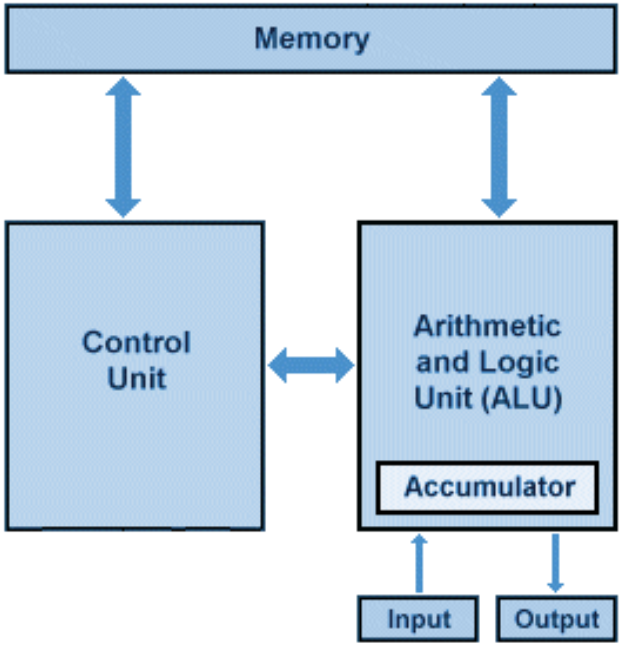
Chapter 0
Prelude

Von Neumann Architecture



⇒

Procedural

Object-oriented

Imperative Programming
= Assignment + Control Flow + IO

Cobol

Smalltalk

Pascal

Fortran

Scheme

Algol

Java

C

:= set!
= <-

Assignment statement

Imperative Programming

Assignment

Sequencing

```
do-something-useful() {
  do-this();
  do-that();
  while(this-is-true()) {
    do-foo();
    do-bar() }
  continue();
  if(machine-status-is-whatever())
    do-something-special()
 }
```

Statements & procedures
have side effects in memory

Assignment = "Root of All Evil"

```
<-    :=    set!    =
```

Functional Decomposition	↔	"Side Effects"
Concurrency	↔	"Race Conditions"
Proving Correctness	↔	"Variables that vary"

The problem with computer science is that variables actually vary [C. Strachey]

Assignment as Root of All Evil

... led to FP

John Backus

- Inventor of Fortran

- 1977 Turing Award Lecture:

  "Can Programming Be Liberated from the Von Neumann Style ?"

Implicit Side Effects

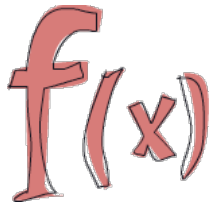
```
cin >> ...

                    cout << ...

rocket.launch()

                        socket.send()
```

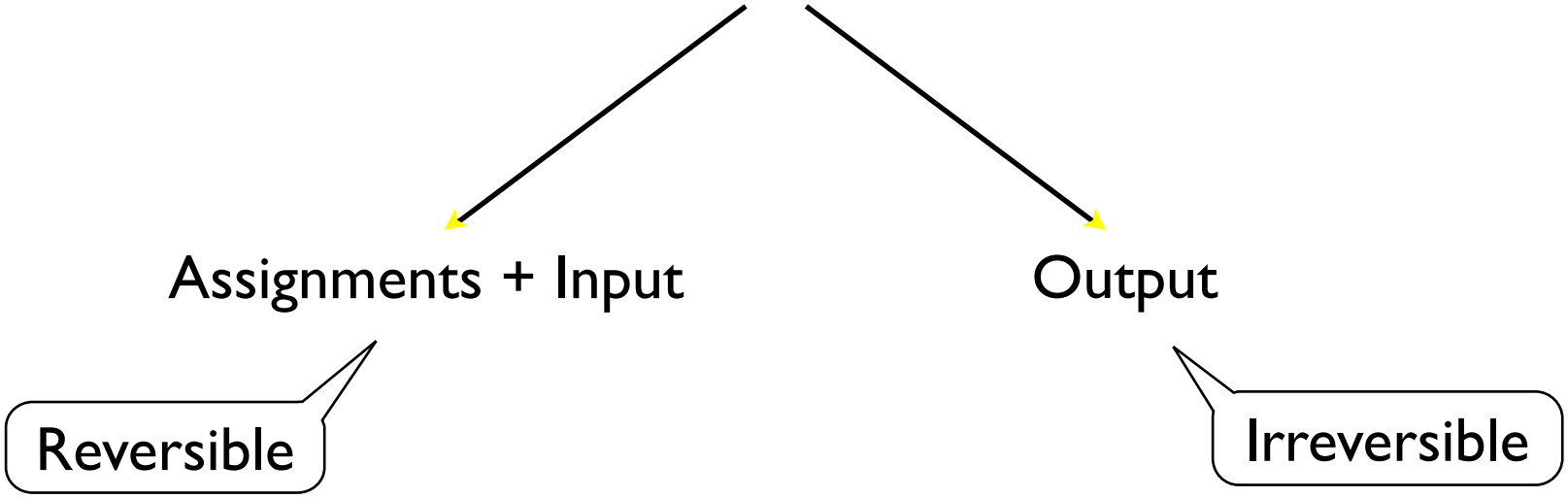
Referential Transparency



The defining property of Functional Programming

$$f(x) = f(x)$$

Enemies of Referential Transparency



History of Functional Languages

λ-calculus (Alonzo Church - 1936)

Lisp (McCarthy - 1958)

Scheme (Steele - 1975)

SASL (David Turner - 1976)

FP (John Backus - 1977)

ML (Robin Milner - 197x)

