CMPUT 325 - Functional Programming

Dr. B. Price & Dr. R. Greiner

14th September 2004

Functional Programming in *Lisp*

- The Theory of Functions

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview
  - Data Structures (The S-expression)

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview
  - Data Structures (The S-expression)
  - Built-in Functions + Predicates

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview
  - Data Structures (The S-expression)
  - Built-in Functions + Predicates
  - Evaluation (Forms)

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview
  - Data Structures (The S-expression)
  - Built-in Functions + Predicates
  - Evaluation (Forms)
  - Lamda-Expressions

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview
  - Data Structures (The S-expression)
  - Built-in Functions + Predicates
  - Evaluation (Forms)
  - Lamda-Expressions
  - Special Forms

Functional Programming in *Lisp*

- The Theory of Functions
- *Lisp*Basics
  - Overview
  - Data Structures (The S-expression)
  - Built-in Functions + Predicates
  - Evaluation (Forms)
  - Lamda-Expressions
  - Special Forms
  - Functional Arguments + Label Lambda-Expressions

Issues in Functional Programming

- Issues

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency
  - Funarg Problem (Scoping)

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (EVAL, NLAMBDA, OOP)

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (EVAL, NLAMBDA, OOP)
  - Lambda Calculus

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (EVAL, NLAMBDA, OOP)
  - Lambda Calculus
  - SECD Machine

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (EVAL, NLAMBDA, OOP)
  - Lambda Calculus
  - SECD Machine
  - Lazy evaluation and Series

Issues in Functional Programming

- Issues
  - Recusion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (EVAL, NLAMBDA, OOP)
  - Lambda Calculus
  - SECD Machine
  - Lazy evaluation and Series
- Example (polynomials)

"Non-Functional" *Lisp*

- Practical "Extensions" to *Lisp*

"Non-Functional" *Lisp*

- Practical "Extensions" to *Lisp*
  - Functions with Side effects

"Non-Functional" *Lisp*

- Practical "Extensions" to *Lisp*
  - Functions with Side effects
  - Numbers

"Non-Functional" *Lisp*

- Practical "Extensions" to *Lisp*
  - Functions with Side effects
  - Numbers
  - Dotted-Pair, Association & Property Lists

"Non-Functional" *Lisp*

- Practical "Extensions" to *Lisp*
  - Functions with Side effects
  - Numbers
  - Dotted-Pair, Association & Property Lists
  - Lisp qua Procedural Languages

Relations – Definition

- A n-ary relation relates items drawn from sets

Relations – Definition

- A n-ary relation relates items drawn from sets
- The set of performer and tune can be defined by a relation

Relations – Definition

- A n-ary relation relates items drawn from sets
- The set of performer and tune can be defined by a relation
- A relation has two parts:

Relations – Definition

- A n-ary relation relates items drawn from sets
- The set of performer and tune can be defined by a relation
- A relation has two parts:
  - The sets in the relation $X_1, X_2, \ldots, X_n$.

Relations – Definition

- A n-ary relation relates items drawn from sets
- The set of performer and tune can be defined by a relation
- A relation has two parts:
  - The sets in the relation $X_1, X_2, \ldots, X_n$.
  - A "graph" over the tuples taken from the elements which maps each tuple to true or false: $G : X_1 \times X_2 \times \cdots \times X_n \rightarrow \mathcal{B}$

Relations – Definition

- A n-ary relation relates items drawn from sets
- The set of performer and tune can be defined by a relation
- A relation has two parts:
  - The sets in the relation $X_1, X_2, \ldots, X_n$.
  - A "graph" over the tuples taken from the elements which maps each tuple to true or false: $G : X_1 \times X_2 \times \cdots \times X_n \rightarrow \mathcal{B}$
- E.g. Perhaps: G(rolling_stones,start_me_up) $\rightarrow$true

Relations – Definition

- A n-ary relation relates items drawn from sets
- The set of performer and tune can be defined by a relation
- A relation has two parts:
  - The sets in the relation $X_1, X_2, \ldots, X_n$.
  - A "graph" over the tuples taken from the elements which maps each tuple to true or false: $G : X_1 \times X_2 \times \cdots \times X_n \rightarrow \mathcal{B}$
- E.g. Perhaps: G(rolling_stones,start_me_up) $\rightarrow$true
- Note each performer has many songs, each song can have multiple performers

Functions – Definition

- Def'n: A function f is a *mapping*
  - from one set, D (the domain),
  - to another set, R (the range),
  - where f has **exactly one** value for **every** domain element

Functions – Definition

- Def'n: A function f is a *mapping*
  - from one set, D (the domain),
  - to another set, R (the range),
  - where f has **exactly one** value for **every** domain element
- Formally:
  - $f(d)$ defines **at least one** value: $\forall d \in D. \exists r \in R. f(d) = r$
  - $f(d)$ defines **at most one** value:

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

Examples of Functions

- The age of a person is a function.
  - Formally: **age-of**(Mary)=15

Examples of Functions

- The age of a person is a function.
  - Formally: **age-of**(Mary)=15
- The address of a department is a function.
  - Formally: **birth-mother**(russ)=claire

Examples of Functions

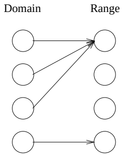
- The age of a person is a function.
  - Formally: **age-of**(Mary)=15
- The address of a department is a function.
  - Formally: **birth-mother**(russ)=claire
- The square of a number is a function.
  - Formally: $3^2 = 9$

Example Functions as Maps

Function	Domain Set	Range Set
age-of	all persons	positive reals
birth-mother	all people	people
square	numbers	numbers

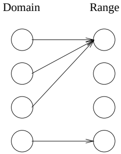
Functions and Non-functions

Are these examples functions?

