Functional Programming in Education

George Wilson

Data61/CSIRO

george.wilson@data61.csiro.au

22nd May 2018

	Content
Week 1	Basic expressions
Week 2	procedure declarations
Week 3	if-statement
Week 4	while-statement
Week 5	for-statement
...	

Structure and Interpretation of Computer Programs

Second Edition

Harold Abelson and
Gerald Jay Sussman
with Julie Sussman

```scheme
(define (map proc items)
  (if (null? items)
      nil
      (cons (proc (car items))
            (map proc (cdr items)))))
```

```scheme
(define (abs x)
  (cond ((> x 0) x)
        ((= x 0) x)
        ((< x 0) (- x))))
```

Evaluation by substitution

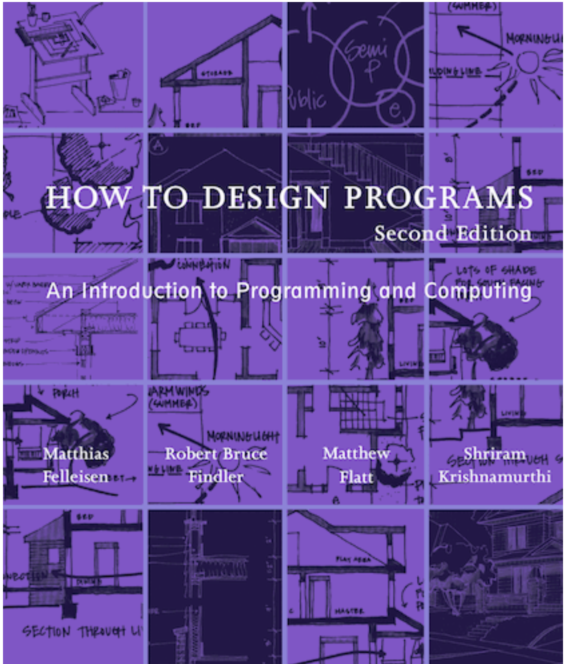
```
      (sum-of-squares 3 4)

=>    (+ (sq 3) (sq 4))
=>    (+ (* 3 3) (sq 4))
=>    (+ 9 (sq 4))
=>    (+ 9 (* 4 4))
=>    (+ 9 16)
=>    25
```

```scheme
(define (factorial n)
  (if (<= n 1)
      1
      (* n (factorial (- n 1)))))
```

```
(factorial 6)
(* 6 (factorial 5))
(* 6 (* 5 (factorial 4)))
(* 6 (* 5 (* 4 (factorial 3))))
(* 6 (* 5 (* 4 (* 3 (factorial 2)))))
(* 6 (* 5 (* 4 (* 3 (* 2 (factorial 1))))))
(* 6 (* 5 (* 4 (* 3 (* 2 1)))))
(* 6 (* 5 (* 4 (* 3 2))))
(* 6 (* 5 (* 4 6)))
(* 6 (* 5 24))
(* 6 120)
720
```

Incredible breadth of content

- recursion
- complexity analysis
- meta-linguistic abstraction
- interpreters
- object-oriented programming
- logic programming
- many other things



HOW TO DESIGN PROGRAMS

Second Edition

An Introduction to Programming and Computing

Matthias Felleisen
Robert Bruce Findler
Matthew Flatt
Shriram Krishnamurthi

[illegible]

received April 198

A critique of Abelson and Sussman
- or -
Why calculating is better than scheming

Philip Wadler
Programming Research Group
11 Keble Road
Oxford, OX1 3QD

Abelson and Sussman have written an excellent textbook which may start a revolution in the way programming is taught [Abelson and Sussman 1985a, b]. Instead of emphasizing a particular programming language, they emphasize standard engineering techniques as they apply to programming. Still, their textbook is intimately tied to the Scheme dialect of Lisp. I believe that the same approach used in their text, if applied to a language such as KRC or Miranda, would result in an even better introduction to programming as an engineering discipline. My belief has strengthened as my experience in teaching with Scheme and with KRC has increased.

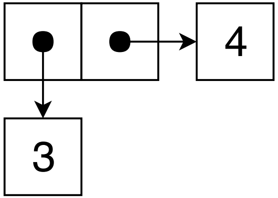
—

```scheme
(define (map proc items)
  (if (null? items)
      nil
      (cons (proc (car items))
            (map proc (cdr items)))))
```

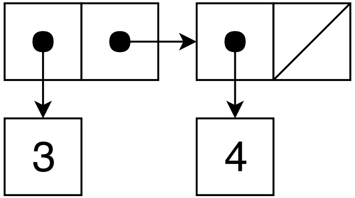
```haskell
map :: (a -> b) -> [a] -> [b]
map f l =
  case l of
    []        -> []
    (x : xs) -> f x : map f xs
```

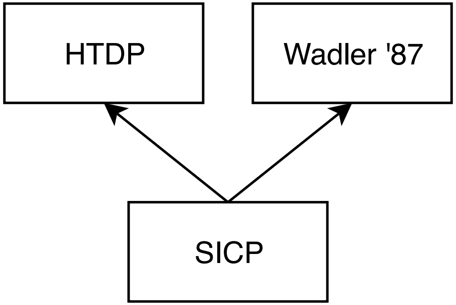
```
(list 3 4)

(cons 3 4)
```

```
(cons 3 4)
```

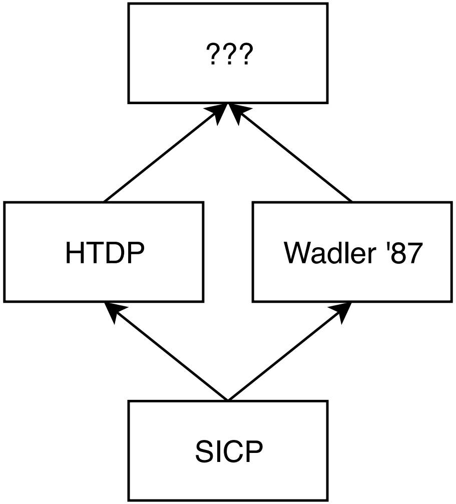


```
(list 3 4)
```




HTDP
Wadler '87
SICP

$$\begin{tikzcd}
 & ??? & \\
\text{HTDP} \arrow[ur] & & \text{Wadler '87} \arrow[ul] \\
 & \text{SICP} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$



Thanks for listening!