dissertation