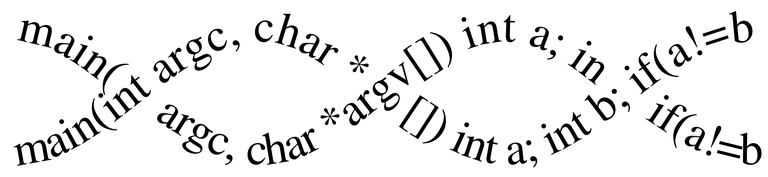
Automated Program Repair through the Evolution of Assembly Code



Eric Schulte[1] eschulte@cs.unm.edu
Stephanie Forrest[1] forrest@cs.unm.edu
Westley Weimer[2] weimer@virginia.edu

[1] *Dept. of Computer Science, University of New Mexico*
[2] *Dept. of Computer Science, University of Virginia*

Summary

We demonstrate a method for *automated bug repair* in extant software using *evolutionary computation*. The technique is applicable to any program that can be compiled to x86 assembly or Java byte code.

Motivation

By some estimates *software maintenance and repair* represents *90% of the lifetime cost* of a typical software project.

Benefits

General: Applicable to any language that compiles to Java byte code or x86 assembly code

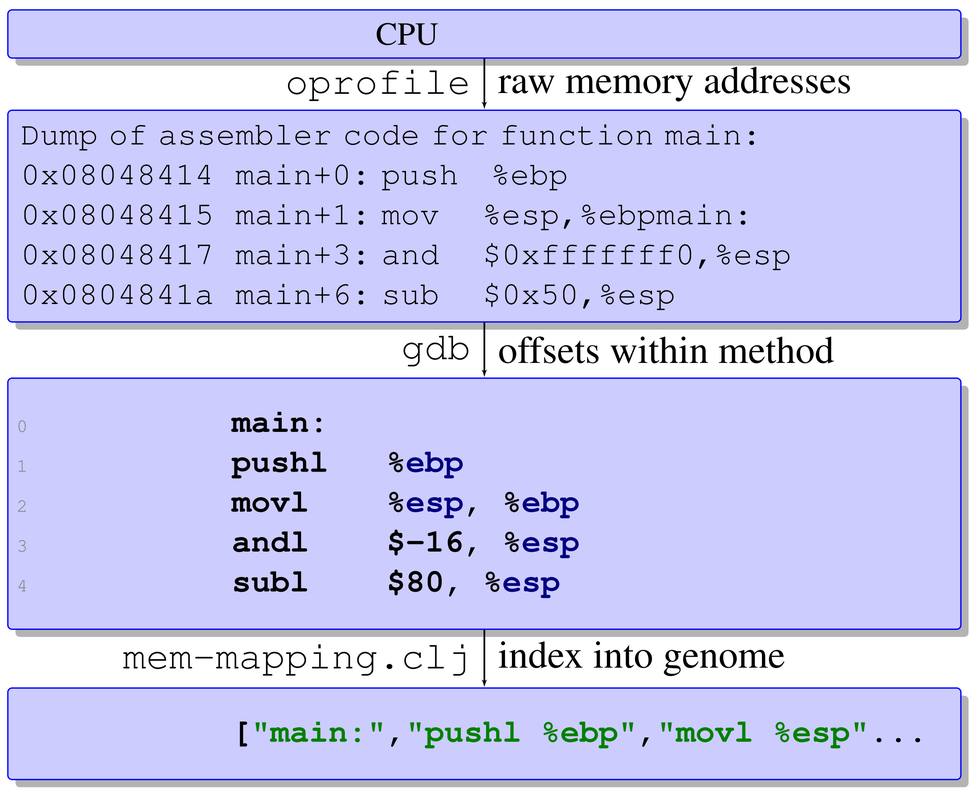
Expressive: The small scale of assembly instructions is capable of expressing repairs not possible at the C statement level

Robust: The functionality of assembly programs is surprisingly robust to random mutation

Coverage: The small alphabet of assembly commands and large number of commands in assembly programs provides access to a large subset of the space of possible programs

Statistical Fault Localization

We localize faults by combining statistical sampling of PC addresses with object file mappings from addresses to instructions. The sampled instructions are then used to weight the elements of the program genome.



