Dependently Type Pattern Matching Summary

Gregory Malecha

January 29, 2008

The paper develops pattern matching in DML, a dependently typed language based on SML which was presented in previous work. The paper is founded in two main technical contributions.

The first contribution is an understanding of a disconnect between static and dynamic semantics of `case` expressions. While the specification states that the order in which patterns are checked is non-deterministic, during execution branches are tested sequentially. This difference is handled by ensuring that overlapping patterns do not exist which can be performed while the exhaustiveness of the patterns is checked.

The previous contribution gives way to the second contribution which is a procedure for computing a minimal extension to a pattern set so that the pattern set covers all possible values in the type. This procedure handles dependently typed data in addition to standard data types.

The technical contributions of the work are grounded in two applications. The first is the ability to automatically expand general cases to make code more compact and maintainable. The second application deals with optimizing code based on dependent types; in particular, eliminating redundant tests during pattern matching, which, experimental results show, can result in considerable increases in performance (up to 34% with the SML compiler).