Programming Model-Checkers using a Fixed-Point Calculus

Salvatore La Torre
University of Salerno, Italy

Abstract

Implementing symbolic model-checking algorithms is, in general, a complex task. Often traditional programming languages like C or Java, with a BDD-library to perform the symbolic operations, are used for these implementations. This forces the programmer to deal with programming aspects which are not related to the specific proposed algorithm and often concern low-level programming, such as managing memory and caching, decisions on ordering of variables, deciding order of evaluations, and many others. Moreover, small changes to the algorithm may require considerable changes in the design of the program, discouraging the testing of new ideas, and severely affecting the productivity of the programmer.

Starting from the observation that most symbolic model-checking algorithms are essentially fixed-point computations and thus can be compactly described in a high-level fixed-point calculus, we propose a fixed-point calculus as a high-level programming language to easily, correctly, and succinctly describe model-checking algorithms. Once the algorithm is expressed in such a formalism, a model-checker can be obtained simply by utilizing a symbolic fixed-point solver. The fixed-point calculus we propose allows us to efficiently implement symbolic model-checkers without resorting to low-level programming, though allowing algorithmic aspects to be specified.

These ideas have been implemented in the tool Getafix ("get a fix using fixed-points"). Getafix takes as an input sequential, concurrent and parameterized Boolean programs, and model-checks them against reachability specifications. The resulting model-checking tools are surprisingly efficient and are competitive in performance with mature existing tools that have been fine-tuned for these problems.