

Proof and Programs

Informatics 1

Functional Programming Lecture 17

Don Sannella

University of Edinburgh

26 November 2012

Last tutorial this week, usual time/place

Extra tutorial 1:10-2:00 / 2:10-3:00 on Wed 28 Nov
Appleton Tower, room 5.04

Do the extra tutorial exercise (TBA) and bring your answer

Extra tutorial 1:10-2:00 / 2:10-3:00 on Wed 5 Dec

Extra tutorial Wed 12 Dec?

What is a Proof?

```
square :: Integer -> Integer
```

```
square x = x * x
```

```
prop_squares :: Integer -> Integer -> Bool
```

```
prop_squares x y =
```

```
  square (x + y) == x * x + 2 * x * y + y * y
```

```
*Main> quickCheck prop_squares
```

```
+++ OK, passed 100 tests.
```

```
*Main>
```

What is a Proof?

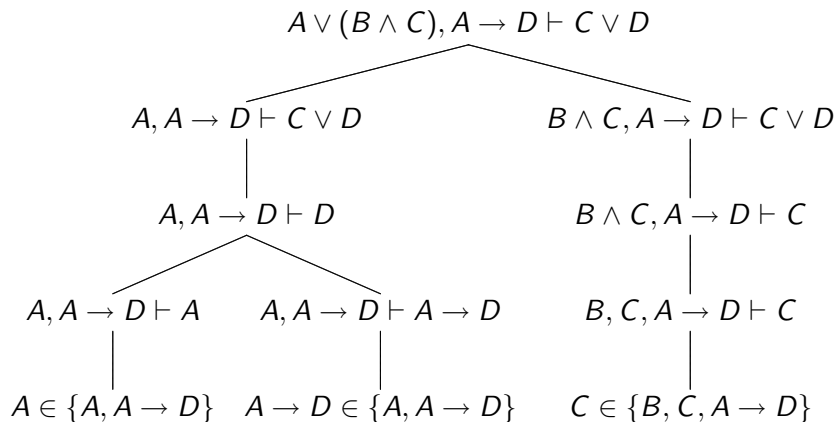
According to *Thinking Mathematically* (1982)

1. Convince yourself
2. Convince a friend
3. Convince an enemy

What about...

Convince a computer

What is a Proof?



Why do Proof?



Rule of Leibniz

- ▶ Indiscernability of Identity
- ▶ Identity of Indiscernables
- ▶ Equality is reflexive:
 $x = x$
- ▶ Equals may be substituted
for equals



The number I am thinking of
now is **not** the number I am
thinking of *now*.



`i++ != i++`

1. President of the United States = Barack Obama
2. Abraham Lincoln was President of the United States in 1861
3. Abraham Lincoln was Barack Obama in 1861

A Simple Function

```
square :: Integer -> Integer
```

```
square x = x * x
```

```
prop_squares :: Integer -> Integer -> Bool
```

```
prop_squares x y =
```

```
    square (x + y) == x * x + 2 * x * y + y * y
```

$$x + 0 = x$$

$$x * 1 = x$$

$$x + y = y + x$$

$$x * y = y * x$$

$$(x + y) + z = x + (y + z)$$

$$(x * y) * z = x * (y * z)$$

$$x * (y + z) = x * y + x * z$$

$$2 = 1 + 1$$

Algebraic Proof

square (x + y) = x * x + (2 * (x * x) + y * y)

square (x + y)

= (x + y) * (x + y) *-- Distrib.*

= (x + y) * x + (x + y) * y *-- Commut.*

= x * (x + y) + (x + y) * y *-- Commut.*

= x * (x + y) + y * (x + y) *-- Distrib.*

= (x * x + x * y) + y * (x + y) *-- Distrib.*

= (x * x + x * y) + (y * x + y * y) *-- Assoc.*

= x * x + (x * y + (y * x + y * y)) *-- Commut.*

= x * x + (x * y + (x * y + y * y))

Algebraic Proof

$$\begin{aligned} & x*x + (2*(x*y) + y*y) \\ &= x*x + ((1+1) * (x*y) + y*y) && \text{-- Commut.} \\ &= x*x + ((x*y) * (1+1) + y * y) && \text{-- Distrib.} \\ &= x*x + (((x*y) * 1 + (x*y) * 1) + y*y) && \text{-- Id.} \\ &= x*x + ((x*y + (x*y) * 1) + y*y) && \text{-- Id.} \\ &= x*x + ((x*y + x*y) + y*y) && \text{-- Assoc.} \\ &= x * x + (x * y + (x * y + y * y)) \end{aligned}$$

Natural Numbers

```
data Nat = Zero
         | Succ Nat
```

```
(+) :: Nat -> Nat -> Nat
```

```
x + Zero = x
```

```
x + Succ y = Succ (x + y)
```

```
(*) :: Nat -> Nat -> Nat
```

```
x * Zero = Zero
```

```
x * Succ y = x + (x * y)
```

```
one = Succ Zero
```

```
two = Succ one
```

```
three = Succ two
```

```
four = Succ three
```

If I have two beans, and
I add two more beans,
what do I have?



Proof!

