A Regular Type Analysis of Logic Programs

Lunjin Lu and John G. Cleary
Department of Computer Science
The University of Waikato

We present a new type analysis for logic programs. It is *prescriptive* in that type definitions for function symbols are provided by the programmer. Previous prescriptive type analyses are not precise enough for the following reasons. Firstly, they do not allow set operators in type expressions, forcing an over approximation when the least upper bound of types is needed. Secondly, they infer deterministic regular types by disallowing non-deterministic type definitions. Thirdly, they describe type information in a set of program states (substitutions) by a single *variable typing* that is a mapping from variables to types, severing type dependency between variables.

The new type analysis is designed as a specialization of an abstract interpretation framework for logic programs. It aims to remove the above three sources of imprecision. It uses a type language that consists of unrestricted regular types and allows set union and set intersection operators. It represents type information in a set of substitutions by a set of variable typings. The core operation of the new type analysis, called abstract unification, closely simulates concrete unification. The abstract unification is reduced to computing a set of variable typings that describes the set of those substitutions that satisfy both a set of equations in solved form and an initial set of variable typings. This is done by propagating type information in the initial set of variable typings via the equations in two steps. The first step uses type information about the variables on the lefthand sides of the equations to derive more type information about the variables occurring on the righthand sides. The derived type information is then used to strengthen the initial set of variable typings. The second step uses type information about the variables occurring in the terms on the righthand sides to derive more type information about the variables on the lefthand sides. The derived type information is used to strengthen the result of the first step.

The use of regular types and sets of variable typings makes the new type analysis more precise than those found in the literature. On the other hand, the fundamental problem of checking the emptiness of types is more complex in the new analysis. Experimental results, however, show that careful use of tabling reduces the increase of execution time to an average of 15%.