Informatics 1

Functional Programming Lecture 1

Functional Programming, Types and Values

Don Sannella

University of Edinburgh

Part I

Functional Programming

Computation and language

“Computer science is no more about computers than astronomy is about telescopes.”

Edsgar Dijkstra, 1930–2002

“Language shapes the way we think, and determines what we can think about.”

Benjamin Lee Whorf, 1897–1941

“The limits of my language mean the limits of my world.”

Ludwig Wittgenstein, 1889–1951

“A language that doesn’t affect the way you think about programming, is not worth knowing.”

Alan Perlis, 1922–1990

Programming paradigms

- Functional programming (FP)

  Agda, Coq, Elm, Erlang, F#, Haskell, Hope, Idris, Isabelle, Javascript, Lisp, ML, OCaml, Racket, Scala, Scheme

  - Higher level
  - More compact programs

- Object-oriented (OO)

  C++, F#, Java, Javascript, OCaml, Perl, Python, Ruby, Scala

  - More widely used
  - More libraries

FP in industry

- Google MapReduce
- Facebook Haxl, Haskell library for concurrency
- Twitter backend implemented in Scala
- Financial institutions Barclays, Standard Chartered, Credit Suisse, Jane Street, Tsuro Capital
- Cryptocurrency IOHK Cardano (Plutus), Tezos (Liquidity), Simplicity
- Ericsson AXE phone switch in Erlang (up 99.9999999% time)

FP in teaching

- FP taught first in Edinburgh, Oxford, Cambridge, Imperial, . . .
- Puts experienced and inexperienced programmers on an equal footing
- Operate on data structure *as a whole* rather than *piecemeal*

FP influence on other languages

- Garbage collection Java, Javascript, C#, Python, Ruby, Swift
- Lambda expressions Java, Javascript, C#, Python, Ruby, Swift, Excel
- Generics Java, C#, Swift, Go
- Type classes Java bounds, C++ concepts, Swift protocols
- List comprehensions C#, Python

Part II

Values and Types

We compute with values

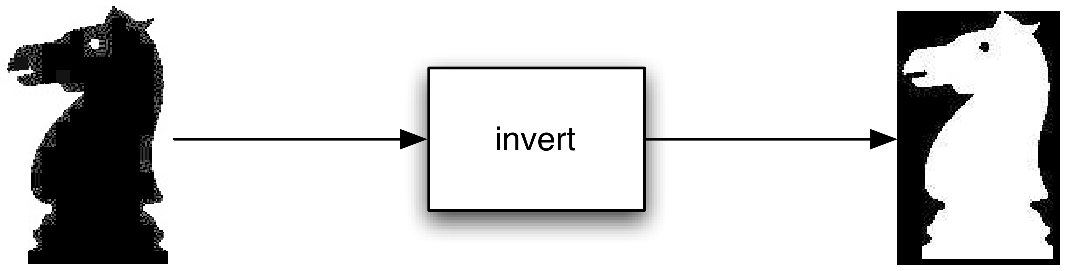
```
42

"Hello!"

False

28 Jun 1963

Julius Caesar

sqrt

+

length

isAlive
```

Every value has a type $v :: t$

```
42 :: Int

"Hello!" :: String

False :: Bool

28 Jun 1963 :: Date

Julius Caesar :: Person

sqrt :: Float -> Float

+ :: Int -> Int -> Int

length :: String -> Int

isAlive :: Person -> Bool
```