Representation and Genetic Operations

Individuals are *linear genomes* of weighted assembly commands.

```
("main:" "pushl %ebp" "movl %esp" ... )
```

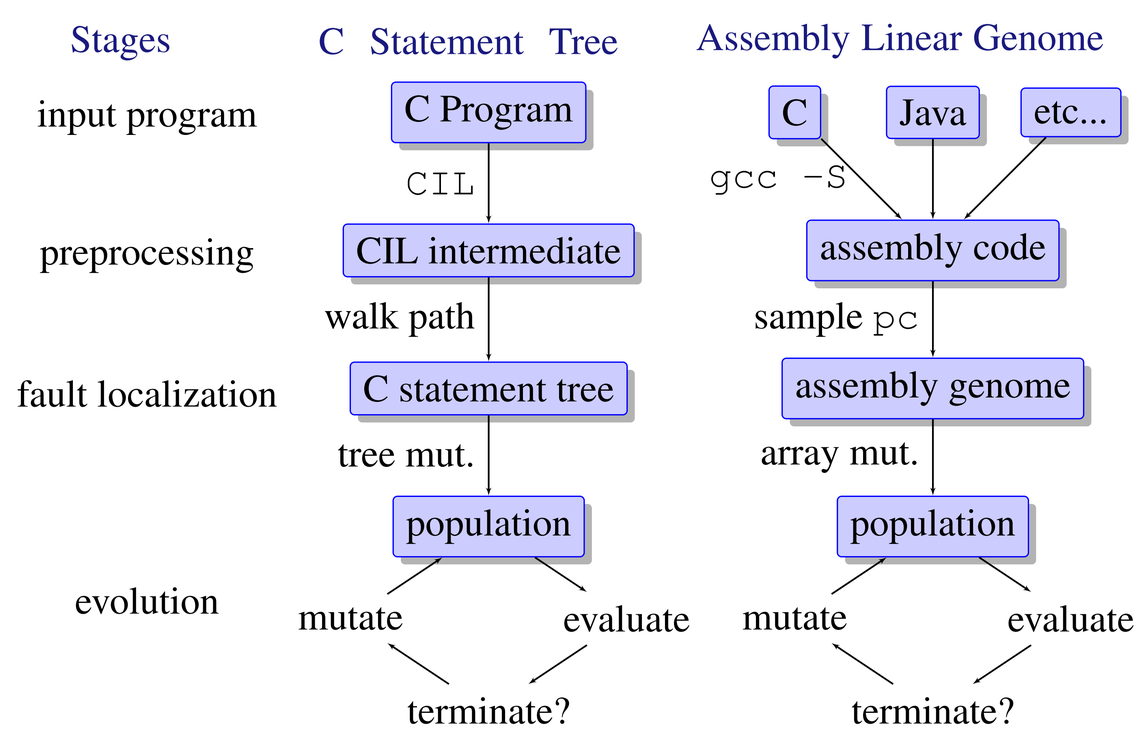
insert: selects an instruction, selects a location, copies the instruction and inserts in the location

