Software Component Technology Group

Master's Project

Proving well-foundedness of interface specifications

Geraldine von Roten

April 2007

Introduction Static program verifiers are used to show that a given program is correct relative to its specification. In order to prove interesting properties of a program, complicated specifications need to be written. The use of calls to so-called pure methods (methods without side-effects) can help writing more abstract, readable, compact, and maintainable specifications. However, the use of method calls in specifications can lead to unsoundness. One source of unsoundness is the use of (direct or indirect) recursive specifications that lead to non-well-foundedness. Thus, such specifications should only be used for verification purposes if they are well-founded.

Goal The goal of this project is to work out a technique to show that the specifications of a given program are well-founded. That is, we are interested in finding out what the proof-obligations are that need to be generated to prove well-foundedness.
The project will be realized in the Spec# programming system.

Main parts of the project are:

1. Work out representative examples that demonstrate well-founded as well as non-well-founded recursive specifications.
2. Work out the precise proof-obligations that need to be proven to show that specifications are well-founded when they contain direct recursive calls.
3. Implement the proof-obligation generation mechanism in the Spec# system.
4. Work out and implement in the Spec# system a mechanism that forbids the occurrence of indirect cycles in specifications. Preferably the solution should be modular.

Possible Extensions The main focus of the project is on showing well-foundedness of direct recursion. This could be extended to indirect recursion too, instead of rejecting specifications that contain indirect recursion.

The project is supervised by Prof. Peter Müller, Ádám Darvas, and Arsenii Rudich.