Regression Tests
and the
Inventor’s Dilemma

Leonardo de Moura
Microsoft Research
How do we make progress?

We are trying to solve HARD problems
Automated Reasoning Tools use heuristics
Progress is not monotonic
How do we release new versions?
How do we find bugs?
Regression tests

Input

Tool

Expected Output

Produced Output
Regression tests: the 1\textsuperscript{st} directive

Every bug report becomes a regression test
Fuzzer

“Random” input generator
Syntactically valid input
It is not trying to expose bugs in the parser
Runtime assertions

Input

Inefficient Version

Efficient Version

Output 1

Output 2
External Oracle

Input

Procedure

Output, Certificate

Oracle
Example: Oracles in nlsat
Example: Oracles in \text{nlsat}

Current Assignment
\[ x = \sqrt{2}, \quad y \rightarrow 1 \]

Conflicting Core
\[ \neg z < 0, \]
\[ 2z^2 - y^2 = 0, \]
\[ \neg x z - y = 0 \]
Example: Oracles in nlsat

Current Assignment

\[ x \rightarrow \sqrt{2}, \quad y \rightarrow 1 \]

Conflicting Core

\[ \neg z < 0, \]
\[ 2z^2 - y^2 = 0, \]
\[ \neg x \ z - y = 0 \]

\[ z \rightarrow -\frac{1}{\sqrt{2}}, \quad z \rightarrow \frac{1}{\sqrt{2}} \]
Example: Oracles in \textit{nlsat}

Current Assignment

\begin{align*}
  x & \rightarrow \sqrt{2}, \\
  y & \rightarrow 1
\end{align*}

Conflicting Core

\begin{align*}
  \neg z & < 0, \\
  2z^2 - y^2 & = 0, \\
  \neg x z - y & = 0
\end{align*}

\[
  z \rightarrow -\frac{1}{\sqrt{2}}, \quad z \rightarrow \frac{1}{\sqrt{2}}
\]
Example: Oracles in $\text{nlsat}$

Current Assignment

$x \rightarrow \sqrt{2}, \quad y \rightarrow 1$

Conflicting Core

$\neg z < 0, \quad 2z^2 - y^2 = 0, \quad \neg x \quad z - y = 0$

\[ z \rightarrow -\frac{1}{\sqrt{2}}, \quad z \rightarrow \frac{1}{\sqrt{2}} \]
Example: Oracles in \textit{nlsat}

\begin{align*}
\text{Current Assignment} & \quad \text{Conflicting Core} \\
\sqrt{2} & \quad \neg z < 0, \\
\sqrt{2} & \quad 2z^2 - y^2 = 0, \\
\sqrt{2} & \quad \neg xz - y = 0
\end{align*}

\text{To Oracle: Mathematica}

\text{Resolve}\left[\forall\{x, y, z\}, \neg (x = \text{Root}[#1^2 - 2 &, 2] \land y = 1) \land \\
\quad z < 0 \land \\
\quad \neg(2z^2 - y^2 = 0) \land \\
\quad xz - y = 0, \right] \text{Reals}
“Telemetry”

“Call home” feature.
Collect stats from every run.
Store stats in a server.
Inventor’s Dilemma

“All the new version is slower.”

“All the new version fails on my problem.”
SDV: STATIC DRIVER VERIFIER

SLAM

Ella Bounimova, Vlad Levin, Jakob Lichtenberg, Tom Ball, Sriram Rajamani, Byron Cook, ...
Overview

http://research.microsoft.com/slam/

SLAM/SDV is a software model checker.
Ships with DDK
Application domain: *device drivers*.
Architecture:

- **c2bp**  C program → boolean program (*predicate abstraction*).
- **bebop** Model checker for boolean programs.
- **newton** Model refinement (check for path feasibility)

SMT solvers are used to:

- Perform predicate abstraction,
- Check path feasibility.

c2bp makes several calls to the SMT solver.
The formulas are relatively small.
SLAM/SDV Summary

Regression tests are extensively used

Rigid process for incorporating new modules

Several months to move from Z3 1.x → Z3 3.x

Long process for integrating Yogi
CLOUSOT: STATIC ANALYZER
Clousot checks the code as you type

0 It reports warnings and **verified code fixes**
Architecture

- Command Line
- VS 08,10,12
- Roslyn
- Warnings report
- Code repairs
- Contracts Propagation
- Assertion checking
- Facts Discovery
- IL, Contract Reader
- CCI1
- CCI2
- Roslyn