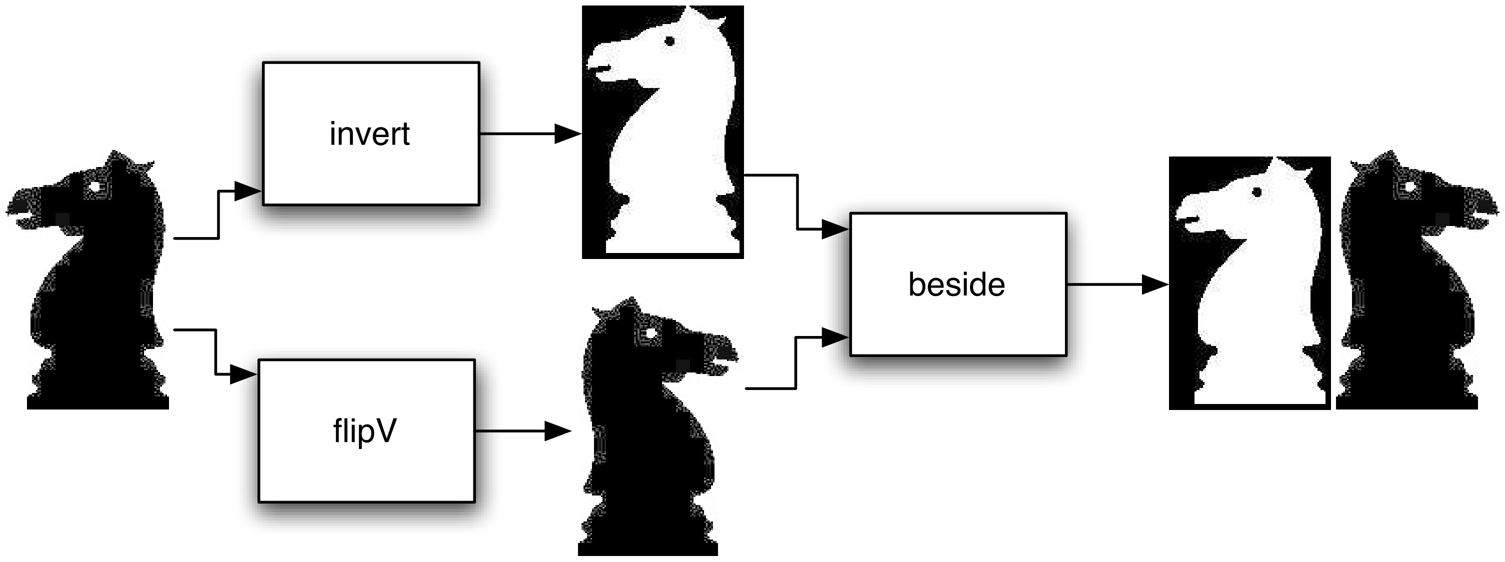
Applying a function

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

invert knight
```



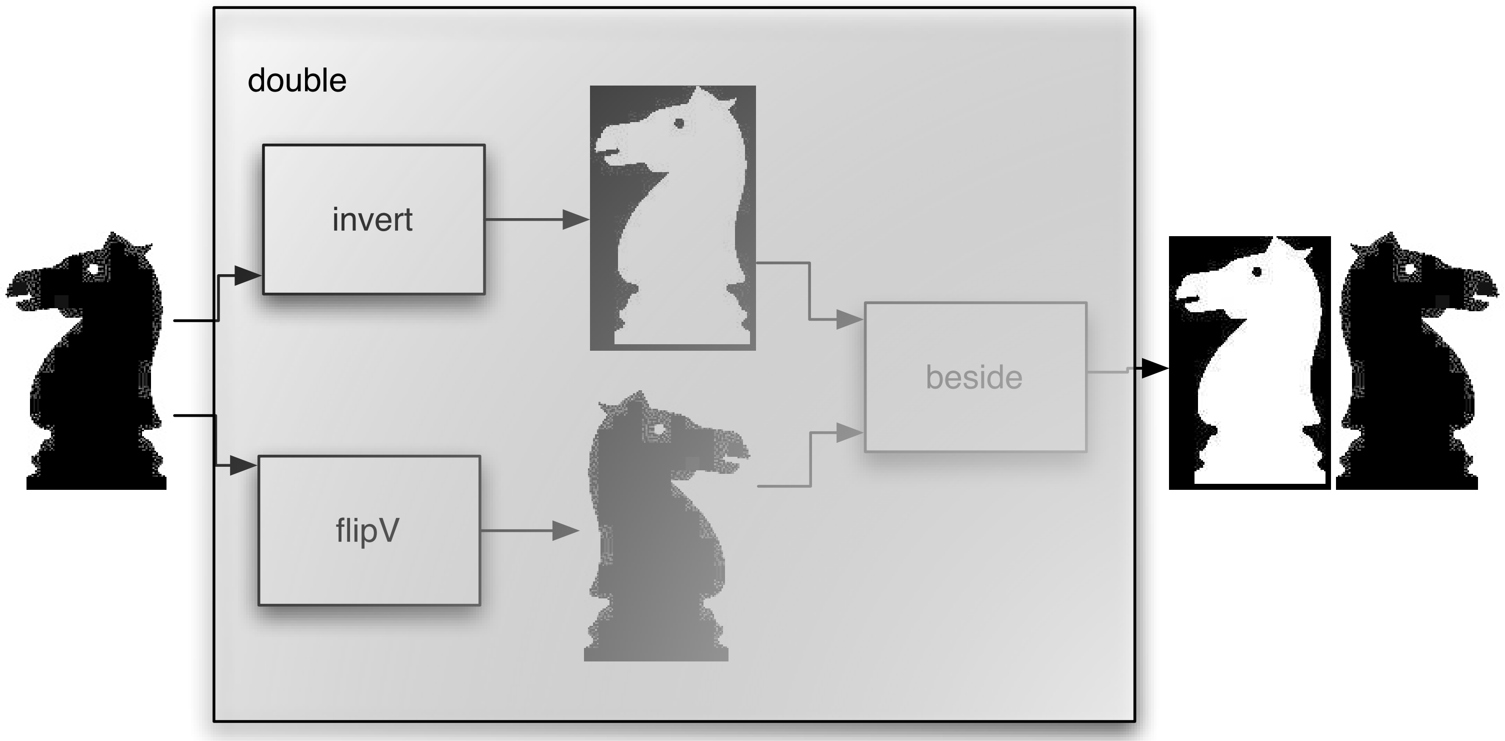
Combining functions

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

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



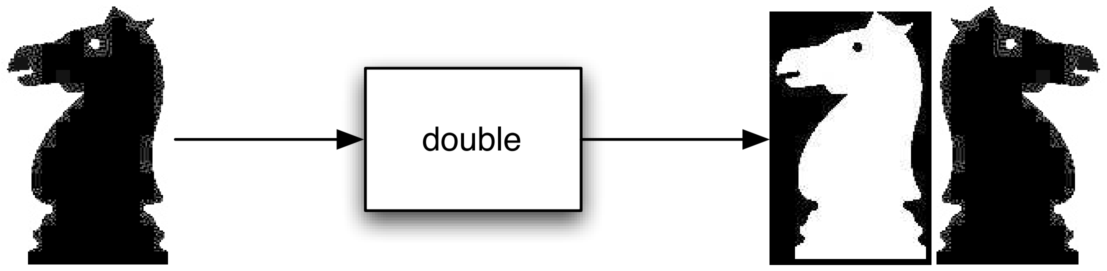
Defining a new function

