Improved Automation for SPARK Verification Conditions

Paul Jackson and Grant Olney Passmore
Paul.Jackson@ed.ac.uk, G.Passmore@ed.ac.uk

Background

SPARK is a subset of Ada which adds in assertions such as pre-conditions, post-conditions and loop invariants. It is used for high-integrity applications in the aerospace, defense, rail and security industries. The current automatic prover for SPARK program verification conditions (VCs) typically proves fewer than 100% of VCs. Proving the rest by manual means or interactive prover takes specialist skills and is very tedious. Typically assertions are restricted to checks of exception freedom in order to keep the level of proof automation reasonable.

Provers Considered

- **Z3**: SMT solver from Microsoft
- **Yices**: SMT solver from SRI
- **CVC3**: SMT solver from NYU and U. Iowa.
- **Simplify**: Legacy prover. Used in ESC/Java.
- **Simplifier**: Current VC prover from Praxis UK.

VCs are translated into a variety of different formats. The translation is 2 phase, firstly to a prover-independent standard form with full typing information, secondly to particular prover languages. The translator can work in an offline mode where translated VCs are written to files or an online mode where solvers are immediately invoked on the translated VCs. For the online mode, results are output in a database-friendly tabular format, and analysis is aided by command-line functions for merging and filtering tables.

VC Translator

Tokeneer is a just-released case study commissioned by NSA for evaluating Praxis’s SPARK-based high-integrity software development methodology.

Simplifier’s better coverage is because of its better support for non-linear arithmetic.

Provers Considered

- **Z3**: SMT solver from Microsoft
- **Yices**: SMT solver from SRI
- **CVC3**: SMT solver from NYU and U. Iowa.
- **Simplify**: Legacy prover. Used in ESC/Java.
- **Simplifier**: Current VC prover from Praxis UK.

VC Examples

- Proven only by Cvc3 and Simplifier
  - \( C_1: (x \mod y) \cdot (y \mod x) \leq 214783647 \)
- Proven only by Simplifier
  - \( C_1: (x \mod y) \cdot (y \mod x) \leq 214783647 \)
- Unproven by any
  - \( C_1: \langle 100 \times f \rangle \div (f + x) \leq 100 \)

Remarks on Experiments

- **Supplementary axioms**: It was useful to provide such axioms as
  - \( \forall x, y: z, 0 < y \Rightarrow 0 \leq x \mod y \)
  - \( \forall x, y: z, 0 \leq x \land 0 < y \Rightarrow y \times (x \div y) \leq x \)
- **Soundness**: Simplify will ‘prove’
  - \( (z \times x) \leq (2 \times x + z) \) (L4G)
  - \( (z \times x) \leq (2 \times x + z) \) (L4G)

- **Coverage of VCs (%):**
  - **Z3** Yices Cvc3 Simplify Simplifier
    - Autopilot 94.2 95.5 96.2 98.0
    - Simulator 94.5 93.9 94.5 92.6 95.3
    - Tokeneer 94.6 94.8 94.1

Support is desired for features such as integer division and modulus, non linear multiplication, transcendental functions, axiomatically characterised functions (e.g. square root). While problems can be in general undecidable, often actual problems have structure that heuristics can exploit. Currently, Passmore has a system RAHD (Real Algebra in Higher Dimensions) which heuristically combines Gröbner basis calculations, semi-definite programming and real quantifier elimination (QEP-CAD). He is developing RAHD while visiting SRI. RAHD is already integrated into an internal version of SRI’s theorem prover PVS and there are plans to integrate it into the Yices SMT solver.

Possible Directions

- Invariant generation (following Ellis and Ireland?)
- Counter-example presentation
- Proof explanation
- Tool integration architectures

Research Priorities

- Developing a HOL-Light based VC prover, integrating SMT solvers and other automated provers
- Improving support for non-linear arithmetic