- Abstract Interpretation
Clousot/CodeContracts impact

- API .NET standard since v4
- Externally available
  - > 60,000 downloads
  - Active forum (>1,500 threads)
  - Book chapters, blogs ...
- Internal and External adoption
  - Mainly professional programmers
  - A few university courses
- Publications, talks, tutorials
  - Academic (POPL, OOPSLA, ECOOP, VMCAI, SAS ...)
  - Programmers conferences
Clousot Summary

Regression tests

Inventor’s dilemma scenario

User X invests time in the following loop:
- Inspect warnings
- Add more contracts
- Fix bugs

Finally the code is warning free

New version is released → New warnings

Some users use multiple versions
Clousot Summary

Inventor’s Dilemma Apocalypse
May ship as part of Visual Studio
Potential for millions of users
SAGE: TEST-CASE GENERATION
Test Generation is Big Business

- #1 application for SMT solvers today (CPU usage)
- **SAGE @ Microsoft:**
  - 1st whitebox fuzzer for security testing
  - 400+ machine years (since 2008)
  - 3.4+ Billion constraints
  - 100s of apps, 100s of security bugs
  - Example: Win7 file fuzzing
    - \( \sim 1/3 \) of all fuzzing bugs found by SAGE
      (missed by everything else...)
  - Bug fixes shipped (quietly) to 1 Billion+ PCs
  - Millions of dollars saved
    - for Microsoft + time/energy for the world
SAGAN: Fuzzing in the (Virtual) Cloud

• Since June 2010, new centralized server collecting stats from all SAGE runs!
  - 200+ machine-years of SAGE data (since June 2010)

• Track results (bugs, concrete & symbolic test coverage), incompleteness (unhandled tainted x86 instructions, Z3 timeouts, divergences, etc.)

• Help troubleshooting (SAGE has 100+ options...)

• Tell us what works and what does not
Picking and Choosing

• Typical SAGE run can fill up 300 GB in 1 week

• Problem: 100s of machines * 300+ GB = lots of data

• Solution: pick and choose what to ship up
  - Configuration files
  - Counters from each SAGE execution
  - Run “heartbeats” at random intervals
  - Crashing test information

• Key principles
  - Enough information to reprod SAGE run results
  - Support key analyses for improving SAGE
Sage Summary

Tool as a service

Extensive use of “telemetry”

Automatically find issues that are relevant for their customers
Satisfiability

\[ x^2 + y^2 < 1 \text{ and } xy > 0.1 \] 
\[ x^2 + y^2 < 1 \text{ and } xy > 1 \]

Is execution path \( P \) feasible?

Is assertion \( X \) violated?

Is Formula \( F \) Satisfiable?

Solution/Model

sat, \( x = \frac{1}{8}, y = \frac{7}{8} \)

unsat, Proof
Z3 is a collection of Symbolic Reasoning Engines

- DPLL
- Simplex
- Rewriting
- Superposition
- Congruence Closure
- Groebner Basis
- ∀∃ elimination
- Euclidean Solver
Z3 Impact

Z3 is used by many research groups (> 700 citations)
More than 18k downloads
Z3 placed 1st in 17/21 divisions in the SMT-COMP 2011
2007-2008 Competition-oriented years

Just check if it produces the right answer

2007  Z3 0.1 (SMT-COMP’07)  Z3 1.0 released later
Tested using 1 machine with 8 cores

2008  Z3 2.0 (SMT-COMP’08)
Small cluster with 12 cheap machines
Painful transition from Z3 1.x → Z3 2.x
2009-2010 Decline

Machines in the small cluster started dying
No regressions or measurements
Randomly adding features and fixing/adding bugs
No idea whether making progress or not
Many users consider Z3 2.19 much worse than 2.16
2011-2012 Revival Z3 3.x and 4.x
  Fuzzers running nonstop 24x7
  Rerun all SMT-LIB and key benchmarks every night
  Huge shared cluster
  Z3 3.0 won 17/21 divisions in SMT-COMP’11
  Z3 3.0 best 9/10 divisions in SMT-COMP’12
  Thousands of regression tests executed every night
    Testing several internal modules
    Testing exposed APIs
Next Steps...

Adding telemetry
More model/proof validation regression tests
More unit tests
Multiplatform testing: Linux and OSX versions
Monitoring system a-la Sagan
SMT-Lib benchmarks

Not good for testing corner cases
   Fuzzer is great for that
Manually written tests
Most benchmarks use only the basic SMT 2.0 features
New Fuzzer?
Z3 & Inventor’s Dilemma

“The new Z3 is 20% slower on my problem”
Come on, move on

“The new Z3 is 3x slower on my problem”
Does he work for Microsoft?

“The new Z3 is 10x slower on my problem”
Let me check what is going on
Z3 & Inventor’s Dilemma

Incorporate problems from key customers in the regression tests.

It is very hard to make progress.

Z3 2.x search engine is still there.

