Informatics 1

Introduction to Computation

Functional Programming Lecture 1

Functions

Philip Wadler

University of Edinburgh

Part I

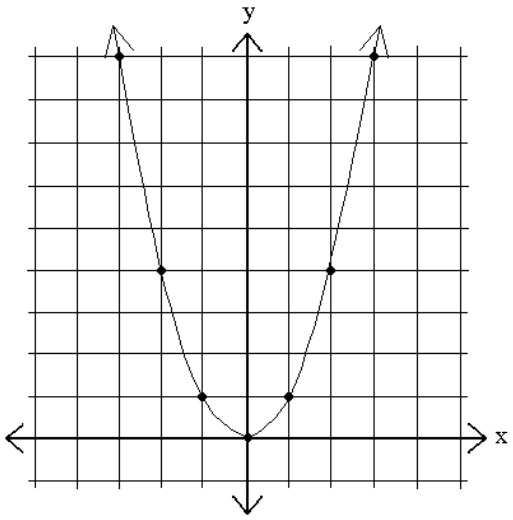
Functions

What is a function?

- A recipe for generating an output from inputs:
  "Multiply a number by itself"
- A set of (input, output) pairs:
  (1,1) (2,4) (3,9) (4,16) (5,25) ...
- An equation:

$$f\ x = x^2$$

- A graph relating inputs to output (for numbers only):

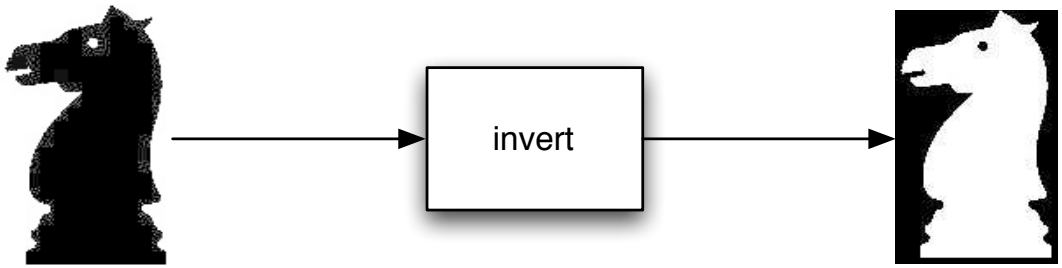


Kinds of data

- Integers: `42`, `-69`
- Floats: `3.14`
- Characters: `'h'`
- Strings: `"hello"`
- Booleans: `True`, `False`
- Pictures: 

