Multi-Level Languages are Generalized Arrows

Adam Megacz

12-Apr-2011

Lambda Calculus Review

$$\tau := \texttt{Int} \mid \tau\texttt{->}\tau \mid \ldots$$
$$\texttt{e} := \texttt{x} \mid \texttt{e e} \mid \lambda\texttt{x.e}$$

$$\text{[Var]}\ \frac{}{\texttt{x}:\tau \vdash \texttt{x}:\tau}$$

$$\text{[Lam]}\ \frac{\Gamma, \texttt{x}:\tau_1 \vdash \texttt{e}:\tau_2}{\Gamma \vdash (\lambda\texttt{x.e}):\tau_1\texttt{->}\tau_2}$$

$$\text{[App]}\ \frac{\Gamma_1 \vdash \texttt{e}_1:\tau_1 \qquad \Gamma_2 \vdash \texttt{e}_2:\tau_1\texttt{->}\tau_2}{\Gamma_1, \Gamma_2 \vdash \texttt{e}_2\texttt{e}_1:\tau_2}$$

Flattening, in Detail

What follows is the *simplified translation*, which works only when higher-order functions are *not* used. Specifically, the context (area to the left of the turnstile) may not contain any variables with ->'s in their types.

Flattening, in Detail

Let's try an example. We're going to flatten this expression:

```
subtract_second_from_third :: Int -> (Int -> (Int -> Int))
subtract_second_from_third x y z = z - y
```

But first, we need to *desugar* it. The expression above is really just syntactic sugar for this:

```
subtract_second_from_third = \x -> \y -> \z -> (-) z y
```

Also, note that this expression has `(-)` as a free variable, with the following type:

```
(-) :: Int -> Int -> Int
```

We'll come back to this.

Flattening, in Detail

Let's try an example. We're going to flatten this expression:

```
subtract_second_from_third :: Int -> (Int -> (Int -> Int))
subtract_second_from_third x y z = z - y
```

But first, we need to *desugar* it. The expression above is really just syntactic sugar for this:

```
subtract_second_from_third = \x -> \y -> \z -> (-) z y
```

Also, note that this expression has `(-)` as a free variable, with the following type:

```
(-) :: Int -> Int -> Int
```

We'll come back to this.

Flattening, in Detail

Let's try an example. We're going to flatten this expression:

```
subtract_second_from_third :: Int -> (Int -> (Int -> Int))
subtract_second_from_third x y z = z - y
```

But first, we need to *desugar* it. The expression above is really just syntactic sugar for this:

```
subtract_second_from_third = \x -> \y -> \z -> (-) z y
```

Also, note that this expression has `(-)` as a free variable, with the following type:

```
(-) :: Int -> Int -> Int
```

We'll come back to this.

Flattening, in Detail

The (simplified) flattening process involves six steps:

1. Construct the proof that the expression is well-typed
2. Skolemize the proof
3. Make contraction, and exchange explicit in the proof
4. Make left and right expansion explicit in the proof
5. Make weakening explicit in the proof
6. Walk the proof

In practice, steps 2-5 are performed simultaneously as one big step. In these slides they are presented as separate steps in an order which keeps each of the proofs small enough to fit on a slide.

Flattening, in Detail

The (simplified) flattening process involves six steps:

1. Construct the proof that the expression is well-typed
2. Skolemize the proof
3. Make contraction, and exchange explicit in the proof
4. Make left and right expansion explicit in the proof
5. Make weakening explicit in the proof
6. Walk the proof

In practice, steps 2-5 are performed simultaneously as one big step. In these slides they are presented as separate steps in an order which keeps each of the proofs small enough to fit on a slide.

Step 1: Construct the Proof

```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```

$$\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-) : Int->Int->Int}}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]$$

$$\dfrac{\dfrac{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]}{}\,[\text{Lam}]$$

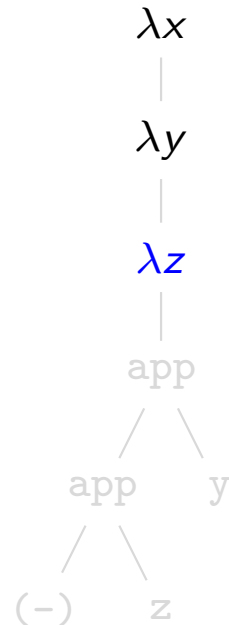
Step 1: Construct the Proof

```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-) : Int->Int->Int}}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

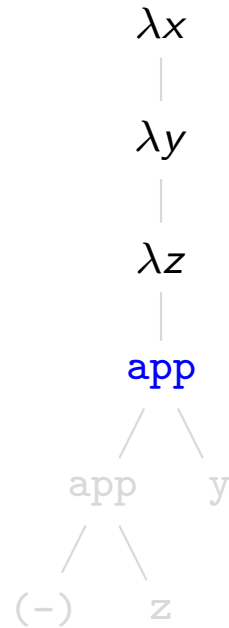
```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

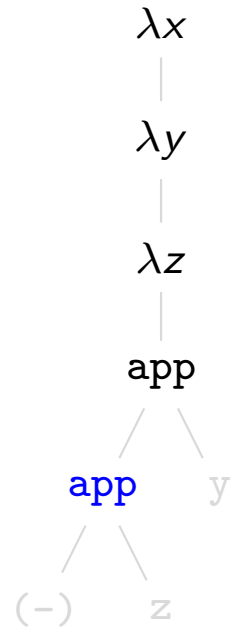
```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-) : Int->Int->Int}}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

