Formally prove control flow graph algorithms in SPARK

AdaCore
46 rue d'Amsterdam
75009 Paris, France

dross@adacore.com

Brief company description

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a state-of-the-art programming language designed for large, long-lived applications where safety, security, and reliability are critical. AdaCore's flagship product is the GNAT Pro development environment, which comes all open-source with expert on-line support and is available on more platforms than any other Ada technology.

Our main offices are located in Paris, New-York and Lexington.

Internships coordination: Olivier Hainque

This internship

SPARK (http://www.spark-2014.org) is a subset of the Ada programming language for which AdaCore develops formal verification tools, allowing to prove statically that no run-time errors can occur (buffer overflow, division by zero, etc.).

The main tool for the verification of SPARK code is called GNATprove. It performs two kinds of analysis:

- data flow analysis, based on the construction of a control flow graph, to track the flow of values between variables in the program
- formal proof, also known as deductive verification, to statically check that no runtime error can occur in the program.

Goals

The aim of this internship is to formally verify some of the graph algorithms used in GNATproves data flow analysis using GNATprove. For this work, the algorithms should be rewritten using a new SPARK compatible container library which is currently under implementation at AdaCore. It can be found on github (https://github.com/AdaCore/ada-traits-containers) and has been described in a blog post (http://blog.adacore.com/traits-based-containers).

The graph algorithms used in data flow analysis range from rather simple algorithms such as depth first search to more complex ones such as transitive closure and denominator

trees. Here are some articles dealing with interactive or semi-automatic proofs of such algorithms:

- Chen, Ran, and Jean-Jacques Levy. Readable semi-automatic formal proofs of Depth-First Search in graphs using Why3. (2015).
- Blazy, Sandrine, Delphine Demange, and David Pichardie. Validating dominator trees for a fast, verified dominance test. International Conference on Interactive Theorem Proving. Springer International Publishing, 2015.
- Thry, Laurent. Formally-Proven Kosarajus algorithm. (2015).

Skills required

- Knowledge in static analysis, deductive proof if possible
- Knowledge of Ada or C is a plus

Location & Timeframe

Paris, during 2017 - 4 to 6 months

Contact

Claire Dross