Concolic Testing 을 사용한 Smartphone SW 검증: Busybox 사례 연구

Yunho Kim

(with Liu Yuyang and Moonzoo Kim)

Provable Software Lab, CS Dept., KAIST

Why Concolic Testing?

- Our customer(Samsung Electronics) want
  - Automated V & V techniques
  - Techniques applicable to an embedded C program as is
  - Techniques which help to detect a subtle error
  - No false positives

- Concolic testing is an automated test generation technique through combined CONCrete + symbOLIC execution.

Why Busybox?

- Open-source C program
  - Confidential issues
- Widely used in embedded systems
  - Also it is used in smartphones
- Clear specification of inputs and outputs
  - Functions of applets in Busybox are well-known

Busybox

- The Swiss army knife for embedded Linux

- It provides many standard UNIX/Linux command line utilities
  - 19 groups, 341 applets in Busybox 1.17
  - Core utils, text editors, shells, network utils, and so on
  - Configurable at compile time

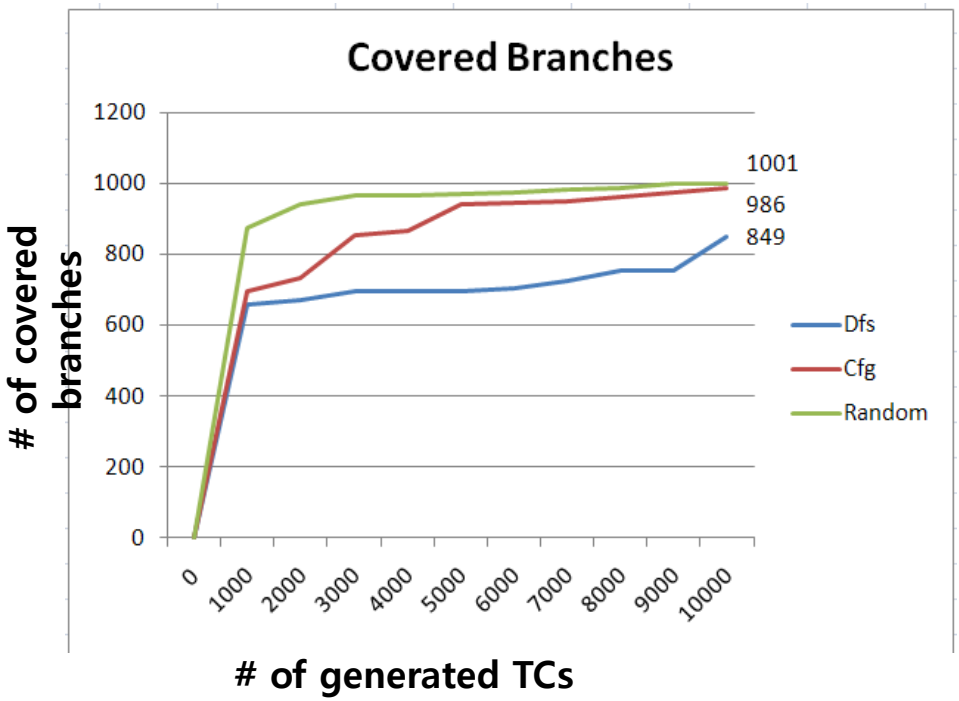
What We Did

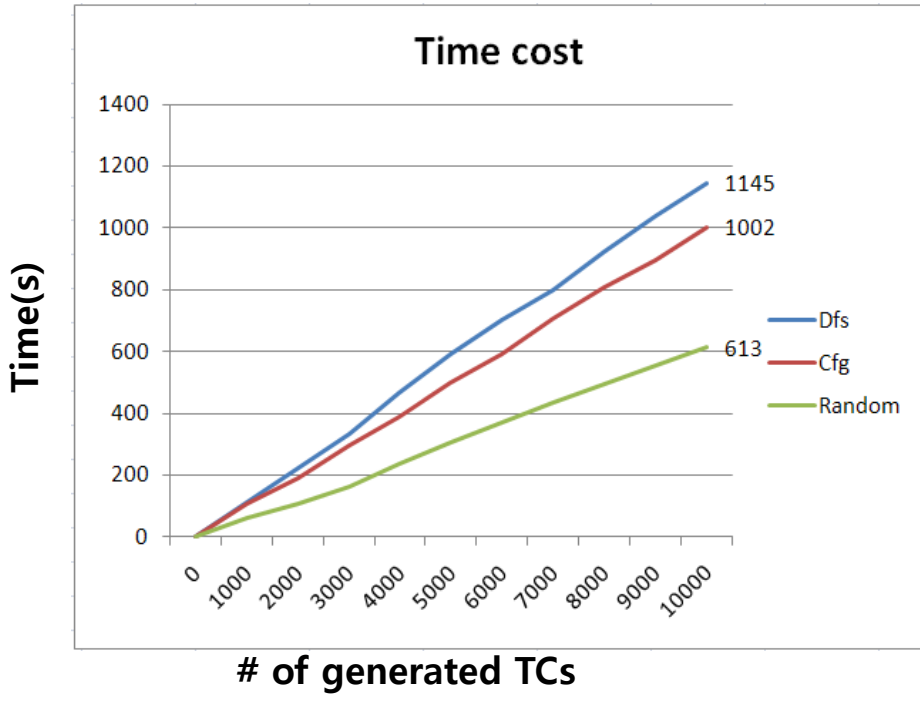
- We applied the concolic testing technique to Busybox
  - ‘grep’ and ‘vim’ are chosen among 341 applets
  - 771LOC and 4000LOC, respectively

- Experimental setup
  - Intel Core2duo 3.0GHz with 4GB memory
  - Fedora 8 32bit version
  - CREST 0.1.1 with Yices 1.0.26

Results

Strategy	DFS	CFG	Random
Covered branches of **grep.c** Total: 178	109 (61.2%)	87 (48.9%)	120 (67.4%)
Covered branches of **vi.c** Total: 1476	817 (55.4%)	924 (62.6%)	957 (64.8%)





Future Work

- Apply concolic testing to more applets
  - sed, tr, sort, and so on

- Improve branch coverage by utilizing manual test cases

Results

- Grep results

Strategy	DFS	CFG	Random	merge
Total covered branches	209	184	248	248
Covered branches of grep.c Total: 178	109 (61.2%)	87 (48.9%)	120 (67.4%)	120 (67.4%)

- Vi results

Strategy	DFS	CFG	Random	merge
Total covered branches	849	986	1001	1015
Covered branches of vi.c Total: 1476	817 (55.4%)	924 (62.6%)	957 (64.8%)	979(66.3%)