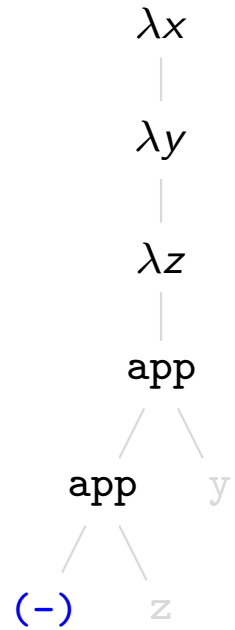
```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z} : \texttt{Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

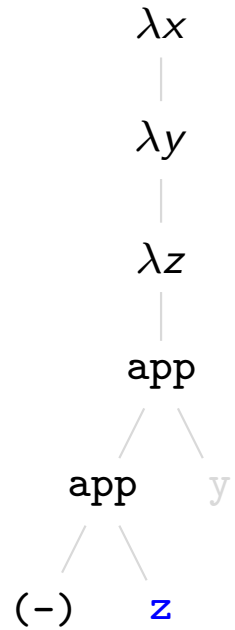
```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

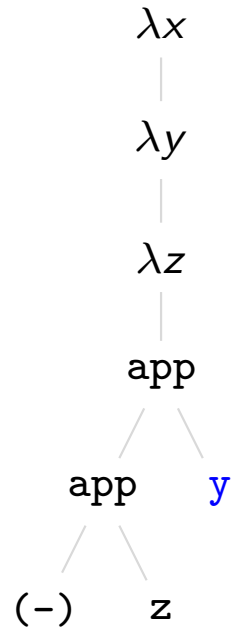
```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-)} : \texttt{Int->Int->Int}}{\texttt{z} : \texttt{Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

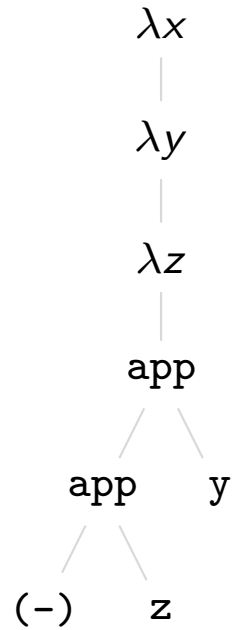
```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 1: Construct the Proof

```
subtract_first_from_third =
    \x ->
      \y ->
        \z ->
           ((-) z) y
```



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-) : Int->Int->Int}}{\texttt{z : Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 2: Skolemize the Proof

Now, we *skolemize* the proof to eliminate function types `a->b`. Wherever an expression of function type appears to the right of the turnstile, like this:

$$\Gamma \vdash \texttt{e} : \alpha \texttt{->} \beta$$

Step 2: Skolemize the Proof

Now, we *skolemize* the proof to eliminate function types `a->b`. Wherever an expression of function type appears to the right of the turnstile, like this:

$$\Gamma \vdash \texttt{e} : \alpha \texttt{->} \beta$$

we will add a new *skolem variable* with a fresh name to the left of the turnstile and apply the expression to it, like this:

$$\texttt{a} : \alpha, \Gamma \vdash \texttt{e a} : \beta$$

Step 2: Skolemize the Proof

Now, we *skolemize* the proof to eliminate function types `a->b`. Wherever an expression of function type appears to the right of the turnstile, like this:

$$\Gamma \vdash \texttt{e} : \alpha \texttt{->} \beta$$

we will add a new *skolem variable* with a fresh name to the left of the turnstile and apply the expression to it, like this:

$$\texttt{a} : \alpha, \Gamma \vdash \texttt{e a} : \beta$$

Repeated application of the procedure above will leave us with a proof which has no `->`'s to the right of the turnstile.

Skolemizing Lam

$$\frac{\mathsf{y}:\alpha, \Gamma \vdash \mathsf{e}:\beta}{\Gamma \vdash (\lambda \mathsf{y}.\mathsf{e}):\alpha \texttt{->} \beta}\ [\mathsf{Lam}]$$

Skolemizing Lam

$$\frac{\mathtt{y}:\alpha, \Gamma \vdash \mathtt{e}:\beta}{\Gamma \vdash (\lambda \mathtt{y}.\mathtt{e}):\alpha \texttt{->} \beta} \text{ [Lam]}$$

Notice what happens to the Lam rule when we skolemize it:

$$\frac{\mathtt{y}:\alpha, \Gamma \vdash \mathtt{e}:\beta}{\mathtt{a}:\alpha, \Gamma \vdash (\lambda \mathtt{y}.\mathtt{e})\ \mathtt{a}:\beta} \text{ [Skolemized-Lam]}$$

Skolemizing Lam

$$\frac{\mathsf{y}:\alpha, \Gamma \vdash \mathsf{e}:\beta}{\Gamma \vdash (\lambda \mathsf{y}.\mathsf{e}):\alpha\texttt{->}\beta}\ [\text{Lam}]$$

Notice what happens to the Lam rule when we skolemize it:

$$\frac{\mathsf{y}:\alpha, \Gamma \vdash \mathsf{e}:\beta}{\mathtt{a}:\alpha, \Gamma \vdash (\lambda \mathsf{y}.\mathsf{e})\ \mathtt{a}:\beta}\ [\text{Skolemized-Lam}]$$

