Data Types

Haskell → MinHS

```
data Direction = East | West | North | South
data Foo = Foo Int Bool Int
data Tree = Node Tree Tree | Leaf
```

Each nullary dataconstr. is isomorphic to 1

Recursive occurrences?

MinHS types isomorphic to these types?

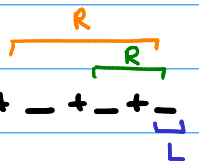
Direction $\simeq$ $1+(1+(1+1))$

Foo $\simeq$ $1 \times (\text{Int} \times (\text{Bool} \times \text{Int})) \cong \text{Int} \times (\text{Bool} \times \text{Int})$

Tree $\simeq$ $\text{rec}\, t.\, ((t \times t)+1)$

MinHS terms of these types?

North $\simeq$ $\text{InR}\,(\text{InR}\,(\text{InL}\,()))$



Foo 3161 False 3141 $\simeq$ $(3161, (\text{False}, 3141))$

Node Leaf Leaf $\simeq$ $\text{roll}\,(\text{InL}\,(\text{roll}\,(\text{InR}\,()), \text{roll}\,(\text{InR}\,())))$??

Last term is a bit tricky. Notice that when using InL and InR, one of the types in the sum type $\tau_1 + \tau_2$ is free/arbitrary. So

choose this as τ_1

$\text{InR}\,() : (\text{rec}\, t.\,((t \times t)+1), \text{rec}\, t.\,((t \times t)+1)) + 1$

$\cong (t \times t + 1)[t := \text{rec}\, t.\,((t \times t)+1)]$

$\Rightarrow$ $\text{roll}\,(\text{InR}\,()) : \text{rec}\, t.\,((t \times t) + 1)$ ← This is how we write Leaf

$\Rightarrow$ $(\text{roll}\,(\text{InR}\,()), \text{roll}\,(\text{InR}\,())) : (\text{rec}\, t.\,((t \times t)+1)) \times (\text{rec}\, t.\,((t \times t)+1))$

$\Rightarrow$ $\text{InL}\,((\text{roll}\,(\text{InR}\,()), \text{roll}\,(\text{InR}\,()))) : ((\text{rec}\, t.\,(t \times t)+1) \times (\text{rec}\, t.\,(t \times t)+1)) + 1$

$\cong ((t \times t)+1)[t = \text{rec}\, t.\,(t \times t)+1]$ Choose this as τ_2

$\Rightarrow$ $\text{roll}\,(\text{InL}\,(\ldots)) : \text{rec}\, t.\,((t \times t)+1)$

This is how we write Node Leaf Leaf.

Recursive Types

rec t. (Int + t):

$$\dfrac{\dfrac{}{3161 : \text{Int}}}{\text{InL } 3161 : \text{Int} + (\text{rec t. (Int + t)})}$$
$$\text{roll (InL } 3161) : \text{rec t. (Int + t)}$$

T_2 is free in the typing rule for InL so I can let it be whatever I want.

How about rec t. (Int x t)? Does not have any finite terms!
But can construct using Recfuns:

recfun f x = roll (x, f x) where x : Int
: (Int, rec t. (Int x t))

Isomorphism Equivalence

Which are isomorphic assuming a strict semantics? ⇒ Finite terms only!

Impossible to construct any finite terms

rec t. (t x t) 1 rec t. (t x 1) → Impossible to construct any finite terms

0 1 x 0 rec t. (t + 1)

≅ 0

only finite term is roll (InR ()) ⟷ () (bij.)

Curry-Howard Isomorphism (programs ≅ proofs!)

Type is A+B → B+A

Prove A ∨ B ⇒ B ∨ A by writing a suitable MiniHS term:

recfun f x = case x of InL x_A → InR x_A : B+A
InR x_B → InL x_B : B+A

This program is a PROOF that A ∨ B ⇒ B ∨ A !