Functional Programming

- Functions

- *Lisp*'s Foundations
  - Overview
  - Data Structures (S-expression)
  - Built-in Functions + Predicates
  - Evaluation (Forms)
  - Lambda-Expressions
  - Special Forms
  - Functional Arguments + Label Lambda-Expressions

- "Extensions" to *Lisp*
  - Side Effects
  - Numbers
  - Dotted-Pair, Association & Property Lists
  - *Lisp qua* Procedural Language

- Issues
  - Recursion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (Eval, nlambda, OOp)
  - Lambda Calculus
  - SECD machine

- Example (polynomials)

Examples of Functions

- "`Age`" as in "`Age(Mary) = 15`"

  Age takes one argument (a person)

  and returns that person's age (a number).

- "`Father`" as in "`Father(Mary) = Fred`"

  Father-hood takes one argument (a person)

  and returns that person's father (a person).

- "!" as in "$5! = 120$"

  Factorial takes one argument (a positive integer)

  and returns its factorial (a positive integer).

- "$+$" as in "$2+3=5$"

  Addition takes two arguments (each a real value)

  and returns their sum (a real value).

Functions – Definition

- Def'n:
  A function f is a *mapping*
  from one set, D (the domain),
  into another set, R (the range),
  such that:

  – f is defined on all of D:

  $$\forall d \in D.\, \exists r \in R.\, f(d) = r.$$

  – $f(d)$ has only one value $\forall d \in D$:

  $$\forall d \in D.\, \forall r, s \in R.\, [\ f(d) = r \& f(d) = s\ ] \Rightarrow r = s.$$

Functions (con't)

- Examples:

Function	Domain	Range
age	Person	$\mathcal{R}^+$
Father	Person	Person
!	$\mathcal{Z}^+$	$\mathcal{Z}^+$
$+$	$\mathcal{R} \times \mathcal{R}$	$\mathcal{R}$

- Notation

Age	of	Mary	is	15
Father	of	Mary	is	Fred
!	of	5	is	120
$+$	of	10 and 9	is	19
!	of	-11	is	*undefined*
!	of	Mary	is	*undefined*

- If D is a set of n-tuples,
  then f is *a function of* n *arguments*.

  Write "$+(2,3)$" as "$2+3$"
  (for BINARY functions)

  "f is applied to an argument of D
  to give a value in R."

Composing Functions

- $f \circ g$ is *composition* of functions f and g

- Example:

  $$\underline{! \circ +}(2,3) = ![+(2,3)] = ![5] = 120$$

  $! \circ +$ is a function
  taking two real numbers, and
  returning factorial of their sum (if $\in \mathcal{Z}^+$)

- In gen'l:
  Given functions f and g,
  "$f \circ g$" is function which
  first applies g to args,
  then applies f to the result.

Theoretical Underpinnings

- What is SEMANTICs of
  Functional Programming?

  (What does a LISP program *mean*?
  ...what should it do?)

- Referentially transparent naming mechanism

- Define objects in terms of functions

- Recursion

- Interpretation

$\Rightarrow$ Lambda Calclulus (λ)

[Later]