Because $(\lambda \mathsf{y}.\mathsf{e})$ a reduces via β-reduction to $\mathsf{e}[\mathsf{y} := \mathsf{a}]$, Skolemized-Lam is nothing more than variable renaming. We can eliminate it entirely by changing the names of the variables in the proof tree above it:

$$\frac{\dfrac{\vdots}{\mathsf{y}:\alpha, \Gamma \vdash \mathsf{e}:\beta}}{\Gamma \vdash (\lambda \mathsf{y}.\mathsf{e}):\alpha\texttt{->}\beta} \Rightarrow \frac{\dfrac{\vdots}{\mathsf{y}:\alpha, \Gamma \vdash \mathsf{e}:\beta}}{\mathtt{a}:\alpha, \Gamma \vdash (\lambda \mathsf{y}.\mathsf{e})\ \mathtt{a}:\beta} \Rightarrow \frac{\vdots\ [\mathsf{y}:=\mathsf{a}]}{\mathtt{a}:\alpha, \Gamma \vdash \mathsf{e}[\mathsf{y}:=\mathsf{a}]:\beta}$$

Skolemizing App

Something interesting happens to the App rule, as well:

$$\frac{\Gamma_1 \vdash \texttt{e}:\alpha \qquad \Gamma_2 \vdash \texttt{f}:\alpha\texttt{->}\beta}{\Gamma_1, \Gamma_2 \vdash \texttt{f e}:\beta}\ [\text{App}]$$

$$\frac{\Gamma_1 \vdash \texttt{e}:\alpha \qquad \texttt{a}:\alpha, \Gamma_2 \vdash \texttt{f a}:\beta}{\Gamma_1, \Gamma_2 \vdash \texttt{f e}:\beta}\ [\text{App}]$$

Look at what happens if we erase the variable names and expressions:

$$\frac{\Gamma_1 \vdash \alpha \qquad \alpha, \Gamma_2 \vdash \beta}{\Gamma_1, \Gamma_2 \vdash \beta}\ [\text{App}]$$

This is none other than the famous *cut rule* from logic!

Main Idea: if an expression does not use higher-order functions, we can skolemize its proof to obtain one with no Lam rules or ->'s, and in which the App rules are natural-deduction cuts.

Step 2: Skolemize the Proof

Here is our proof again, showing in red the parts which will be removed by skolemization:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-)} : \texttt{Int->Int->Int}}{\texttt{z} : \texttt{Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

Step 2: Skolemize the Proof

Here is our proof again, showing in red the parts which will be removed by skolemization:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \vdash \texttt{(-)} : \texttt{Int->Int->Int}}{\texttt{z} : \texttt{Int} \vdash \texttt{(-) z:Int->Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int} \vdash \texttt{\textbackslash z -> (-) z y:Int->Int}}\,[\text{Lam}]}{\texttt{x:Int} \vdash \texttt{\textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->Int)}}\,[\text{Lam}]}{\vdash \texttt{\textbackslash x -> \textbackslash y -> \textbackslash z -> (-) z y:Int->(Int->(Int->Int))}}\,[\text{Lam}]$$

And here what is left after skolemizing it; the skolem variables are shown in red.

$$\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}] \qquad \texttt{a} : \texttt{Int}, \texttt{b:Int} \vdash \texttt{(-) a b} : \texttt{Int}}{\texttt{b:Int}, \texttt{z} : \texttt{Int} \vdash \texttt{(-) z b:Int}}\,[\text{App}]}{\texttt{x:Int, y:Int, z:Int} \vdash \texttt{(-) z y:Int}}\,[\text{App}]$$

Step 3: Make Contraction and Exchange Explicit

You'll notice that I played fast and loose with the order of the variables in the context; we must now use explicit rules to re-arrange the context. Here is the rule for exchange:

$$\frac{A, B \vdash C}{B, A \vdash C}\ [\text{Exch}]$$

And here is our proof with exchange made explicit:

$$\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\ [\text{Var}] \qquad \dfrac{\dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\ [\text{Var}] \qquad \texttt{a : Int}, \texttt{b:Int} \vdash \texttt{(-) a b : Int}}{\texttt{z:Int}, \texttt{b : Int} \vdash \texttt{(-) z b:Int}}\ [\text{App}]}{\texttt{b : Int}, \texttt{z:Int} \vdash \texttt{(-) z b : Int}}\ [\text{Exch}]}{\texttt{x:Int}, \texttt{y:Int}, \texttt{z:Int} \vdash \texttt{(-) z y:Int}}\ [\text{App}]$$

Step 4: Make Left and Right Expansion Explicit

Our [App] rule looks like this:

$$\frac{A \vdash B \qquad X, B \vdash C}{X, A \vdash C}\ [\text{App}]$$

Notice that in the hypotheses, the "inner pair" don't match. We would prefer to use [App'], which requires that they *do* match:

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\ [\text{App'}]$$

$$\frac{A \vdash B}{X, A \vdash X, B}\ [\text{Left}] \qquad \frac{A \vdash B}{X, A \vdash B, X}\ [\text{Right}]$$

We can rewrite any use of [App] using [App'] and [Left] or [Right]:

$$\frac{\dfrac{A \vdash B}{X, A \vdash X, B}\ [\text{Left}] \qquad X, B \vdash C}{X, A \vdash C}\ [\text{App'}]$$

