KLEE Symbolic Execution Tool Test-Comp 2019 Entry

Cristian Cadar and Martin Nowack
Department of Computing
Imperial College London

Imperial College London

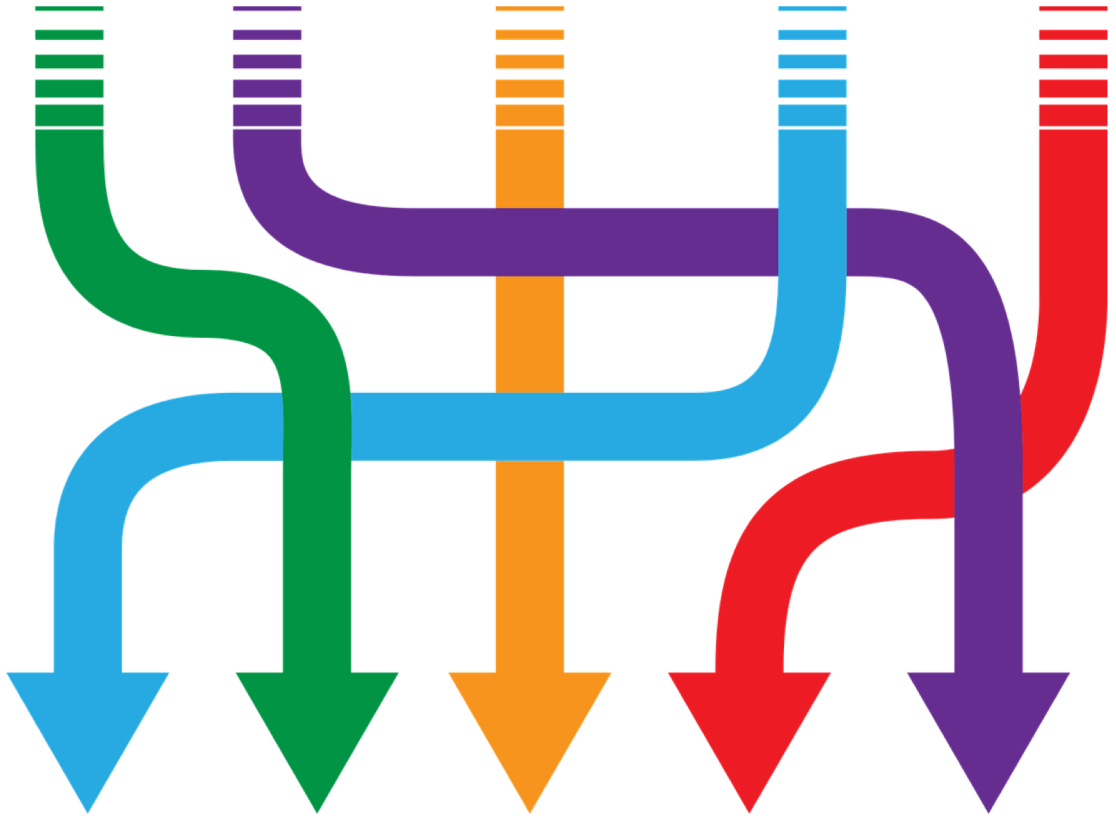
1st Test-Comp, 6 April 2019
Prague, Czech Republic

KLEE uses Dynamic Symbolic Execution

Technique for automatically exploring multiple paths, reasoning about path feasibility using a constraint solver.

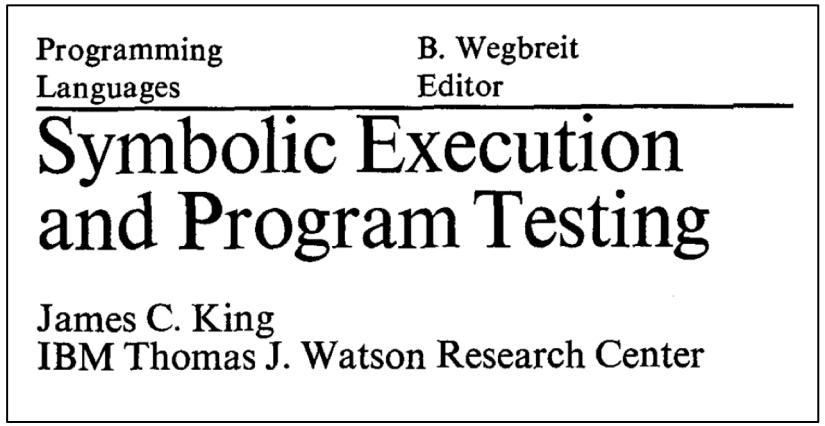
Applications in:

