Implementing Tensor Calc functions in Diderot: Concatenation

Charisee Chiw

March 28, 2018

Shorthand *Diderot_Dev* = https://github.com/cchiw/Diderot-Dev
Exs = https://github.com/cchiw/latte/
Doc = *Exs*/writeup/paper.pdf
dissertation = Chiw's dissertation

1 Overview

Functionality: **Concatenation**
Syntax: "concat()"
field#k(d)[α] $\times$ field#k(d)[α] $\rightarrow$ field#k(d)[2,α]
Branch: *Diderot_Dev*
Text: Mentioned in *dissertation*-FW. Details provided in *Doc*
Issues: None
Examples: *Exs*/concatenation
Future work: Use syntax "[","]"

2 Design and Implementation

A user can define new tensors by concatenating tensors together. A Diderot program

```
tensor[d1]S;
tensor[d2]T;
tensor[d1,d2]M = [S,T];
```

A user can refer to components of tensor fields by using the slice operation as shown in the following code. A Diderot program

```
field#k(d)[d1,d2]A;
field#k(d)[d1,d2]B;
field#k(d)[d1]F = A[:,0];
field#k(d)[d1]G = B[:,1];
```

We would like to provide a way to define new tensors fields by concatenating components together. Using the tensor field variables F and G defined earlier in the program the Diderot code should support the line

```
field#k(d)[d1,d1]H = [F, G];
```

We illustrate the structure of H below.

$$H = \begin{bmatrix} F_0 & F_1 \\ G_0 & G_1 \end{bmatrix}$$

Representation We can use EIN expressions as building blocks to represent field concatenation. In EIN each field term is represented by an expression and it is enabled with a delta function

$$\underset{init}{\Longrightarrow} H = \lambda F, G.\langle F_j\delta_{0i} + G_j\delta_{1i}\rangle_{i:2,j:2}(\mathrm{F,G})$$

After substitution the new EIN operator would be

$$\underset{subst}{\Longrightarrow} H = \lambda A, B.\langle A_{j0}\delta_{0i} + B_{j1}\delta_{1i}\rangle_{\hat{i}\hat{j}}(\mathrm{A,B}).$$

In the compiler we choose to create generic versions of an EIN operator that can be instantiated to certain types.

$$\lambda F, G.\langle F_\alpha \delta_{0i} + G_\alpha \delta_{1i} \rangle_{i:2\hat{\alpha}}(\mathrm{F,G})$$
$$\lambda F, G, H.\langle F_\alpha \delta_{0i} + G_\alpha \delta_{1i} + H_\alpha \delta_{2i} \rangle_{i:3\hat{\alpha}}(\mathrm{F,G,H})$$

To implement this operator we need to add to cases to the Diderot typechecker and add the generic representations but not much else. Since we are solely using existing EIN expressions to represent this computation, we can rely on the existing code to handle the EIN syntax.

3 Documented (solved) bugs

concat-B1 One was in the creation of the EIN operator for the concat operator. The indices on the terms in the concat EIN operator were switched.

concat-B2 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.