Step 4: Make Left and Right Expansion Explicit

Here is what our proof looks like with [App'] instead of [App] and explicit [Left] and [Right]:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z:Int}, \texttt{b:Int} \vdash \texttt{z:Int}, \texttt{b:Int}}\,[\text{Right}] \qquad \texttt{a} : \texttt{Int}, \texttt{b:Int} \vdash \texttt{(-) a b} : \texttt{Int}}{\texttt{z:Int}, \texttt{b} : \texttt{Int} \vdash \texttt{(-) z b:Int}}\,[\text{App'}]}{\texttt{b} : \texttt{Int}, \texttt{z:Int} \vdash \texttt{(-) z b} : \texttt{Int}}\,[\text{Exch}]$$

$$\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}]}{\texttt{y:Int}, \texttt{z:Int} \vdash \texttt{y:Int}, \texttt{z:Int}}\,[\text{Right}] \qquad \vdots}{\texttt{x:Int}, \texttt{y:Int}, \texttt{z} : \texttt{Int} \vdash \texttt{(-) z y:Int}}\,[\text{App'}]$$

Step 5: Make Weakening Structurally Explicit

Finally, you'll notice that although we never used the variable `y`, I never explicitly threw it away. We will now do that using the [Weak] rule:

$$\frac{}{\texttt{q}:\alpha \vdash \langle\rangle}\ [\text{Weak}]$$

Here is the result:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\ [\text{Var}]}{\texttt{b:Int, z:Int} \vdash \texttt{b:Int, z:Int}}\ [\text{Left}] \qquad \texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int}}{\texttt{z:Int, b : Int} \vdash \texttt{(-) z b:Int}}\ [\text{App'}]}{\texttt{b : Int, z:Int} \vdash \texttt{(-) z b : Int}}\ [\text{Exch}]$$

$$\dfrac{\dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\ [\text{Var}]}{\texttt{z:Int, y:Int} \vdash \texttt{z:Int, y:Int}}\ [\text{Left}] \qquad \vdots}{\texttt{y:Int, z : Int} \vdash \texttt{(-) z y:Int}}\ [\text{App'}]$$

$$\dfrac{\dfrac{\dfrac{}{\texttt{x:Int} \vdash \langle\rangle}\ [\text{Weak}]}{\texttt{x:Int}, \langle \texttt{y:Int, z:Int} \rangle \vdash \langle\rangle, \langle \texttt{y:Int, z:Int} \rangle}\ [\text{Right}] \qquad \dfrac{\vdots}{\texttt{y:Int, z : Int} \vdash \texttt{(-) z y:Int}}}{\langle\rangle, \langle \texttt{y:Int, z:Int} \rangle \vdash \texttt{(-) z y:Int}}\ [\text{LeftUnit}]}{\texttt{x:Int}, \langle \texttt{y:Int, z:Int} \rangle \vdash \texttt{(-) z y:Int}}\ [\text{App'}]$$

