Fixing flattening's space complexity flaws with flow fusion

Amos Robinson

September 2, 2013

Parallel programming is hard

- Writing parallel programs by hand is hard and error-prone
- Nested data parallelism makes it easier for the programmer
- Compiling nested data parallelism efficiently is hard

Space complexity problem with NDP

Flattening can actually ruin a program's space complexity.
Finding the furthest distance between any points.

```
maxdist :: Vector Point -> Distance
maxdist vs
 = maximum
 $ map maximum
 $ map (\v → map (distance v) vs) vs
```

Evaluating this naïvely, we end up with an n^2 array in memory:

```
maxdist [p, q, r]
==>
maximum $ map maximum $
[ [distance p p, distance p q, distance p r]
, [distance q p, distance q q, distance q r]
, [distance r p, distance r q, distance r r]]
```

Fusion fixes space complexity

- Fusion removes intermediate arrays
- The n^2 array is intermediate
- So it should be fused away

Short-cut fusion is fragile and relies on inlining

Short-cut fusion:

- Existing fusion systems are fragile
- Rely on complicated compiler optimisations
- Not obvious when fusion will apply

Flow fusion:

- Flow fusion can fix more cases of space complexity
- Unknown whether *all* cases
- Unlikely all cases

Plan

- Which cases does flow fusion fix
- Compiler warnings for other cases
- Prove it

End

end.