Relational Lenses: A language for defining updateable views

Aaron Bohannon Jeffrey A. Vaughan Benjamin C. Pierce

Key Idea

Every expression in the language denotes a bidirectional transformation called a lens (a view definition together with a view update policy).

The View Definition Language

- The sources and targets of lenses are database states (sets of named relations with associated schemas).
- Schemas include predicates and functional dependencies. Both play a significant role in determining view update policies.
- A small set of basic lenses is provided.
  - Each basic lens corresponds to a simple relational operation.
  - Additional parameters determine view update policies.
- A composite lens expression can be read from left to right to describe a composite view definition and from right to left to describe a composite view update policy.

A Composite Lens:
join Tracks, Albums *deleting from* Tracks;
project Tracks *on* Track, Rating, Album, Quantity *with default* {Date=Unknown};
select from Tracks *where* Quantity > 2

Static Checking

- We are interested in lenses that are well behaved:
  - View definition and update policy "fit together" in a suitable sense.
  - Any consistent updated view state is mapped to a consistent updated database state (totality).
- Well-behavedness is guaranteed by static checking.
  - Each primitive lens comes with a typing constraint guaranteeing its well-behavedness.
  - Well-behavedness of well-typed composite lenses follows by construction.

Key Research Challenge

Detailed design of the basic lenses and their typing constraints.