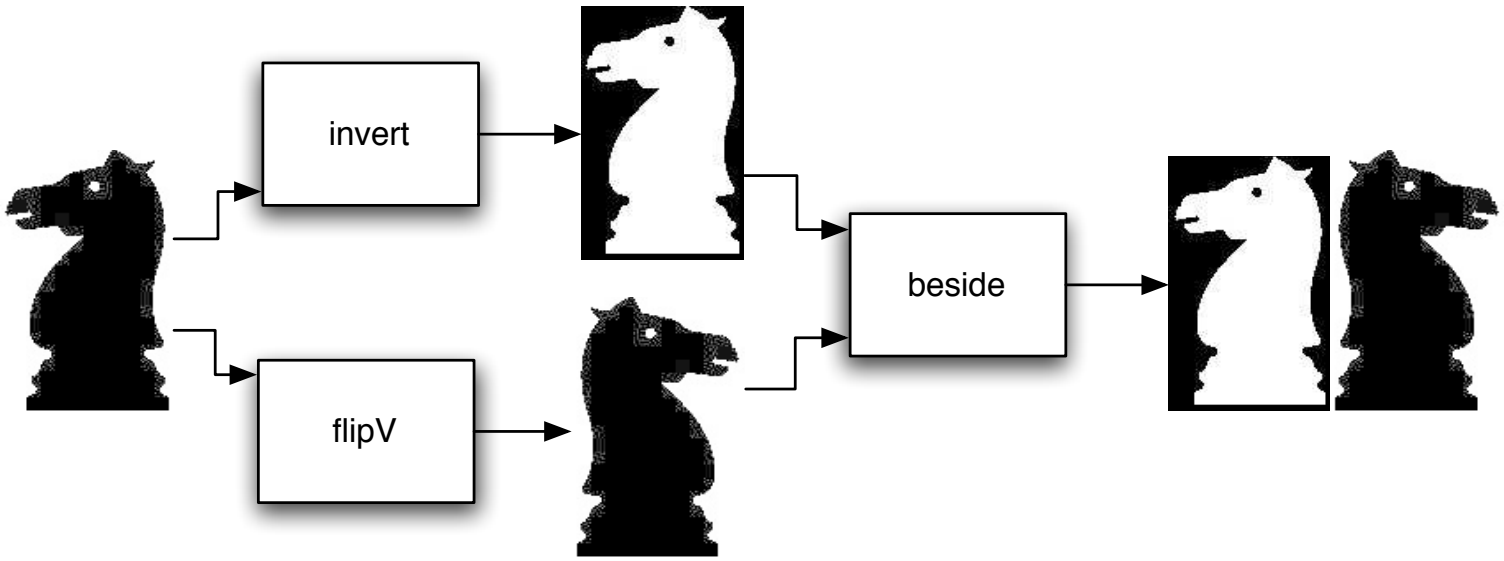
Applying a function

```
invert :: Picture -> Picture
knight :: Picture

invert knight
```



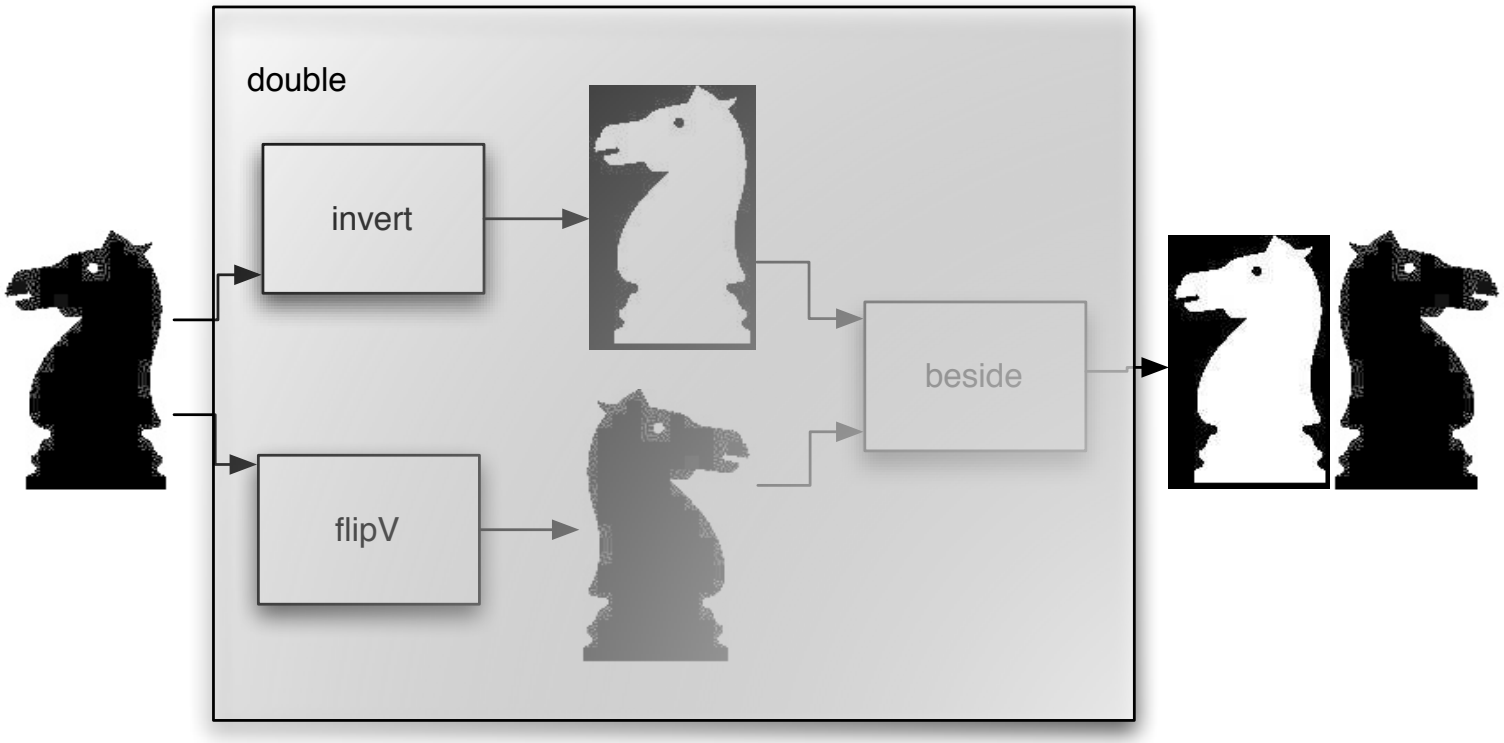
Composing functions

```
beside :: Picture -> Picture -> Picture
flipV :: Picture -> Picture
invert :: Picture -> Picture
knight :: Picture

beside (invert knight) (flipV knight)
```



