ALPHA 5 COMPILATION

OLIVER KENNEDY

1. Compiler Work Loop

The compiler maintains a work queue datastructure. As long as the work queue is non-empty, the compiler will perform the following steps on whatever expression is at the head of the work queue[1]:

(1) **Optimize the expression** The expression is simplified and factorized as discussed below.
(2) **Prematerialize the expression**. For every Lift term that appears in the expression, a copy of the subexpression being lifted is set aside and stored for use in the materialization stage.
(3) **Subdivide the expression** (optional) In the future, we may wish to subdivide the expression into smaller chunks, each of which (after we compute deltas) is guaranteed to be part of a separate materialized subexpression. For now, this is ignored.
(4) **Compute Deltas**. For each *stream* appearing in the expression, apply the delta operation to the expression to get a set of delta expressions.
(5) **Materialize the delta expressions**. Subexpressions are pulled out and replaced with Externals. This process is discussed in detail below.
(6) **Update the work queue**. Newly instantiated datastructures are added to the work queue. The recently compiled datastructure is also stored.

1.1. Expression Optimization.

Simplification is the application of a set of rewrite rules that provides the following guarantees:

(1) Values appearing in sum and product terms are merged together and constants are combined/curried.
(2) Equality comparisons are converted into Lifts as much as possible.
(3) Lifts are unified as much as possible.
(4) Lifts and AggSums are un-nested as far as possible: Expressions that always evaluate to 0 or 1 (comparisons, lifts, and keyed relations) can always be pulled out of a lifted product. If all of the output variables of an expression are group-by variables of an AggSum of a product that the expression appears in, the expression can be lifted out. These variables can then be removed from the AggSum's group-by variable list.

[1]This section assumes that the head is a standard map datastructure. If not, then we might do something more interesting. We'll find out once we actually start supporting non-map datastructures.

(5) AggSums are factorized: AggSum([...], R(A) * S(B)) becomes AggSum([...], R(A)) * AggSum([...], S(B))
(6) Unnecessary AggSums are eliminated if the aggregate does not project away any variables (i.e., if the group-by variables of the AggSum are identical to the aggregated expression's output variables).

2. Expression Materialization

3. Domain Maintenance