Summary of implementation methods

1. ISWIM $\longrightarrow$ λ-calculus $\longrightarrow$ SECD machine;
2. λ-calculus $\longrightarrow$ combinators $\longrightarrow$ graph reduction;
3. de Bruijn indexing.

ISWIM: If you See What I Mean

- The next 700 programming languages, **Landin**
- λ-calculus plus additional constructs
- ISWIM $\longrightarrow$ λ-calculus $\longrightarrow$ SECD machine
- call-by-value reduction strategy

Syntax for ISWIM

λ-terms $\quad x, \lambda x.M, MN$

Local declarations:

let $x = M$ **in** N

let $fx_1 \ldots x_k = M$ **in** N

letrec $fx_1 \ldots x_k = M$ **in** N

Simple data structures

integers, arithmetic and relational operators, booleans and connectives, conditionals

ISWIM continued

Ordered pairs

pair constructor (M, N)

projection functions **fst**, **snd**

Pattern matching

let $(x, y) = M$ **in** N

let $f(x, y) = M$ **in** N

Mutual recursion

letrec $f_1\vec{x}_1 = M_1$ **and** $f_2\vec{x}_2 = M_2 \ldots$ **and** $f_k\vec{x}_k = M_k$ **in** N

Computation steps for the SECD machine

Const

$$(S,\ E,\ k;C,\ D) \to (k;S,\ E,\ C,\ D)$$

Var

$$(S,\ E,\ x;C,\ D) \to (a;S,\ E,\ C,\ D)$$

λ

$$(S,\ E,\ \lambda x.M;C,\ D) \to (\mathbf{Clo}(x,M,E);S,\ E,\ C,\ D)$$

SECD machine

S	stack	$a ::= \text{const} \mid \mathbf{Clo}(x, M, E)$
E	environment	list of $(x = a)$
C	command	list of λ-terms or **app**'s
D	dump	(S, E, C, D) or _

$$(S,\ E,\ MN;C,\ D) \to (S,\ E,\ N;M;\mathbf{app};C,\ D)$$

$$(f;a;S,\ E,\ \mathbf{app};C,\ D) \to (f(a);S,\ E,\ C,\ D)$$

$$(\mathbf{Clo}(x,M,E');a;S,\ E,\ \mathbf{app};C,\ D) \to (\text{-},\ x=a;E',\ M,\ (S,E,C,D))$$

$$(a,\ E',\ \text{-},\ (S,E,C,D)) \to (a;S,\ E,\ C,\ D)$$