SIRP: Signed, Random Programs - A Proposed Method for Computer Platform Testing

Samuel A. Kirk
MrBesterTester@gmail.com
August 12, 2009

Signed, Random Programs (SIRP) is a signature style test method for computer platforms that leverages constrained random program generation tools from compiler testing, the Valgrind utility and standard checksum/hash-code algorithms. SIRP is intended as a screening tool for the manufacturing test of a population of computers all of the same model and configuration.

A random program generation tool (such as randprog [1], explained in [2]) is used to generate a suite of testing programs randomly with suitable constraints. (The testing programs themselves are randomly generated; they are **not** simply programs that are random number generators.)

Each such testing program has a 4-part signature consisting of:

1. a serial number that uniquely identifies it within the suite of randomly generated testing programs
2. a static hash-code [3,4] representing its executable text and static data,
3. a dynamic hash-code representing the its computational results and those of its intermediates (subroutines), and
4. the execution time it takes (duration not elapsed).

Items (2) and (3) are a mutually exclusive, jointly exhaustive partitioning of a given testing program's value-bearing components, i.e., its memory and register contents. Together they represent the computational spatial envelope of the testing program. Item (4) adds a purely time representation (unlike item (3) which is a mixture of both time and space). All three together yield a complete representation of a testing program's computational boundary. Individual machine descriptors and information such as the platform's serial number and any MACs are excluded from the signature.

Using Valgrind [5] (as inspired by [2]), the hash-codes are instrumented at the binary level, independently of the programming language(s) in which the testing programs are generated.

A comparable platform-level signature is also calculated based upon the signatures of all the testing programs in the suite.

At least one known good computer is designated as the reference (REF) upon which is generated the REF signature. Consistency is checked on the platform under test (PUT) by comparing its signature with that of the REF: the PUT passes if their signatures match; fails otherwise.

Failure analysis is routinely performed by comparing the signatures of individual testing programs from the PUT with their counterpart on the REF to identify which specific test programs failed. In the case of a dynamic hash-code failure, further analysis of the failing testing program may be performed by compared the execution of the failing program on the PUT with that on the REF using an approximately binary search with debuggers (hard or soft) or with other features of Valgrind.

REFERENCES

[1] Eric Eide & John Regen, Compiler-Testing Tools, randprog (v1.0.0, Oct 20, 2008), http://www.cs.utah.edu/%7Eeeide/emsoft08

[2] Eric Eide & John Regen, Volatiles Are Miscompiled, and What to Do about It, EMSOFT (Oct 2008), http://www.cs.utah.edu/~regehr/papers/emsoft08-preprint.pdf

[3] Wikipedia, Checksum, http://en.wikipedia.org/wiki/Checksum

[4] Wikipedia, List of Hash Functions, http://en.wikipedia.org/wiki/List_of_checksum_algorithms

[5] Valgrind, http://valgrind.org