delete: selects an instruction and deletes it

swap: selects two instructions and swaps them

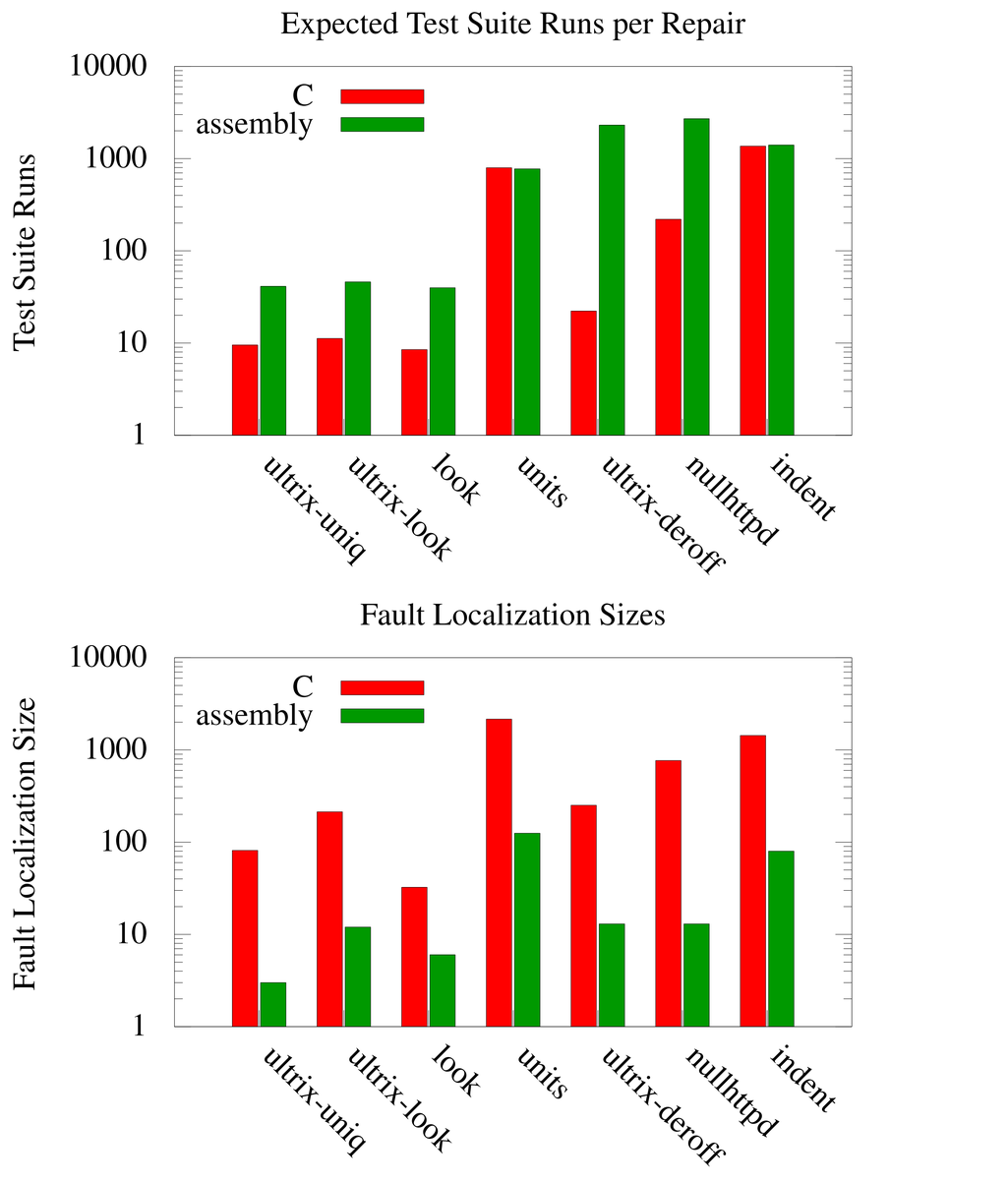
crossover: selects a point in each of two individuals and exchanges all instructions after that point

Technical Approach



Experimental Results

- Fixed 7 documented bugs in 6 Unix utilities and 1 web server totalling 22899 lines of assembly code
- Operating at the assembly level is 17% slower than at the level of C abstract syntax trees
- Fault localization in assembly code is significantly more precise than in C abstract syntax trees
- Discovered systematic differences between languages in the amenability of their compiled assembly code to evolution



	C	Haskell	Java
Program Length	79	885	33
Unique Solutions	2	10	1

'Program length' refers to the total number of assembly-code instructions of the baseline individual.

'Unique Solutions' is the total number of unique solutions found in 500 total runs.