A Formal Approach for a Subset of the SPARK Programming Language

Semantics and Program Verification

Eduardo Brito

- Introduction
- SPARK
- Program Verification in SPARK (Comparative Case Study)
- mSPARK: A Subset of SPARK
- Syntax and Semantics of mSPARK
- mSPARK's Program Logic
- Conclusion
- Future Work

- The thesis addresses formal software verification of safety-critical systems;
- Ada is one of the most used programming languages in the safety-critical domain;
- SPARK, a strict subset of Ada, has been "pushing" the use of (semi-)formal methods in the industry and academia for the safety-critical domain;
- Even so, much work remains to be done in formal software verification, both theoretical and practical.

A safety-critical system using software developed in SPARK

SPARK is a programming language and a toolset

- It is a strict subset of Ada;
- The toolset checks for conformance with SPARK rules and allows to do formal verification of programs;
- The toolset does not include a compiler; SPARK code is meant to be compiled using (certified) Ada compilers.

Program Verification in SPARK

- We compared, using a simple case study, the program verification features of SPARK and Frama-C/ACSL;
- This case study was published on Ada Europe 2010 proceedings (Springer LNCS Vol. 6106);
- The Hi-Lite project is actually addressing the issues we expressed in the paper.
  - (and the paper was written before the project was public).

mSPARK – A Subset of SPARK

(mSPARK stands for mini/minhoSPARK).

- The creation of the subset followed some empirical decisions;
  - What features are most interesting to work on, what are more useful...
- The development was backed up by theoretical and practical work;
  - Our development cycle incorporated theoretical work and verification as well as validation through tool development and implementation;
- We believe this to be an important contribution of the thesis, as it provides a workbench for further scientific development.

Syntax

- mSPARK syntax is very close to SPARK (and Ada) syntax;
- A mSPARK program contains type declarations, global variable declarations, function and procedure declarations and (optionally) an executable part.

Semantics

- We provided an informal static semantics that allows us to establish arguments on what is valid mSPARK code;
- A formal operational semantics was developed to establish the rules of program execution;
- Formal semantics is necessary for arguing about the soundness of program logics.

mSPARK's Program Logic

- We identified some of the most important constructs of our subset and developed the program logic for it;
  - We deal with specific features of Ada such as range types and expression evaluation errors;
- Our program logic deals with the *adaptation problem* for subprograms with contracts;
- The soundness theorem that is usual in Hoare Logic had to be adapted/changed;
  - This stems mostly from the need to deal with range types and the need of providing *safety* conditions;

- This thesis resulted in several contributions:
  - A comparative case study was published at Ada Europe 2010;
    - Two other publications and a communication (poster).
  - The creation of a program logic for a programming language with safety restrictions and range types;
  - Development of Operational Semantics for the programming language;
  - Proof of soundness of our program logic;
- Tools were prototyped for the validation and application of our approach;
- Further work will be developed using these results.

- Extending the mSPARK language;
  - Packages, Object Orientation, Generics, **Concurrency, Dynamic Memory**...
    - This will be used to develop work on deductive verification on these areas;
  - Extending the Behavioral Interface Specification Language;
- Completing the proofs;
  - The soundness is not proved for subprograms;
- Further develop the tools that were prototyped.

Images used (Creative Commons - Attribution), in the respective order, from: Matthew Simantov @ flickr, Ronnie Macdonald @ flickr