EIN and DATm updates

Charisee Chiw

August 20, 2017

1 Testing

Once we have added a new operator to the surface language, it is natural to write some test programs by hand. The tests we wrote by hand were straightforward, but limited and unhelpful since it missed bugs. We can apply a more rigorous approach by using the testing system, *DATm*. We add the concat operator to *DATm* (by creating a new operator object) and used targeted testing to only generate test cases that use the concat operator. *DATm* created and ran 126 test programs that use the concat operator.

1.1 *DATm* implementation

inside tests

adjustments

third-arity

1.2 Results from using *DATm*

Bug 1 Mistake in index scope when using substitution.

```
field#k(2)[2,2] F0;
field#k(2)[2] F1;
field#k(2)[2] F2;
field#k(2)[2,2] G = (F0 ∘ F1) • F2;
```

There was a mistake in the substitution method. The scope of the composition indices were handled incorrectly. The following is the expected and observed representation of the computation in the EIN IR.
Expected: $e = \sum_{\hat{j}} A_{ij} \circ [\langle B_i \rangle_{\hat{\beta}}] * C_j$
Observed: $e = \sum_{\hat{k}} A_{ik} \circ [\langle B_i \rangle_{\hat{\beta}}] * C_j$
in $\lambda(A, B, C)\langle e \rangle_{\beta}$(F0,F1,F2).

Bug 2 Missing cases in split method.
Probes of a composition are handled differently before reconstruction.
$\sum F(x)$ and $\sum(\text{Comp}(F, G, -))(x)$.
Missing case in method leads to a compile time error. Additionally (Comp(Comp -)-)

Bug 3 Differentiate a composition
The jacobian of a field composition:

```
field#k(d1)[d] F0;
field#k(d)[d1] F1;
field#k(d1)[d,d1] G = ∇⊗ (F0 ∘ F1);
```

is represented as $\langle \nabla_j(\text{Comp}(A_i, B_i, i)) \rangle_{ij}$
In accordance with the chain rule ((f ∘ g)' = (f' ∘ g) · g') the rewriting system multiplies the inner derivative (g') with a new composition operation (f' ∘ g). In practice, the implementation does a point-wise multiplication when it should do an inner product.
Expected: $\sum_{\hat{k}}(\nabla_k A_i \circ [\langle B_i \rangle_{\hat{\beta}}] * \nabla_j G_k)$
Observed: $\nabla_j A_\beta \circ [\langle B_i \rangle_{\hat{i}}] * (\nabla_j G_i)$
in $\lambda(A, B)\langle e \rangle_{ij}$(F0,F1).

False positive There was an error in evaluating the correct answer in *DATm* for multiple applications of the compose operator. We replace the position variable in one field with the field expression of the other. The position variables need to be renamed in order to avoid mixing them. Without renaming them it resulted in a false positive.

False positive *DATm* generates inside tests to see if we are probing the field in the right position. They did not account for all the different types of restrictions.

Bug 4 One was in the creation of the EIN operator for the concat operator..

Bug 5 The bug arose when computing the determinant of the concatenation of a field.

```
field#k(d)[2]F,G;
field#k(d)[]H = det(concat(F,G));
```

The bug was caused by the rewriting system. Our rewriting system applies index-based rewrites to reduce EIN expressions. A specific index rewrite is applied when the index in the delta term matches an index in tensor (or field) term. The rewrite checked if two indices were equal and did not distinguish between variable and constant indices. It is mathematically incorrect to reduce constant indices, because they are not equivalent to variable indices.

XT6 Using If Wrapper with other field operators

```
field#4(2)[]G = compose(minF((F0),(F1)),(F2*0.1));
```

Field operators need to be applied to the leaves in if wrapper. The composition is a field operator so it needed to be pushed to the leaves.

$$(\text{If}(c, e3, e4)) \circ es \rightarrow \text{If}(c, e3 \circ es, e4 \circ es)$$