- Bug finding
- Test generation
- Vulnerability detection and exploitation
- Equivalence checking
- Debugging
- Program repair
- etc. etc.



A Bit of History: 1975-76

- Symbolic execution developed independently by several researchers

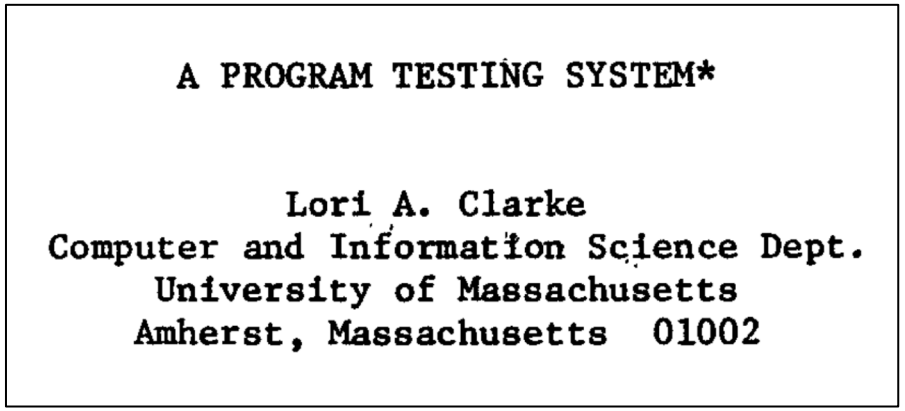


Programming Languages — B. Wegbreit Editor

Symbolic Execution and Program Testing

James C. King
IBM Thomas J. Watson Research Center

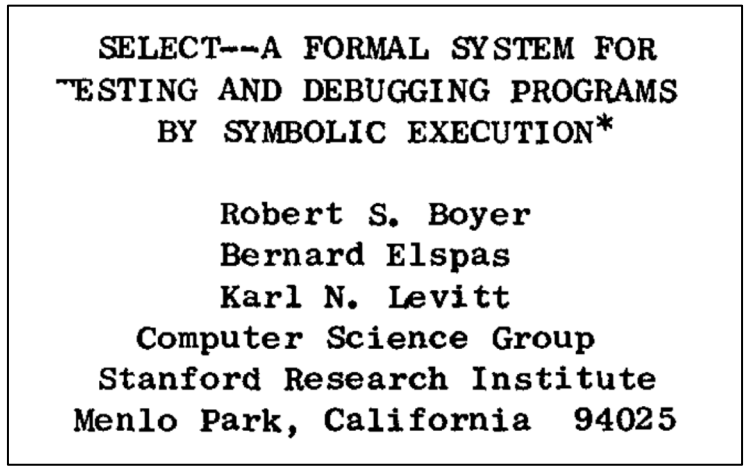
Symbolic execution of PL/I programs



A PROGRAM TESTING SYSTEM*

Lori A. Clarke
Computer and Information Science Dept.
University of Massachusetts
Amherst, Massachusetts 01002

