# Evaluation and Parallelization of Functions in Functional + Logic Languages

(Abstract)

Werner Damm, Feixiong Liu and Thomas Peikenkamp

Functional + logic languages integrate functional and logic programming into a single framework. In such a framework, functions are defined in the same way as in a pure functional programming framework. However, function applications can be callled in different modes as a relation in a logic programming language because they can contain logical variables in their arguments. As a result, most functional + logic languages adopt a general evaluation mechanism such as Outermost Resolution and Narrowing for the execution of both deterministic and non-deterministic function applications. Implementations of such languages have shown that some of most important features such as lazy evaluation and sharing in functional programming are more tricky to handle, and parallelization of function applications is more difficult to implement. Moreover, the evaluation of deterministic function applications by the general evaluation mechanism brings inefficiency compared with that by reduction. In this presentation, we report our investigation on these issues and provide an efficient model for the evaluation and parallelization of functions in functional + logic languages.

Our main research efforts are on the two aspects. First, we investigate how to efficiently implement lazy evaluation and sharing of functions in the context of functional + logic models. We provide some basic data structures to support lazy evaluation and sharing of non-dtereministic functions. Second, we intend to use static analysis techniques for the optimization of sequential implementation and parallelization of function applications. By static analysis, we can identify deterministic functions which can then be evaluated in a more efficient way by avoiding creating choice points and reducing the number of suspension. We have observed that the conventional strictness analysis techniques used for functional programming can not be applied to the functions in functional logic languages due to the fact that functions in most functional + logic languages are not required to be totally defined. We propose some solutions on how to modify and extend the conventional strictness analysis techniques in our framewo rk.