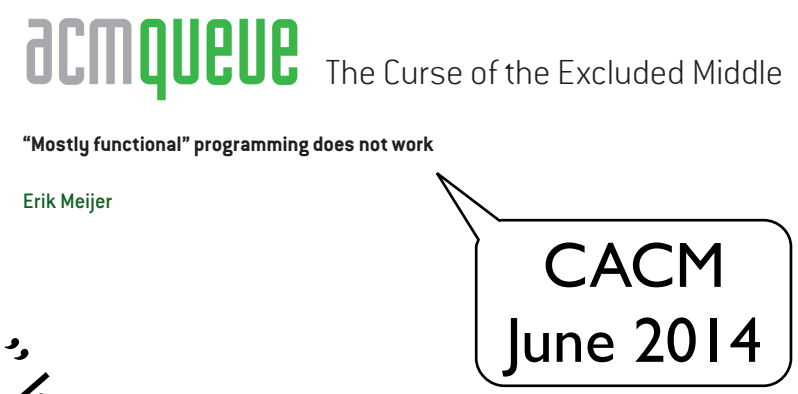
Miranda (David Turner - 1985)

Erlang (Joe Armstrong - 1986)

Haskell (Hudak & al - 1988)

"pure" vs. "impure" languages

strict vs. nonstrict languages



A Note on Typing ...

is about program text

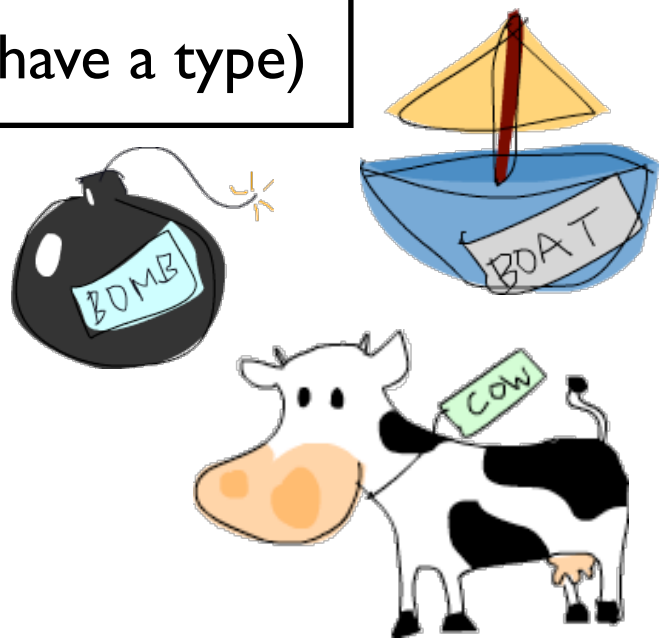
is about runtime heap

Static Typing
=
(variables have a type)

Dynamic Typing
=
(values have a type)

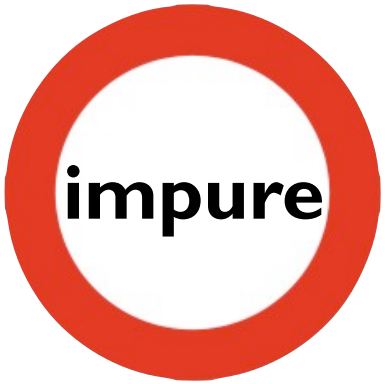
Type Checking

Type Inferencing



Course Contents

- Programming in Haskell
- Advanced FP Topics



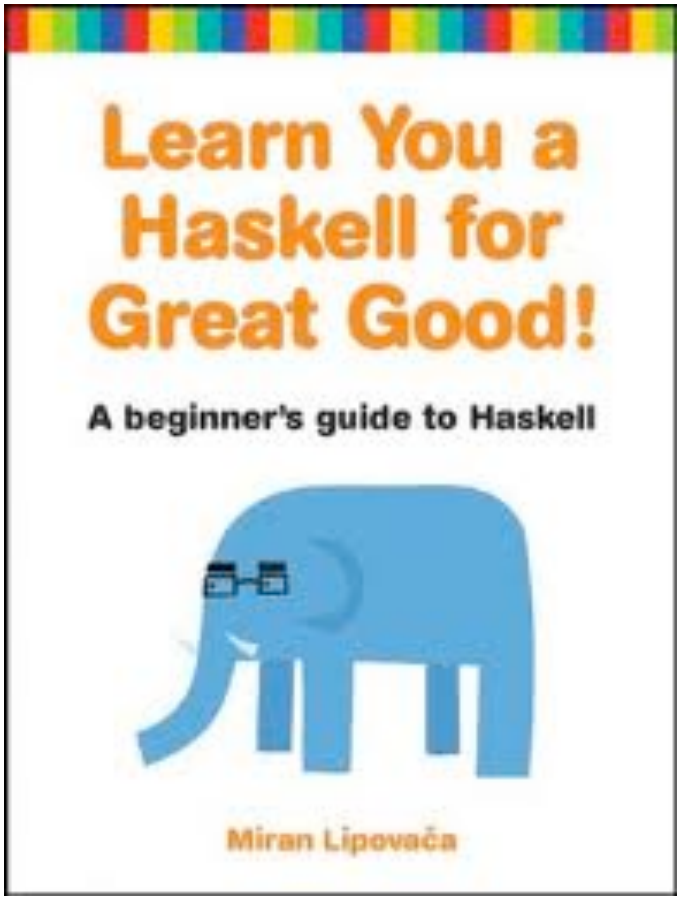
Expose you to
- static typing tradition
- equational reasoning
- monadic side-effects
- concurrency models

Haskell

- Designed by committee
- Pure
- Non-Strict (Lazy)
- Statically Typed
- Recently revisited

Haskell Curry

Reference Book for First Part



[Miran Lipovača]
Learn You a Haskell for Great Good!

Chapter 0