Evolving Programs at Bytecode Level

Eric C. Collom

ABSTRACT

This paper discusses using x86 assembly language and Java Bytecode to evolve programs.

Categories and Subject Descriptors

Software Engineering []: I will change category later

General Terms

I will change general terms later

Keywords

ACM proceedings, LaTeX, text tagging
I will change the keywords later

1. INTRODUCTION

I plan to focus on the topic of taking programs that aren't designed to be evolved and evolving them through bytecode.

In plan to use the following sources: [8] and [7] as my main sources. I will possibly use [5] and [6] as more information on evolving programs for different EC purposes such as problem solving or debugging. Will use [2] and [1] as background into x86 assembly language and Java bytecode. I also have [3] and [4] as more research into using FINCH as an EC tool. However since these articles are so short and brief I don't think I will use them as sources in my final paper.

2. REFERENCES

[1] *The Java Language Specification.* Oracles America, Inc. And/or affiliates, 500 Oracle Parkway, Redwood City, California 94065, U.S.A., 2013.
This provides some needed background on the JVM, compiler, and Java bytecode. It is too long to read it all but will be a good reference for any questions about the JVM .

[2] x86 assembly language. January 2014.
This provides background for a general understanding of x86 assembly language .

[3] A. Benbassat and M. Sipper. Evolving artificial neural networks with finch. In *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, GECCO '13 Companion, pages 1719–1720, New York, NY, USA, 2013. ACM.
I don't think I will use this because it is too short and lacks substance.This paper talks about using FINCH to evolve code that implements an artificial neural network .

[4] A. Elyasaf, M. Orlov, and M. Sipper. A heuristiclab evolutionary algorithm for finch. In *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, GECCO '13 Companion, pages 1727–1728, New York, NY, USA, 2013. ACM.
This is a very short paper. I don't think I will be using it since it lacks substance. This paper is about using FINCH along side HeuristicLab to solve symbolic regression and other classic EC problems .

[5] M. Harman, W. B. Langdon, Y. Jia, D. R. White, A. Arcuri, and J. A. Clark. The gismoe challenge: Constructing the pareto program surface using genetic programming to find better programs (keynote paper). In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE 2012, pages 1–14, New York, NY, USA, 2012. ACM.
This paper looks very promising in providing information on evolving programs (not nessesarly using bytecode though). Also, it references FINCH .

[6] W. B. Langdon and M. Harman. Genetically improving 50000 lines of c+. *RN*, 12(09):09, 2012.
This is research notes about using using EC to evolve and improve C++ programs. Since it is research notes it is a bit lengthy. It also mentions evolving things through bytecode. Not sure if I will use yet .

[7] M. Orlov and M. Sipper. Flight of the finch through the java wilderness. *Evolutionary Computation, IEEE Transactions on*, 15(2):166–182, April 2011.
This will be one of the backbones of my paper. Talks about a program called FINCH that the authors developed. Finch is used to evolve programs at the Java bytecode level. This paper Has lots of information and they have about six examples where they evolved programs using FINCH .

UMM CSci Senior Seminar Conference Morris, MN.

[8] E. Schulte, S. Forrest, and W. Weimer. Automated program repair through the evolution of assembly code. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 313–316, New York, NY, USA, 2010. ACM. *This will be a main source for my paper. It talks about evolving programs at assembly language level and its benefits. These authors focus more on doing this to use EC to debug, refactor, and repair programs. It also breifly mentions FINCH* .