Declarative Data Processing

Distribution of the Work

SQL is declarative, but is designed for querying data. Advanced dataflows characterized by heavy use of library methods, control flow, and nesting stretch its limits.

Embedded dataflow DSLs overcome these problems, but are too low-level. Runtime aspects like caching, partitioning, and aggregation need to be hard-coded by the programmer.

The benefits of the two can we combined if we change the embedding strategy.

Comprehension Syntax

Comprehensions generalize SQL and available as first-class syntax in modern general purpose programming languages.

**Maths**

\[
\{ (x, y) | x \in xs, y \in ys, x < y \}
\]

**SQL**

SELECT x, y FROM x AS xs, y AS ys WHERE x > y

**Python**

\[
\{ (x, y) for x in xs, y in ys, x > y \}
\]

**Scala**

\[
\text{for} \ (x < xs; y < ys; y > y) \ \text{yield} \ (x, y)
\]

Comprehension Semantics

Comprehension syntax can be enabled in Scala if we extend the bag type to a monad using three second-order functions: map, flatMap, and withFilter.

**Desugared Comprehension**

\[
xs\text{.flatMap}(e \mapsto y)\text{.withFilter}(y \mapsto x \mapsto y)\text{.map}(y \mapsto (x, y))
\]

Flattened and Filtered Result

\[
\text{Flattened and Filtered Result}
\]

**Example: Transitive Closure**

```
val algorithm = emma.parallelize {
  var edges = read(input, _).distinct()
  var sizeV = 8 // old size
  var sizeN = edges.size // new size
  while (sizeN > sizeV) {
    val closure = for {
      e1 <- edges
      e2 <- edges
      if (e1.dst == e2.src)
        yield (e1.src, e2.dst)
    } yield (edges plus closure).distinct()
    sizeV = sizeN
    sizeN = edges.size
  }
  write(output, _)(edges)
}
algorithm.run(rt.engine("spark")) // or "flink"
```

Emma in Action

Reuse linguistic concepts available in Scala such as while loops, for-comprehensions, and product types.

Develop and test locally. When the code looks good, wrap it inside an emma.parallelize(...) macro.

Emma will (1) identify maximal bag terms, (2) rewrite them holistically, and (3) transparently offload them on a parallel dataflow engine at runtime.

**Basic Principles**

**Demonstration**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-H Queries</td>
<td>Relational</td>
</tr>
<tr>
<td>K-Means</td>
<td>Clustering</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Classification</td>
</tr>
<tr>
<td>Belief Propagation</td>
<td>Statistical inference</td>
</tr>
<tr>
<td>Triangle Count</td>
<td>Graph Analysis</td>
</tr>
</tbody>
</table>

Learn More

* Implicit Parallelism through Deep Language Embedding. SIGMOD Conference 2015: 47-61*

http://www.emma-language.org