Using Haskell's Type Systems to Check Relation Algebraic Programs

Bernd Braßel

Abstract

Relation algebra provides a theoretically well founded framework to state algorithms in a declarative and concise way. Among other properties the language of relations is based on a rigorous typing discipline. Current systems to compute with relations do not, however, provide the user with type inference to ease programming. In addition, the systems lack in other aspects, like the possibility to define new data types, or to use primitive types for, e.g., numbers conveniently, or to easily define new control structures.

We introduce a binding for the lazy functional programming language Haskell to the basic operations implemented in C which underly the relation algebra system RelView. The advantages of such a binding are (at least) twofold:

1) Haskell programmers are provided with the possiblity to write highly efficient relational programs in a concise way.
2) Relational programmer are supported with a means to let infer and check types for their programs. Moreover, they can take advantage of the superior programming possibilities of Haskell.

We will describe three levels of detail for typing relational programs. The most detailed level needs to make use of a number of extensions to the standard Haskell type system.