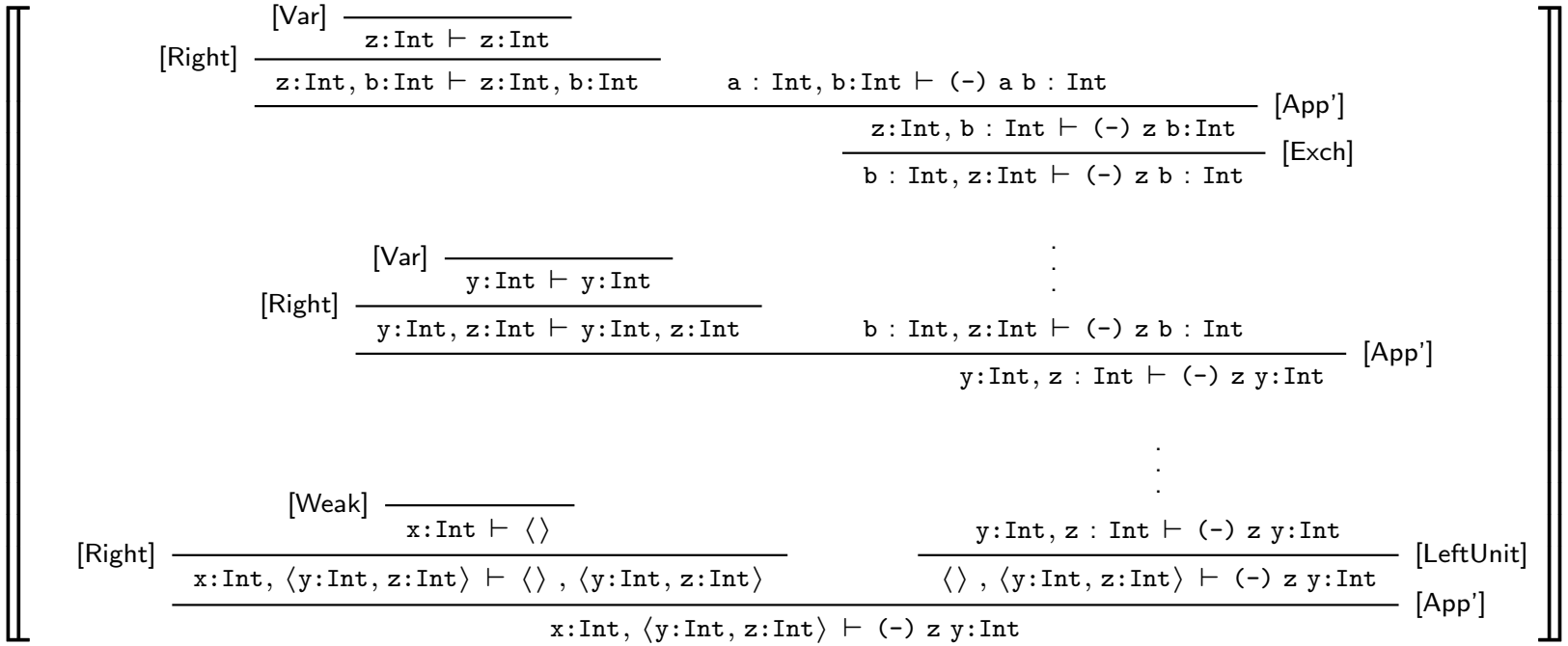
Step 6: Walk the Proof

Almost there! We now "walk" the proof from the bottom to the top, doing a purely local translation. Here is the rule for the translation function $[\![-]\!]$ on proofs, whose argument is a *proof* and whose result is a *Haskell expression*.

$$\left[\!\!\left[\dfrac{\begin{array}{ccc}\vdots & & \vdots \\ H_1 & \cdots & H_n\end{array}}{C}\,[R] \right]\!\!\right] = [\![R]\!] \left[\!\!\left[\dfrac{\vdots}{H_1} \right]\!\!\right] \cdots \left[\!\!\left[\dfrac{\vdots}{H_n} \right]\!\!\right]$$

$[\![\text{Var}]\!]$	= `id`	$[\![\text{Right}]\!]$	= `ga_first`
$[\![\text{App'}]\!]$	= `>>>`	$[\![\text{Left}]\!]$	= `ga_second`
$[\![\text{Lam}]\!]$	= *skolemized away*	$[\![\text{LeftUnit}]\!]$	= `ga_cancell >>>`
$[\![\text{App}]\!]$	= *skolemized away*	$[\![\text{RightUnit}]\!]$	= `ga_cancelr >>>`
		$[\![\text{LeftCancel}]\!]$	= `ga_uncancell >>>`
$[\![\text{Weak}]\!]$	= `ga_drop >>>`	$[\![\text{RightCancel}]\!]$	= `ga_uncancelr >>>`
$[\![\text{Cont}]\!]$	= `ga_copy >>>`	$[\![\text{Assoc}]\!]$	= `ga_assoc >>>`
$[\![\text{Exch}]\!]$	= `ga_swap >>>`	$[\![\text{UnAssoc}]\!]$	= `ga_unassoc >>>`

Step 6: Walk the Proof



Step 6: Walk the Proof

$$[\![\ \text{[Right]}\ \frac{\text{[Weak]}\ \dfrac{}{\texttt{x:Int} \vdash \langle\rangle}}{\texttt{x:Int}, \langle \texttt{y:Int}, \texttt{z:Int} \rangle \vdash \langle\rangle\ ,\ \langle \texttt{y:Int}, \texttt{z:Int} \rangle}\ ]\!] \quad >>>$$

$$[\![\ \text{[Right]}\ \frac{\text{[Var]}\ \dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}}{\texttt{z:Int}, \texttt{b:Int} \vdash \texttt{z:Int}, \texttt{b:Int}} \qquad \texttt{a : Int}, \texttt{b:Int} \vdash \texttt{(-) a b : Int}}{\dfrac{\dfrac{}{\texttt{z:Int}, \texttt{b : Int} \vdash \texttt{(-) z b:Int}}\ \text{[Exch]}}{\texttt{b : Int}, \texttt{z:Int} \vdash \texttt{(-) z b : Int}}\ \text{[App']}}$$