Symbolic execution of Fortran programs



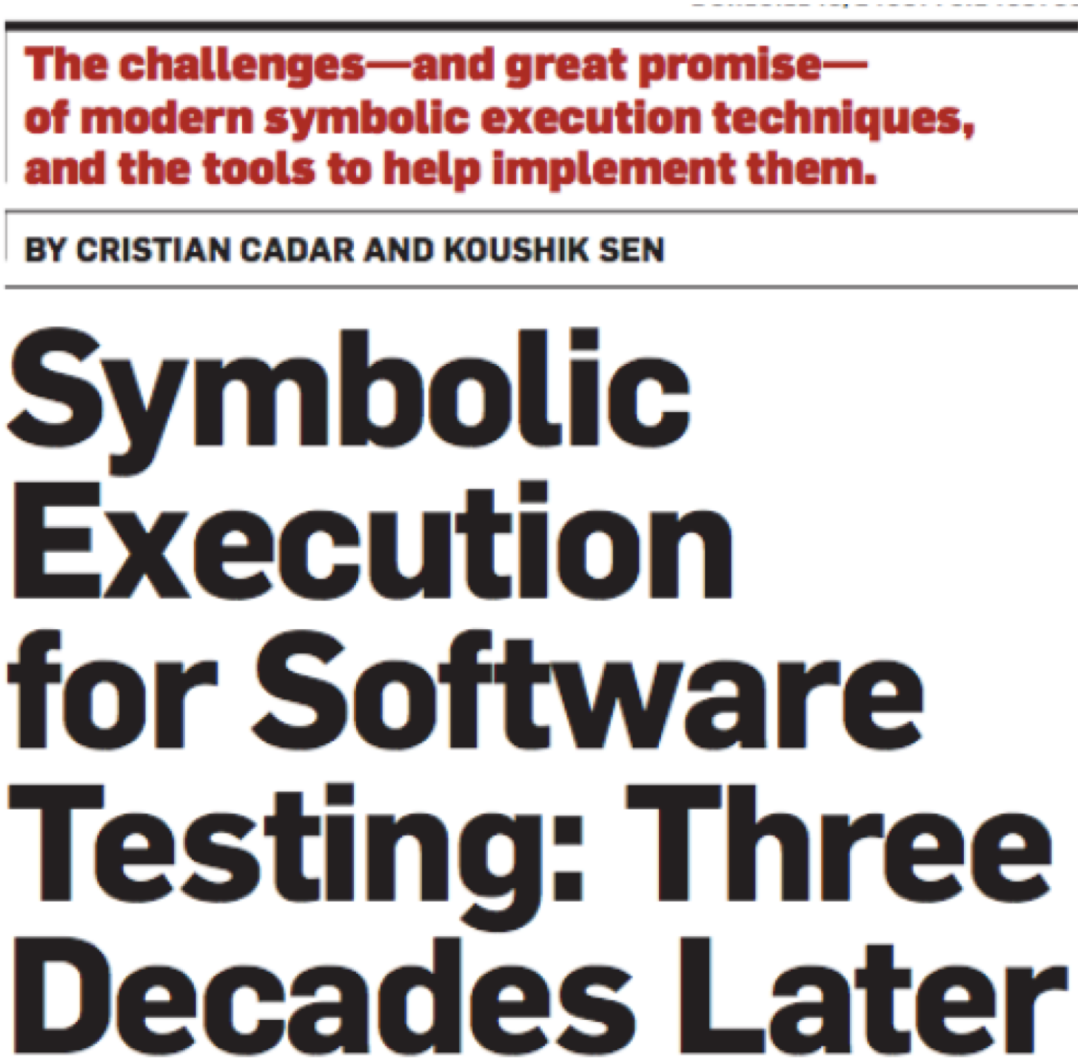
SELECT--A FORMAL SYSTEM FOR TESTING AND DEBUGGING PROGRAMS BY SYMBOLIC EXECUTION*

Robert S. Boyer
Bernard Elspas
Karl N. Levitt
Computer Science Group
Stanford Research Institute
Menlo Park, California 94025

Symbolic execution of LISP programs

1975-76

A Bit of History



The challenges—and great promise—
of modern symbolic execution techniques,
and the tools to help implement them.

BY CRISTIAN CADAR AND KOUSHIK SEN

Symbolic Execution for Software Testing: Three Decades Later

A Bit of History

- Mixed concrete-symbolic execution

- Availability of precise runtime information
- Ability to interact with the environment
- Partly deal with limitations of solvers
- Only relevant code executed symbolically

DART: Directed Automated Random Testing

Patrice Godefroid Nils Klarlund
Bell Laboratories, Lucent Technologies
{god,klarlund}@bell-labs.com

Koushik Sen
Computer Science Department
University of Illinois at Urbana-Champaign
ksen@cs.uiuc.edu

DART system for C

Execution Generated Test Cases: How to Make Systems Code Crash Itself

Cristian Cadar and Dawson Engler*

Computer Systems Laboratory
Stanford University
Stanford, CA 94305, U.S.A.