```
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
```

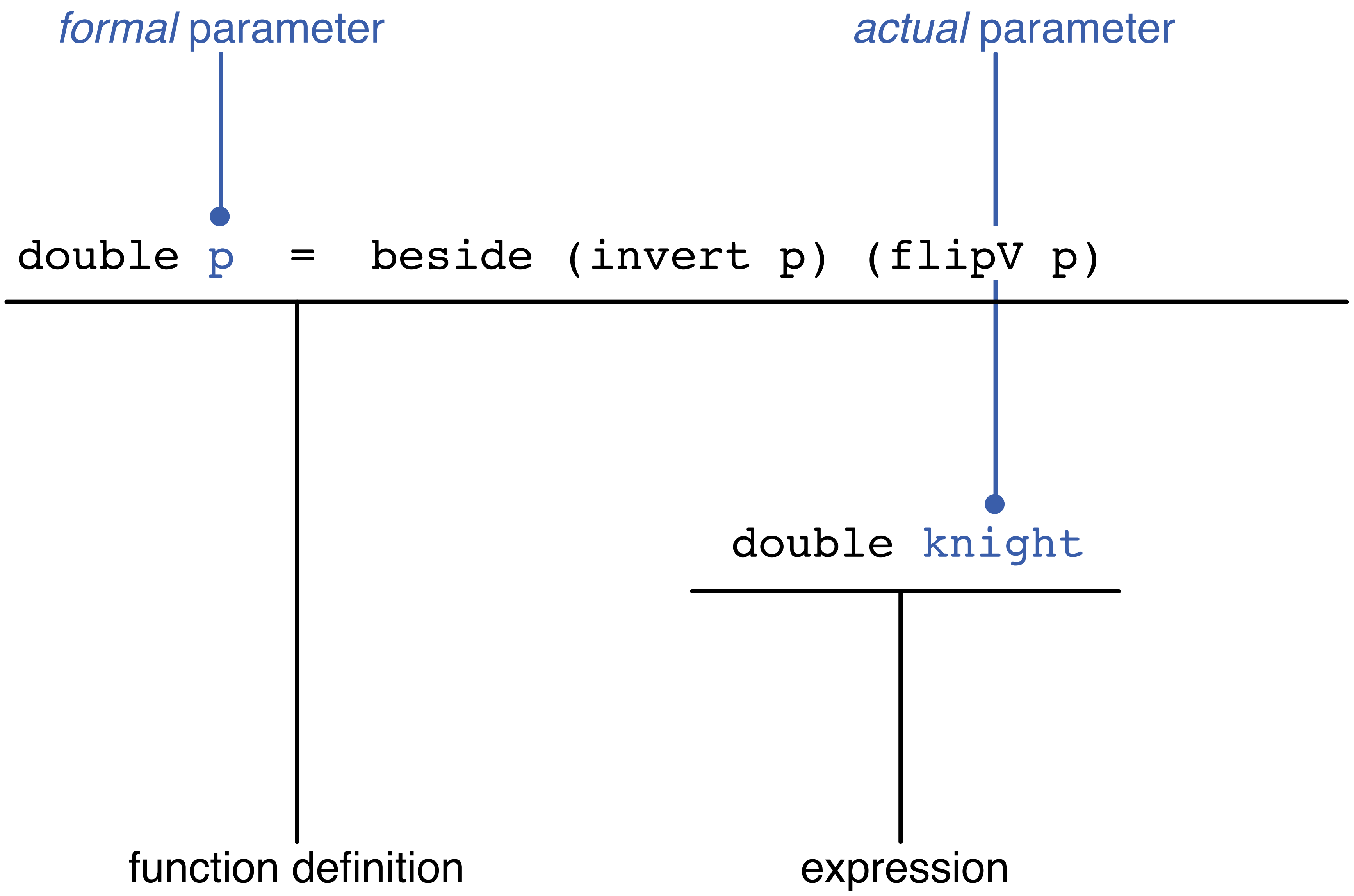
Function definition

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

function name

function body

Terminology



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

```
double knight
```

Defining a new type

```
type PicTrans = Picture -> Picture

double :: PicTrans
double p  =  beside (invert p) (flipV p)


type Trans a = a -> a

double :: Trans Picture
double p  =  beside (invert p) (flipV p)


data Weekday = Monday | Tuesday | Wednesday | Thursday
                      | Friday | Saturday | Sunday

> Monday == Thursday
False
```