Levels of Software Assurance in SPARK

Yannick Moy - AdaCore

SPARK is a formal development and verification platform that has been used in a number of industrial project domains which range across air traffic management, airborne systems, avionics, railway control & protection, security and defence systems. In companies using SPARK like Altran UK, specifications are routinely expressed with the precision of a formal language, and confidence is obtained by a combination of classical techniques plus the guarantees provided by the use of formal verification. While the benefits obtained by formal verification are clearly desirable, it may be intimidating for companies without formal verification know-how to start on this path.

To help with that process, AdaCore has collaborated with Thales throughout the year 2016 to produce a 70-pages detailed guidance document for the adoption of SPARK [1]. These guidelines are based on five levels of assurance that can be achieved on software, in increasing order of costs and benefits: Stone level (valid SPARK), Bronze level (initialization and correct data flow), Silver level (absence of run-time errors), Gold level (proof of key properties) and Platinum level (full functional correctness). The lowest four levels (except Platinum) are the relevant ones during adoption of formal verification, with the highest levels typically applied on a smaller subset of the code. All four levels were achieved in the SPARK adoption experiments carried at Thales in 2016, which we will present. The highest four levels (except Stone) map well with the verification objectives targeted at different DAL/SIL levels at Altran UK, with highest levels typically applied only at highest DAL/SIL levels (DAL A/SIL 4). This presentation of levels for both adoption and established use of formal verification is based on a currently submitted article written with Altran UK and Thales [2].

The availability of specific features in SPARK platform is key to enable this kind of gradual applicability of formal verification. Automatic generation of contracts is the main enabler for the ease of use at levels Stone and Bronze. Automatic proof is similarly important for a cost effective use at level Silver, based on the combined use of static analysis and deductive proof, and a combination of provers with different strategies (e.g. native support of bitvectors or floating-points in the prover vs. axiomatization [4]). We will show in particular how this combination is needed to prove the safe envelope on a trajectory computation. Interaction and automation are equally important at higher levels (Gold and Platinum) to achieve proof of complex properties. We will show in particular how the use of ghost code for auto-active verification allows to functionally prove an implementation of red-black trees [3].

References

[1] AdaCore and Thales. Implementation guidance for the adoption of SPARK. 2017. `http://www.adacore.com/knowledge/technical-papers/implementation-guidance-spark/`.

[2] Claire Dross, Guillaume Foliard, Lionel Matias, Stuart Matthews, Jean-Marc Mota, Yannick Moy, and Romain Soulat. Practical formal verification for reliable software. *Submitted for publication in the special issue of IEEE Software on Reliability Engineering for Software*, 2017.

[3] Claire Dross and Yannick Moy. Auto-active proof of red-black trees in SPARK. *Accepted at the 9th NASA Formal Methods Symposium, NFM 2017.*

[4] Clément Fumex, Claire Dross, Jens Gerlach, and Claude Marché. Specification and proof of high-level functional properties of bit-level programs. In *Proceedings of the 8th NASA Formal Methods Symposium, NFM 2016*.