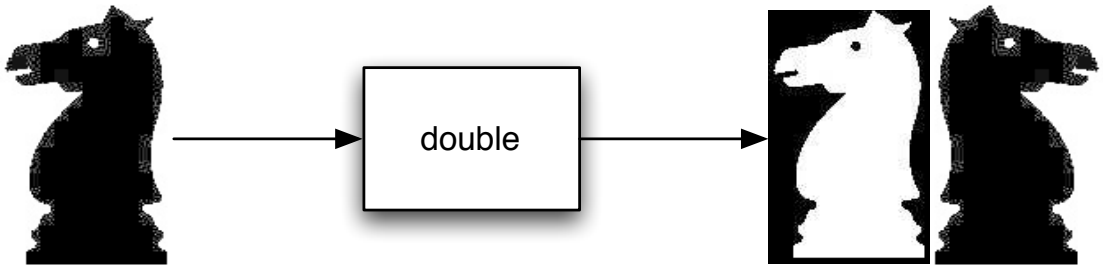
Defining a new function

```
double :: Picture -> Picture
double p  =  beside (invert p) (flipV p)

double knight
```



Defining a new function

```
double :: Picture -> Picture
double p  =  beside (invert p) (flipV p)

double knight
```



Terminology

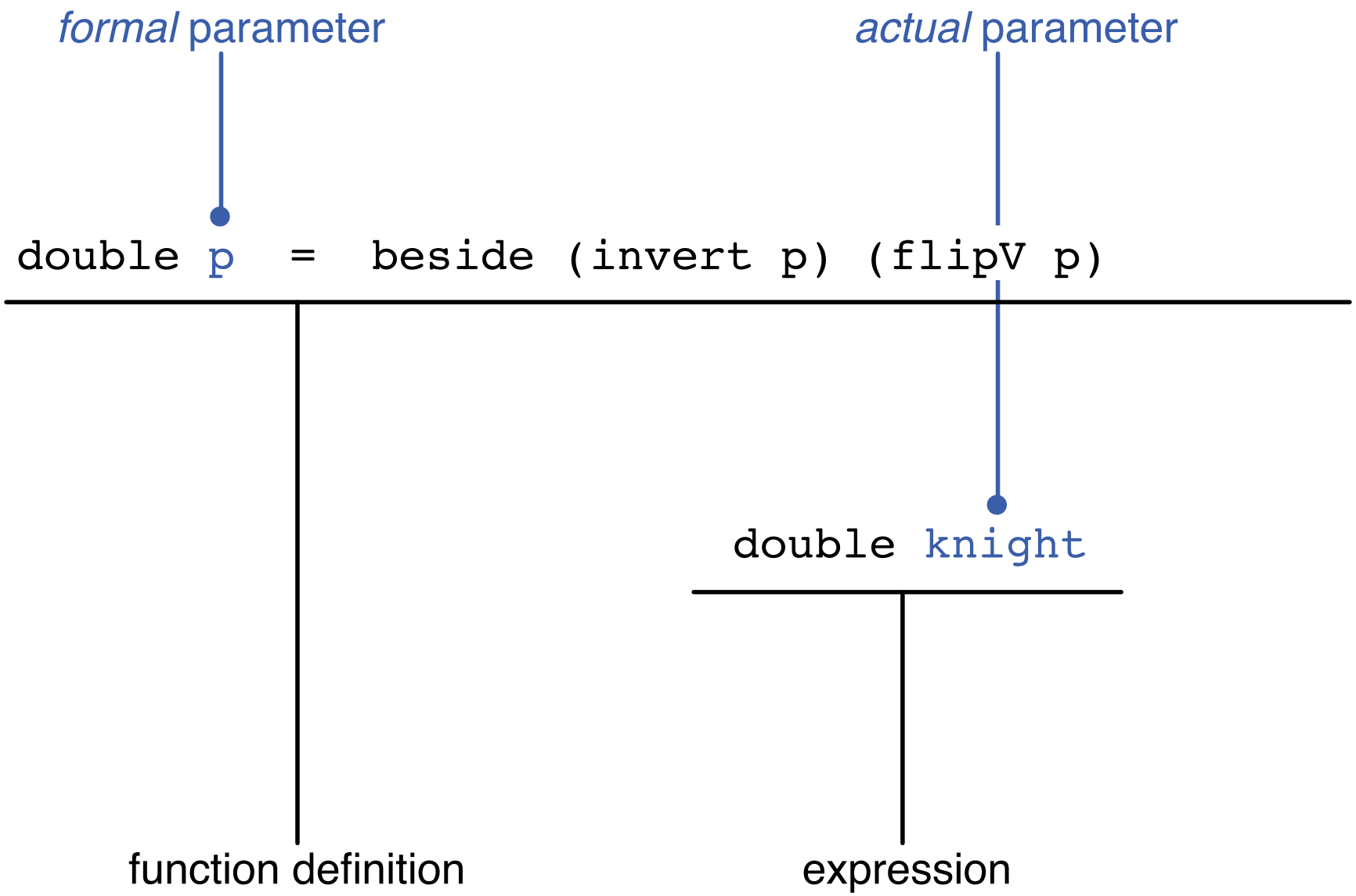
Type signature

```
double :: Picture -> Picture
```

Function declaration

```
double p  =  beside (invert p) (flipV p)
```

Terminology



Part II

Functions on numbers

Operations on numbers

```
[jitterbug]dts: ghci
GHCi, version 7.4.2: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude> 3+3
6
Prelude> 3*3
9
Prelude>
```

Functions over numbers

squares.hs

```
square :: Integer -> Integer
square x  =  x * x

pyth :: Integer -> Integer -> Integer
pyth a b =  square a + square b
```

Testing our functions

```
[jitterbug]dts: ghci squares.hs
GHCi, version 7.4.2: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
[1 of 1] Compiling Main             ( squares.hs, interpreted )
Ok, modules loaded: Main.
*Main> square 3
9
*Main> pyth 3 4
25
*Main>
```

A few more tests

```
*Main> square 0
0
*Main> square 1
1
*Main> square 2
4
*Main> square 3
9
*Main> square 4
16
*Main> square (-3)
9
*Main> square 10000000000
100000000000000000000
```

Declaration and evaluation

Declaration (file squares.hs)

```
square :: Integer -> Integer
square x  =  x * x

pyth :: Integer -> Integer -> Integer
pyth a b  =  square a + square b
```