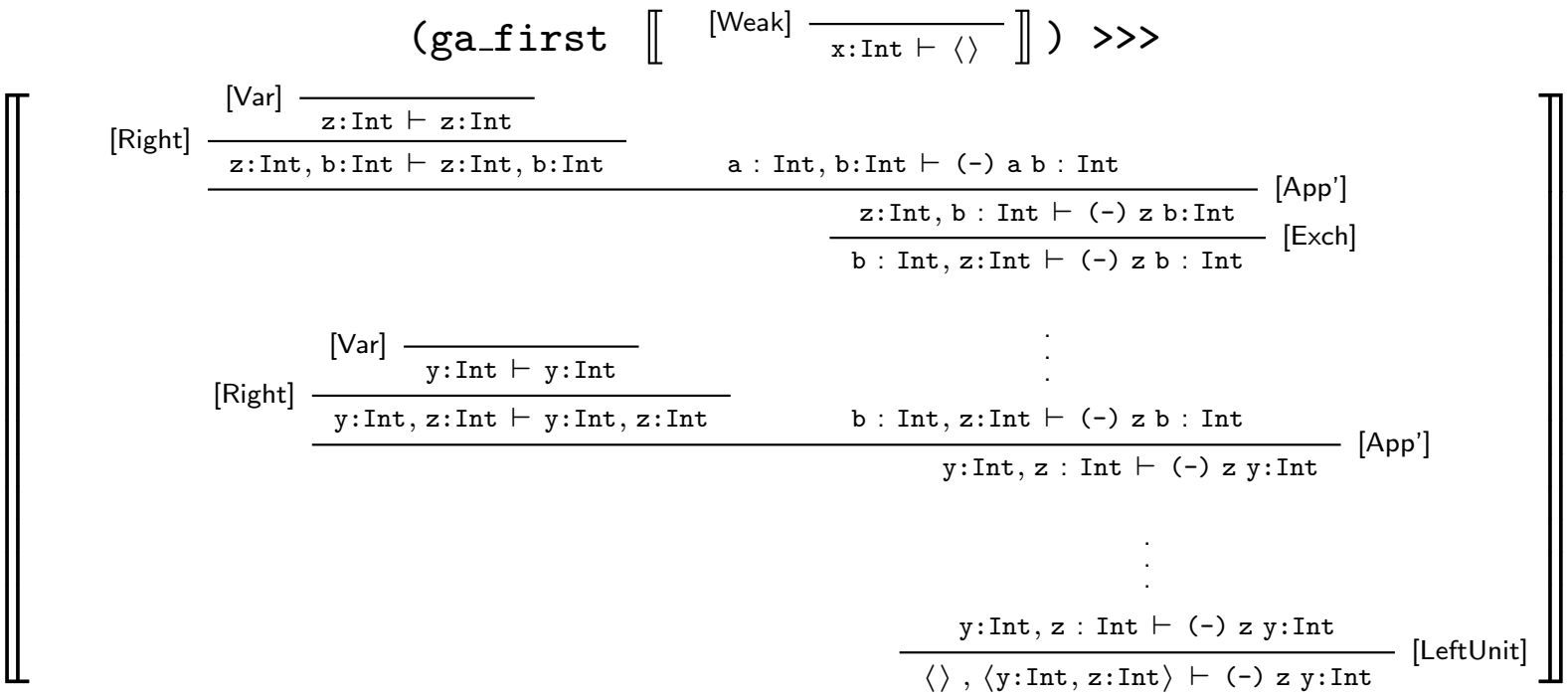
$$\text{[Right]}\ \frac{\text{[Var]}\ \dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}}{\texttt{y:Int}, \texttt{z:Int} \vdash \texttt{y:Int}, \texttt{z:Int}} \qquad \begin{array}{c}\vdots\\ \texttt{b : Int}, \texttt{z:Int} \vdash \texttt{(-) z b : Int}\end{array}$$

$$\frac{}{\texttt{y:Int}, \texttt{z : Int} \vdash \texttt{(-) z y:Int}}\ \text{[App']}$$

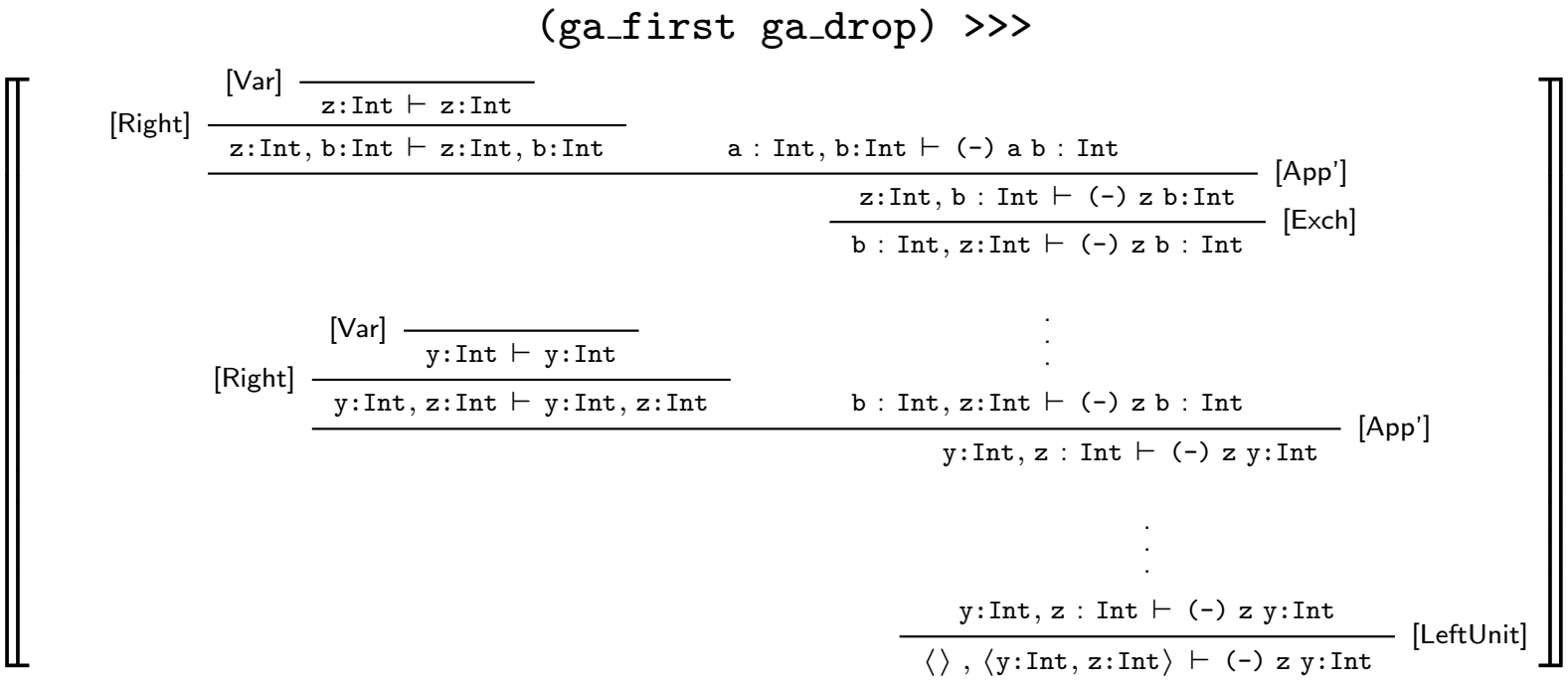
$$\begin{array}{c}\vdots\\ \texttt{y:Int}, \texttt{z : Int} \vdash \texttt{(-) z y:Int}\end{array}$$

