Maximising Axiomatization Coverage and Minimizing Regression Testing Time
Who guards the guardians?

How to improve trust in formal verification systems?

Modern verification systems are large and complex systems
• Soundness bugs are not rare
• Such bugs are often hard to detect in a real proof
“Auto-active” Verification Systems

Validating verification systems by
- Formal methods
- Code inspection
- Testing
- ...
Program Language Semantics

Static checkers  Verifying compilers  Logic frameworks

We have to test both!
But how to determine the quality of the test cases?
Test Cases

A test case is a program $P$, together with requirement and auxiliary specifications.

Manually creating test cases is extremely time-consuming.

Computing coverage for the test cases takes from a few minutes to several hours.
Case study: The KeY System
The KeY System

- Deductive verification system for JavaCard
- Sequent calculus for Java Dynamic Logic, uses symbolic execution for Java programs
- Interactive verification with automatic proof mode

Important
- The semantics of JavaCard is encoded in 1520 axioms ("small, well-understood set of sentences")
Coverage Results (naïve, TAP 2013)

The 319 completeness tests of KeY covered 40% of all axioms (611 out of 1520).
Heuristic Approaches
Reusing Test Cases

Idea: given a test case $T$, run the tool with just a subset of the 1520 axioms.

*drop one essential axiom*

Axioms / \{axiom_1\}
Axioms / \{axiom_2\}
Axioms / \{axiom_3\}

New coverage
Reusing Test Cases

Three simple heuristics to pick the “next axiom to drop”:

1. **Base case**
   611 (40%)

2. **Depth-first**
   701 (46%)

3. **Depth-first, random selection**
   699 (46%)

4. **Greedy (try to remove groups)**
   688 (45%)

5. **Breadth-first**
   687 (45%)

6. **Breadth-first, random selection**
   684 (45%)

Note:
- 24h per heuristic per test case
- Extremely fragile

1520 axioms

1460 axioms

Complimentary by design, verified by experiments.
Maximising Coverage & Minimising Time

- All test cases:
  - 20,000 test cases
  - Runtime: days
  - 48% coverage

- Fast Regr. Testing:
  - 100 test cases
  - Runtime: 1 h
  - 48% coverage

Originally:
- 319 test cases
- 40% coverage
Clustering Results

Problem understanding!
Test Case Selectivity

Only specific test cases, or test cases with broad coverage for an axiom may not be sufficient.
Completeness Coverage

Definition (Completeness Coverage, TAP 2013)
A test case $P + (REQ \cup AUX)$ covers the set of Axioms if
- $\text{Axioms} \vdash P + (REQ \cup AUX)$
- and this does not hold for $\text{Axioms}' \subsetneq \text{Axioms}$

Note: covered set Axioms is not uniquely defined by the test case
Computing Completeness Coverage

Given: set of axioms $Ax$ and completeness test case $T$
Result: completeness coverage by $T$

1. Run tool on $T$, record resource consumptions (to get upper limit for the subsequent runs).
2. If available, analyse proof artifacts to restrict the next step to a subset of $Ax$.
3. Remove stepwise from $Ax$ and check if proof is still valid.
   [repeat until a fix-point is reached]

Computing coverage for most test cases takes from a few minutes to several hours.