Be careful when adding obscure features.

Users will find and use them.
Inventor’s Dilemma: a Solution

Orchestrating Decision Engines

(more about it tomorrow at IWS)

Leonardo de Moura (Microsoft Research)
Grant Passmore (University of Cambridge)
What is a Strategy?

Theorem proving as an exercise of combinatorial search

Strategies are adaptations of general search mechanisms which reduce the search space by tailoring its exploration to a particular class of formulas.
Different Strategies for Different Domains.
The "Message"

SMT solvers are collections of little engines. They should provide access to these engines. Users should be able to define their own strategies.
Main inspiration: LCF-approach

- Tactic
- Goal
- Subgoals
- Proof builder
Main inspiration: LCF-approach

Proofs for subgoals

Proof for goal

Proof builder

Tactic

Proof builder

goal

subgoals
Main inspiration: LCF-approach

Diagram:
- Goal
- Tactic
- Proof builder
- Tactic
- Proof builder
- Tactic
- Proof builder
- Tactic
Main inspiration: LCF-approach
Main inspiration: LCF-approach

- Proof Builder
- Proof Builder
- Proof Builder

thm in LCF terminology

proof in LCF terminology
Tacticals aka Combinators

then( Tactic , Tactic ) = Tactic

orelse( Tactic , Tactic ) = Tactic

repeat( Tactic ) = Tactic
SMT Tactic

goal → Tactic → Proof builder

subgoals

Model builder
\[
\text{goal} = \text{formula sequence} \times \text{attribute sequence}
\]
\[
\text{proofconv} = \text{proof sequence} \rightarrow \text{proof}
\]
\[
\text{modelconv} = \text{model} \times \text{nat} \rightarrow \text{model}
\]
\[
\text{trt} = \text{sat model}
\]
\[
\text{trt} = \text{unsat proof}
\]
\[
\text{trt} = \text{unknown goal sequence} \times \text{modelconv} \times \text{proofconv}
\]
\[
\text{trt} = \text{fail}
\]
\[
\text{tactic} = \text{goal} \rightarrow \text{trt}
\]
end-game tactics:
never return unknown(sb, mc, pc)
non-branching tactics:

sb is a singleton in unknown(sb, mc, pc)
Trivial goals

Empty goal [ ] is trivially satisfiable

False goal [ ..., false, ...] is trivially unsatisfiable

basic : tactic
SMT Tactic example

\[ a = b + 1, (a < 0 \lor a > 0), b > 3 \]

Tactic: elim-vars

\[ (b + 1 < 0 \lor b + 1 > 0), b > 3 \]

Proof builder

Model builder
SMT Tactic example

Proof builder

\[ a = b + 1, \ (a < 0 \lor a > 0), \ b > 3 \]

Tactic: elim-vars

Tactic: elim-vars

\[ (b + 1 < 0 \lor b + 1 > 0), \ b > 3 \]

Model builder

M, M(a) = M(b) + 1

M
SMT Tactic example

\[ a = b + 1, \ (a < 0 \lor a > 0), \ b > 3 \]

Tactic: split-or

Proof builder

[ \ a = b + 1, a < 0, b > 3 \ ]

[ \ a = b + 1, a > 0, b > 3 \ ]

Model builder
then : \((tactic \times tactic) \rightarrow tactic\)

\(\text{then}(t_1, t_2)\) applies \(t_1\) to the given goal and \(t_2\) to every subgoal produced by \(t_1\).

then\(^*\) : \((tactic \times tactic \text{ sequence}) \rightarrow tactic\)

\(\text{then}(t_1, [t_{2_1}, ..., t_{2_n}]\)) applies \(t_1\) to the given goal, producing subgoals \(g_1, ..., g_m\). If \(n \neq m\), the tactic fails. Otherwise, it applies \(t_{2_i}\) to every goal \(g_i\).

orelse : \((tactic \times tactic) \rightarrow tactic\)

\(\text{orelse}(t_1, t_2)\) first applies \(t_1\) to the given goal, if it fails then returns the result of \(t_2\) applied to the given goal.

par : \((tactic \times tactic) \rightarrow tactic\)

\(\text{par}(t_1, t_2)\) executes \(t_1\) and \(t_2\) in parallel.
then(preamble, orelse(mf, pb, bounded, smt))

Simplification
Constant propagation
Interval propagation
Contextual simplification
If-then-else elimination
Gaussian elimination
Unconstrained terms
Conclusion

Regression tests are extensively used at MS “Telemetry”

Analyze your data

Inventor’s Dilemma is a major issue for any tool based on heuristics.

Gets worse as complexity increases
NP, PSPACE, NEXPTIME, Undecidable

Our partial solution:
Orchestrating Decision Engines