A Framework for Assertion-based Debugging in Constraint Logic Programming

Germán Puebla, Francisco Bueno, and Manuel Hermenegildo

{german, bueno, herme}@fi.upm.es
Department of Computer Science
Technical University of Madrid (UPM)

As constraint logic programming matures and larger applications are built, an increased need arises for advanced development and debugging environments. Assertions are linguistic constructions which allow expressing properties of programs. Classical examples of assertions are type declarations. However, herein we are interested in supporting a more general setting [3, 1] in which, on one hand assertions can be of a more general nature, including properties which are statically undecided, and, on the other, only a small number of assertions may be present in the program, i.e., the assertions are optional. In particular, we do not wish to limit the programming language or the language of assertions unnecessarily in order to make the assertions statically decidable. Consequently, the proposed framework needs to deal throughout with approximations [2].

The framework we propose (see [4]) is aimed at detecting deviations of the program behavior (symptoms) w.r.t. the given assertions, either statically (at compile-time) or dynamically (at run-time). Our approach is strongly motivated by the availability of analyzers for constraint logic programs which can statically infer a wide range of properties, from types to determinacy or termination.

We provide techniques for using information from global analysis both to detect at compile-time assertions which do not hold (i.e., static symptoms) and assertions which hold for all possible executions (i.e., statically proved assertions). We also provide program transformations which introduce tests in the program for checking assertions at run-time. Both the static and the dynamic checking are provably safe in the sense that all errors flagged are definite violations of the specifications. A preliminary implementation and evaluation of the framework has been performed. Details can be found in [4].

References