$$\frac{}{\langle\rangle\ ,\ \langle \texttt{y:Int}, \texttt{z:Int} \rangle \vdash \texttt{(-) z y:Int}}\ \text{[LeftUnit]}\ ]\!]$$

Step 6: Walk the Proof



Step 6: Walk the Proof



Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
(ga_cancell >>>
```

$$
\left[\!\!\left[
\dfrac{
\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]
}{
\texttt{z:Int}, \texttt{b:Int} \vdash \texttt{z:Int}, \texttt{b:Int}
}\,[\text{Right}]
\qquad
\texttt{a} : \texttt{Int}, \texttt{b:Int} \vdash \texttt{(-) a b} : \texttt{Int}
}{
\dfrac{
\texttt{z:Int}, \texttt{b} : \texttt{Int} \vdash \texttt{(-) z b:Int}
}{
\texttt{b} : \texttt{Int}, \texttt{z:Int} \vdash \texttt{(-) z b} : \texttt{Int}
}\,[\text{Exch}]
}\,[\text{App'}]
\right.
\right.
\qquad
\left.\!\!\left]
\dfrac{
\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}]
}{
\texttt{y:Int}, \texttt{z:Int} \vdash \texttt{y:Int}, \texttt{z:Int}
}\,[\text{Right}]
\qquad
\begin{array}{c}\vdots\\ \texttt{b} : \texttt{Int}, \texttt{z:Int} \vdash \texttt{(-) z b} : \texttt{Int}\end{array}
}{
\texttt{y:Int}, \texttt{z} : \texttt{Int} \vdash \texttt{(-) z y:Int}
}\,[\text{App'}]
\right]\!\!\right)
$$

Step 6: Walk the Proof

$$
\begin{array}{l}
\texttt{(ga\_first ga\_drop) >>>} \\
\quad \texttt{(ga\_cancell >>>} \\
\quad\quad \Big(\Big[\!\Big[\; \dfrac{\dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}]}{\texttt{y:Int, z:Int} \vdash \texttt{y:Int, z:Int}}\,[\text{Right}] \; \Big]\!\Big] \texttt{ >>>} \\
\Big[\!\Big[\; \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z:Int, b:Int} \vdash \texttt{z:Int, b:Int}}\,[\text{Right}] \qquad \dfrac{\dfrac{\texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int}}{\texttt{z:Int, b : Int} \vdash \texttt{(-) z b:Int}}\,[\text{App'}]}{\texttt{b : Int, z:Int} \vdash \texttt{(-) z b : Int}}\,[\text{Exch}] \; \Big]\!\Big] \; \Big) \Big)
\end{array}
$$

Step 6: Walk the Proof

$$
\begin{array}{l}
\texttt{(ga\_first ga\_drop) >>>} \\
\texttt{(ga\_cancell >>>} \\
\texttt{((ga\_first } [\![\; \dfrac{}{\texttt{y:Int} \vdash \texttt{y:Int}}\,[\text{Var}] \;]\!] \texttt{) >>>} \\
[\![\; \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z:Int, b:Int} \vdash \texttt{z:Int, b:Int}}\,[\text{Right}] \qquad \texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int} }{\dfrac{\texttt{z:Int, b : Int} \vdash \texttt{(-) z b:Int}}{\texttt{b : Int, z:Int} \vdash \texttt{(-) z b : Int}}\,[\text{Exch}]}\,[\text{App'}] \;]\!] \texttt{))}
\end{array}
$$

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
  (ga_cancell >>>
    ((ga_first id) >>>
```

$$[\![\ \text{[Right]}\ \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,\text{[Var]}}{\texttt{z:Int, b:Int} \vdash \texttt{z:Int, b:Int}} \qquad \dfrac{\dfrac{\texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int}}{\texttt{z:Int, b : Int} \vdash \texttt{(-) z b:Int}}\,\text{[App']}}{\texttt{b : Int, z:Int} \vdash \texttt{(-) z b : Int}}\,\text{[Exch]}\ ]\!]\ \texttt{))}$$

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
    (ga_cancell >>>
        ((ga_first id) >>>
```

$$\texttt{(ga\_swap >>>} \quad [\![\quad \dfrac{\dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}\,[\text{Var}]}{\texttt{z:Int, b:Int} \vdash \texttt{z:Int, b:Int}}\,[\text{Right}] \qquad \texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int}}{\texttt{z:Int, b : Int} \vdash \texttt{(-) z b:Int}}\,[\text{App'}] \quad]\!] \quad \texttt{)))}$$

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
  (ga_cancell >>>
    ((ga_first id) >>>
```

$$\texttt{(ga\_swap >>>} \quad [\![\quad \text{[Right]} \; \frac{\text{[Var]} \; \dfrac{}{\texttt{z:Int} \vdash \texttt{z:Int}}}{\texttt{z:Int, b:Int} \vdash \texttt{z:Int, b:Int}} \quad]\!] \quad \texttt{>>> (} \; [\![\quad \frac{}{\texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int}} \quad]\!] \; \texttt{))))}$$