EGT system for C

2005

Fourteen Years Later

KLOVER

Miasm

PyExZ3

SymDroid

PathGrind

jCUTE

Otter

SAGE

CREST

SymJS

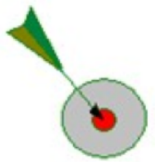
BinSE

Jalangi2

CUTE

Symbolic PathFinder

angr



DART

Kite

LDSE

Pex

Rubyx

JDart

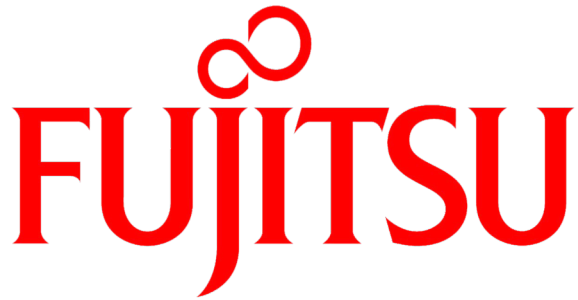
S²E

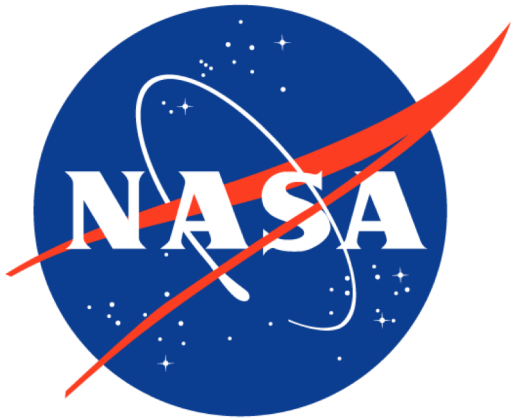
CATG

CiVL

Mayhem

Outside Academia





- Symbolic execution for LLVM bitcode, primarily targeting C
- Started at Stanford [Cadar, Dunbar, Engler, OSDI 2008], incorporating the lessons from our prior system EXE
- Evolved to incorporate many of the ideas developed in the last decade

KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs

Cristian Cadar, Daniel Dunbar, Dawson Engler *
Stanford University

[OSDI 2008]

2008 – present

Daniel Dunbar
principal author of original codebase

Dawson Engler

KLEE – Contributions

- Complete redesign of EXE, incorporating lessons learnt
  - Cadar, Ganesh, Pawlowski, Engler, Dill [CCS 2006 & TISSEC 2008]
  - ACM CCS Test of Time Award 2016
- Based on state-of-the-art LLVM compiler infrastructure
  - Extensible to other languages beside C (has partial support for C++)
- Efficient object-level COW scheme: can keep tens of thousands states in memory
- Pluggable search heuristics
- Extensible chain solver architecture
  - Easy to add partial solvers to the chain, as well as new SMT solvers
- Models for symbolic files and command-line arguments
  - E.g., klee … –sym-args 4 100 -sym-files 2 10 …
- Many constraint-solving and path-exploration optimizations

Webpage: **klee.github.io**
Code: **https://github.com/klee**
Web version: **http://klee.doc.ic.ac.uk**

KLEE is open source

- Available on GitHub under a liberal license (UIUC)
- Currently maintained in the SRG group at Imperial, particularly by me and Martin Nowack
- Recently released version 2.0 in March 2019 (Test-Comp version similar)

Widely-used in both industry and academia

- 1000+ stars
- 300+ forks
- 50+ contributors
- 350+ subscribers to mailing list

KLEE – Academic Impact

- Over 2,200 citations to [OSDI 2008] (Google Scholar, Apr. 2019)
- From many different fields (in no particular order):
  - Testing: ISSTA, ICST, etc.
  - Verification: CAV, HVC, etc.
  - Systems: OSDI, SOSP, EuroSys, etc.
  - Software engineering: ICSE, FSE, ASE, etc.
  - Security: CCS, IEEE S&P, USENIX Security, etc.
  - etc.