Evaluation

```
[jitterbug]dts: ghci squares.hs
GHCi, version 7.4.2: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
[1 of 1] Compiling Main             ( squares.hs, interpreted )
Ok, modules loaded: Main.
*Main> pyth 3 4
25
*Main>
```

Part III

The Rule of Leibniz

The Rule of Leibniz

```
square :: Integer -> Integer
square x  =  x * x

pyth :: Integer -> Integer -> Integer
pyth a b  =  square a + square b
```

```
  pyth 3 4
=
  square 3 + square 4
=
  3*3 + 4*4
=
  9 + 16
=
  25
```

The Rule of Leibniz

- Identity of Indiscernables: "No two distinct things exactly resemble one another." — Leibniz

  That is, two objects are identical if and only if they satisfy the same properties.

- "A difference that makes no difference is no difference." — Spock

- "Equals may be substituted for equals." — My high school teacher

Numerical operations are functions

```
(+) :: Integer -> Integer -> Integer
(*) :: Integer -> Integer -> Integer
```

```
Main*> 3+4
7
Main*> 3*4
12
```

```
3 + 4   stands for   (+) 3 4
3 * 4   stands for   (*) 3 4
```

```
Main*> (+) 3 4
7
Main*> (*) 3 4
12
```

Precedence and parentheses

Function application takes *precedence* over infix operators.

(Function applications *binds more tightly than* infix operators.)

```
  square 3 + square 4
=
  (square 3) + (square 4)
```

Multiplication takes *precedence* over addition.

(Multiplication *binds more tightly than* addition.)

```
  3*3 + 4*4
=
  (3*3) + (4*4)
```

Associativity

Addition is *associative*.

```
   3 + (4 + 5)
=
   3 + 9
=
   12
=
   7 + 5
=
   (3 + 4) + 5
```

Addition *associates to the left*.

```
   3 + 4 + 5
=
   (3 + 4) + 5
```

Part IV

QuickCheck

QuickCheck properties

squares_prop.hs

```
import Test.QuickCheck

square :: Integer -> Integer
square x  =  x * x

pyth :: Integer -> Integer -> Integer
pyth a b =  square a + square b

prop_square :: Integer -> Bool
prop_square x  =
  square x >= 0

prop_squares :: Integer -> Integer -> Bool
prop_squares x y  =
  square (x+y) == square x + 2*x*y + square y

prop_pyth :: Integer -> Integer -> Bool
prop_pyth x y  =
  square (x+y) == pyth x y + 2*x*y
```

```
[jitterbug]dts: ghci squares_prop.hs
GHCi, version 7.4.2: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
[1 of 1] Compiling Main             ( squares_prop.hs, interpreted
*Main> quickCheck prop_square
Loading package array-0.4.0.0 ... linking ... done.
Loading package deepseq-1.3.0.0 ... linking ... done.
Loading package old-locale-1.0.0.4 ... linking ... done.
Loading package time-1.4 ... linking ... done.
Loading package random-1.0.1.1 ... linking ... done.
Loading package containers-0.4.2.1 ... linking ... done.
Loading package pretty-1.1.1.0 ... linking ... done.
Loading package template-haskell ... linking ... done.
Loading package QuickCheck-2.5.1.1 ... linking ... done.
+++ OK, passed 100 tests.
*Main> quickCheck prop_squares
+++ OK, passed 100 tests.
*Main> quickCheck prop_pyth
+++ OK, passed 100 tests.
```

Part V

The Rule of Leibniz (reprise)

Gottfried Leibniz, 1646–1716

Gottfried Leibniz, 1646–1716

Anticipated symbolic logic, discovered calculus (independently of Newton), introduced the term “monad” to philosophy.

> “The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right.”
>
> Gottfried Leibniz, 1646–1716

> “In symbols one observes an advantage in discovery which is greatest when they express the exact nature of a thing briefly and, as it were, picture it; then indeed the labor of thought is wonderfully diminished.”
>
> Gottfried Leibniz, 1646–1716