Step 6: Walk the Proof

$$
\begin{array}{l}
\texttt{(ga\_first ga\_drop) >>>} \\
\quad \texttt{(ga\_cancell >>>} \\
\quad\quad \texttt{((ga\_first id) >>>} \\
\quad\quad\quad \texttt{(ga\_swap >>>} \\
\texttt{(ga\_first } [\![\; \mathsf{[Var]} \; \frac{}{\texttt{z:Int} \vdash \texttt{z:Int}} \;]\!] \texttt{) >>> (} [\![\; \frac{}{\texttt{a : Int, b:Int} \vdash \texttt{(-) a b : Int}} \;]\!] \texttt{))))}
\end{array}
$$

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
    (ga_cancell >>>
  ((ga_first id) >>>
     (ga_swap >>>
(ga_first id) >>> ( ⟦ a : Int, b:Int ⊢ (-) a b : Int ⟧ ))))
```

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
    (ga_cancell >>>
  ((ga_first id) >>>
     (ga_swap >>>
(ga_first id) >>> ( [[  a : Int, b:Int ⊢ (-) a b : Int  ]] ))))
```

Hrm, what do we do about that annoying left-over hypothesis we mentioned earlier?

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
   (ga_cancell >>>
  ((ga_first id) >>>
     (ga_swap >>>
(ga_first id) >>> ( ⟦ a : Int, b:Int ⊢ (-) a b : Int ⟧ ))))
```

Hrm, what do we do about that annoying left-over hypothesis we mentioned earlier?
We turn it into a *variable of garrow type*!

Step 6: Walk the Proof

```
(ga_first ga_drop) >>>
  (ga_cancell >>>
    ((ga_first id) >>>
      (ga_swap >>>
        (ga_first id) >>> ( [[ a : Int, b:Int ⊢ (-) a b : Int ]] ))))
```

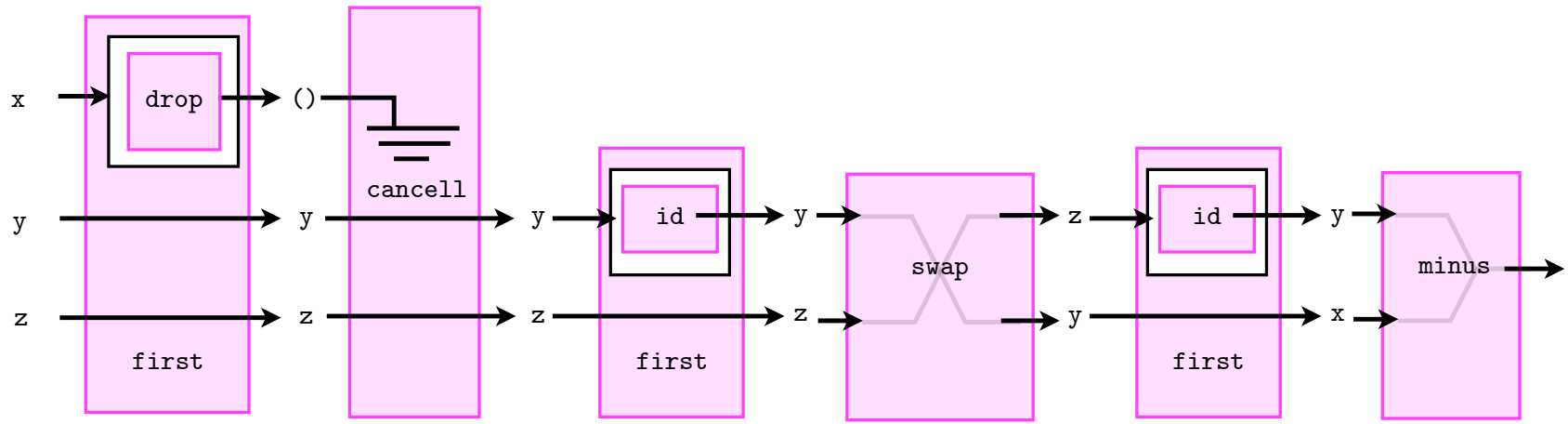
```
flattened
  :: GArrow g (**) u =>
       g (Int**Int)          Int      -- a variable for the flattened "minus"
    -> g (Int**(Int**Int)) Int        -- the resulting term

flattened minus =
  ga_first ga_drop >>>
  ga_cancell       >>>
  ga_first id      >>>
  ga_swap          >>>
  ga_first id      >>>
  minus
```

Boxes and Wires

```
flattened
  :: GArrow g (**) u =>
        g (Int**Int)          Int      -- a variable for the flattened "minus"
     -> g (Int**(Int**Int)) Int        -- the resulting term

flattened minus =
  ga_first ga_drop >>>
  ga_cancell       >>>
  ga_first id      >>>
  ga_swap          >>>
  ga_first id      >>>
  minus
```



For More Information:

http://www.cs.berkeley.edu/~megacz/garrows/

http://arxiv.org/abs/1007.2885v2