KLEE – Academic Impact

- Several well-known systems use KLEE, e.g.:
  - Cloud9 and S2E from EPFL
  - KLEE-UC from Stanford
  - KleeNet from Aachen; etc.
  - AEG from CMU,
  - GKLEE from Utah
  - BugRedux and F3 from GeorgiaTech
  - Symbiotic from Masaryk University
  - SemFix and Angelix from NUS
  - ZESTI and KATCH from Imperial
  - etc.
- http://klee.github.io/publications/ lists over 130 publications/systems that directly extend or use KLEE

KLEE – Impact Outside Academia

Many known uses or trials:

Some publications/talks/blog posts:

- Fujitsu: [PPoPP 2012], [CAV 2013], [ICST 2015], [IEEE Software 2017], [KLEE Workshop 2018]
- Hitachi: [CPSNA 2014], [ISPA 2015], [EUC 2016]
- Trail of Bits: https://blog.trailofbits.com/
- Intel: [WOOT 2015]
- NASA Ames: [NFM 2014]
- Baidu: [KLEE Workshop 2018]

KLEE Workshop 2018

Organization Sponsors

1st International KLEE Workshop on Symbolic Execution

19-20 April, 2018 • London, United Kingdom

Tweets by @kleesymex

kleesymex @kleesymex
Save the date! 1st International KLEE

1st International KLEE Workshop 2018

- 80+ participants from 12 different countries
  (Had to close registration early)
- Representing both academia, industry and government
- Sponsored by EPSRC, Baidu, Bloomberg, Fujitsu, Huawei and Imperial
- Next edition in 2020

KLEE at Test-Comp

- Competitions have an important role to play in advancing research and adoption
- SAT-COMP and SMT-COMP are great examples of impactful competitions
- Thanks to Dirk for the initiative

KLEE at Test-Comp

- Most effort went in understanding the infrastructure and tasks
  - Thanks to Dirk and his team for answering our many questions
- Configured KLEE to run with:
  - LLVM 6.0 (currently supports 3.4 – 8.0)
  - STP (currently supports Boolector, CVC4, Yices2, and Z3, some via metaSMT)
- Made several modifications to KLEE for Test-Comp
  - Extended it to support the XML-based test cases of Test-Comp
  - Configured it differently for the bug finding and coverage categories
  - Fixed some bugs revealed by benchmarks
- Main limitation
  - Does not support FP (although two FP extensions of KLEE exist)
  - Unsurprisingly, it scored poorly on the FP benchmarks

Test-Comp Benchmarks (afternoon session?)

Test-Comp Benchmarks: The Pros

- Cover many different features, discovered a couple of interesting bugs in KLEE
- (Mostly) no undefined behavior
  - Benchmarks with UB that we and others discovered have been generally fixed or removed
- Those in the bug category have specs for the bugs that need to be found
- Most are easy to understand
  - Although some automatically generated ones (or preprocessed via CIL) are not

Test-Comp Benchmarks: The Cons

- Inherited from SV-COMP, so have a verification twist
- Quite small overall
  - 10 to 184,969 ELOC, but with a median of only 1,409 ELOC
  - By contrast, benchmarks used to evaluate testing tools are considerably larger
- Many are hand-crafted
  - Interesting cases, but not always representative of real code
- Involve very large inputs
  - Arrays of 100,000 elements or unrestricted sizes is typical, but this is not representative of how testing tools are usually used

Test-Comp Benchmarks: Going Forward

- New and larger benchmarks would improve future editions
- Main challenge is to incorporate them in the current infrastructure
  - Multi-file benchmarks with complex build systems
  - Varied inputs such as command-line arguments and files
  - Undefined behavior
  - Lack of specifications
- Types of benchmarks
  - File processing apps? Command-line apps?
  - Coverage-only category?
  - Fault injection?

KLEE Symbolic Execution Tool Test-Comp 2019 Entry

Cristian Cadar and Martin Nowack
Department of Computing
Imperial College London

Imperial College London

1st Test-Comp, 6 April 2019
Prague, Czech Republic