(+) :: Nat -> Nat -> Nat

x + Zero = x

x + Succ y = Succ (x + y)

(*) :: Nat -> Nat -> Nat

x * Zero = Zero

x * Succ y = x + (x * y)

two + two

= Succ (Succ Zero) + Succ (Succ Zero)

= Succ (Succ (Succ Zero) + Succ Zero)

= Succ (Succ (Succ (Succ Zero + Zero)))

= Succ (Succ (Succ (Succ Zero)))

= four

Cutting-Edge Mathematics

Prove that:

$$\text{Zero} + x = x$$

$$(+)\ :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$$

$$x + \text{Zero} = x$$

$$x + \text{Succ } y = \text{Succ } (x + y)$$

$$(*)\ :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$$

$$x * \text{Zero} = \text{Zero}$$

$$x * \text{Succ } y = x + (x * y)$$

Uh-oh! Our rules aren't enough!

Induction

To prove that a statement is true for all natural numbers:

1. Prove it is true for Zero;
2. Assuming it is true for n , show it is true for Succ n .

Suppose

```
p :: Nat -> Bool
p Zero = True
p (Succ n) | p n = True
```

Then

```
p n = True
```

Identity of Addition

Prove that:

$$\text{Zero} + x = x$$

Base case:

$$\text{Zero} + \text{Zero} = \text{Zero}$$

Step case:

Supposing that

$$\text{Zero} + x = x$$

We have

$$\begin{aligned} \text{Zero} + \text{Succ } x &= \text{Succ } (\text{Zero} + x) \\ &= \text{Succ } x \end{aligned}$$

Commutativity

Prove that:

$$x + y = y + x$$

Base case:

$$x + \text{Zero} = x$$

$$\text{Zero} + x = x$$

Step case:

Supposing that

$$x + y = y + x$$

We have

$$x + \text{Succ } y = \text{Succ } (x + y)$$

$$\text{Succ } y + x =$$

Uh-oh! We need a lemma.

Commutativity Lemma

Prove that:

$$\text{Succ } y + x = \text{Succ } (y + x)$$

Base case:

$$\text{Succ } y + \text{Zero} = \text{Succ } y$$

$$\text{Succ } (y + \text{Zero}) = \text{Succ } y$$

Step case:

Supposing that

$$\text{Succ } y + x = \text{Succ } (y + x)$$

We have

$$\begin{aligned} & \text{Succ } y + \text{Succ } x \\ &= \text{Succ } (\text{Succ } y + x) \\ &= \text{Succ } (\text{Succ } (y + x)) \\ & \text{Succ } (y + \text{Succ } x) = \\ & \quad \text{Succ } (\text{Succ } (y + x)) \end{aligned}$$

Commutativity again

Prove that:

Base case:

$$x + \text{Zero} = x$$

$$\text{Zero} + x = x$$

Step case:

Supposing that

$$x + y = y + x$$

We have

$$x + \text{Succ } y = \text{Succ } (x + y)$$

$$\text{Succ } y + x = \text{Succ } (y + x)$$

$$= \text{Succ } (x + y)$$

```
data [a] = []  
         | a : [a]
```

```
(++) : [a] -> [a] -> [a]
```

```
[] ++ xs      = xs
```

```
(x : xs) ++ ys = x : (xs ++ ys)
```

```
reverse :: [a] -> [a]
```

```
reverse []      = []
```

```
reverse (x : xs) = reverse xs ++ [x]
```

Associativity of append

Prove that:

$$xs ++ (ys ++ zs) = (xs ++ ys) ++ zs$$

► Base case

$$[] ++ (ys ++ zs) = ys ++ zs$$

$$([] ++ ys) ++ zs = ys ++ zs$$

► Step case

Supposing that

$$xs ++ (ys ++ zs) = (xs ++ ys) ++ zs$$

We have

$$\begin{aligned}(x : xs) ++ (ys ++ zs) \\ &= x : (xs ++ (ys ++ zs))\end{aligned}$$

$$\begin{aligned}((x : xs) ++ ys) ++ zs \\ &= (x : (xs ++ ys)) ++ zs \\ &= x : ((xs ++ ys) ++ zs) \\ &= x : (xs ++ (ys ++ zs))\end{aligned}$$

Reversing Append: Base case

Prove that:

$$\text{reverse } (xs ++ ys) = \text{reverse } ys ++ \text{reverse } xs$$
$$\text{reverse } ([] ++ ys) = \text{reverse } ys$$
$$\begin{aligned} \text{reverse } ys ++ \text{reverse } [] &= \text{reverse } ys ++ [] \\ &= \end{aligned}$$

Reversing Append: Step case

Supposing that

$$\text{reverse } (xs ++ ys) = \text{reverse } ys ++ \text{reverse } xs$$

We have

$$\begin{aligned} & \text{reverse } ((x : xs) ++ ys) \\ &= \text{reverse } (x : (xs ++ ys)) \\ &= \text{reverse } (xs ++ ys) ++ [x] \\ &= (\text{reverse } ys ++ \text{reverse } xs) ++ [x] \\ &= \text{reverse } ys ++ (\text{reverse } xs ++ [x]) \end{aligned}$$

$$\begin{aligned} & \text{reverse } ys ++ \text{reverse } (x : xs) \\ &= \text{reverse } ys ++ (\text{reverse } xs ++ [x]) \end{aligned}$$

Prove that:

`reverse (reverse xs) = xs`

Summary

1. Proof is challenging, **mechanical**
2. Proof shows our programs are correct **rigorously**.
3. Haskell allows equational proof
4. Haskell recursion requires **induction**
5. $2 + 2 = 4!$