- 

Functions and Non-functions

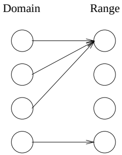
Are these examples functions?

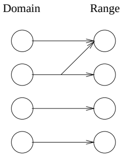


- Yes. Every d has an r (e.g. age-of(d))

Functions and Non-functions

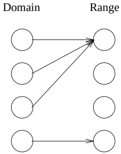
Are these examples functions?

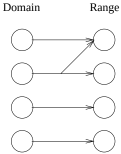
- 

Yes. Every d has an r (e.g. age-of(d))
- 

Functions and Non-functions

Are these examples functions?

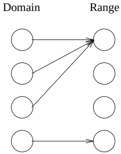
- 

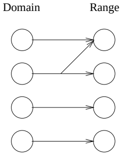
Yes. Every d has an r (e.g. `age-of(`d`)`)
- 

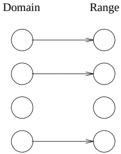
No. Exists d with multiple r (e.g. `parent-of(d)`)

Functions and Non-functions

Are these examples functions?

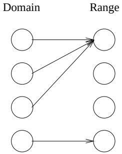
- 

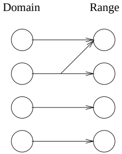
Yes. Every d has an r (e.g. `age-of(`d`)`)
- 

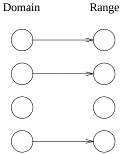
No. Exists d with multiple r (e.g. `parent-of`(d))
- 

Functions and Non-functions

Are these examples functions?

- 

Yes. Every d has an r (e.g. `age-of(`d`)`)
- 

No. Exists d with multiple r (e.g. `parent-of(d)`)
- 

No. Exists d with no r (e.g. `sqrt(d)`)

N-ary Functions

- N-ary domain is the cross product of unary domains

N-ary Functions

- N-ary domain is the cross product of unary domains
  - The domain of positive integers is: $\mathcal{Z}^+ = \{0, 1, 2, 3, \ldots\}$

N-ary Functions

- N-ary domain is the cross product of unary domains
  - The domain of positive integers is: $\mathcal{Z}^+ = \{0, 1, 2, 3, \ldots\}$
  - The domain of integer pairs is:

$$\mathcal{Z}^2 = \mathcal{Z} \times \mathcal{Z} = \begin{bmatrix} (0,0) & (0,1) & \ldots \\ (1,0) & (1,1) & \\ \vdots & & \ddots \end{bmatrix}$$

N-ary Functions

- N-ary domain is the cross product of unary domains
  - The domain of positive integers is: $\mathcal{Z}^+ = \{0, 1, 2, 3, \ldots\}$
  - The domain of integer pairs is:

$$\mathcal{Z}^2 = \mathcal{Z} \times \mathcal{Z} = \begin{bmatrix} (0,0) & (0,1) & \ldots \\ (1,0) & (1,1) & \\ \vdots & & \ddots \end{bmatrix}$$

  - Each element of $\mathcal{Z}^2$ is a binary tuple

N-ary Functions

- N-ary domain is the cross product of unary domains
  - The domain of positive integers is: $\mathcal{Z}^+ = \{0, 1, 2, 3, \ldots\}$
  - The domain of integer pairs is:
    $$\mathcal{Z}^2 = \mathcal{Z} \times \mathcal{Z} = \begin{bmatrix} (0,0) & (0,1) & \ldots \\ (1,0) & (1,1) & \\ \vdots & & \ddots \end{bmatrix}$$
  - Each element of $\mathcal{Z}^2$ is a binary tuple
- Sum is a binary function mapping tuples $(z_1, z_2) \in \mathcal{Z}^2$ to elements in $\mathcal{Z}$
  $$+ : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathcal{Z}$$

Vector Functions

- **range** is the cross product of unary domains

Vector Functions

- **range** is the cross product of unary domains
  - The range of real pairs is:

$$\mathcal{R}^2 = \mathcal{R} \times \mathcal{R} = \begin{bmatrix} (-\infty, -\infty) & & \dots \\ & \ddots & \\ \vdots & & (+\infty, +\infty) \end{bmatrix}$$

Vector Functions

- **range** is the cross product of unary domains
  - The range of real pairs is:

$$\mathcal{R}^2 = \mathcal{R} \times \mathcal{R} = \begin{bmatrix} (-\infty, -\infty) & & \cdots \\ & \ddots & \\ \vdots & & (+\infty, +\infty) \end{bmatrix}$$

  - Each element of $\mathcal{R}^2$ is a binary tuple (r_1, r_2) where $r_1, r_2 \in \mathcal{R}$

Vector Functions

- **range** is the cross product of unary domains
  - The range of real pairs is:

$$\mathcal{R}^2 = \mathcal{R} \times \mathcal{R} = \begin{bmatrix} (-\infty, -\infty) & & \dots \\ & \ddots & \\ \vdots & & (+\infty, +\infty) \end{bmatrix}$$

  - Each element of $\mathcal{R}^2$ is a binary tuple (r_1, r_2) where $r_1, r_2 \in \mathcal{R}$
- The rectangular-to-polar-coordinates function, $\mathrm{aspolar}$, maps $\mathcal{R}^2$ to radius-angle space $\mathcal{R} \times \Theta$

$$\mathrm{aspolar} : \mathcal{R}^2 \rightarrow \mathcal{R} \times \Theta$$

Vector Functions

- **range** is the cross product of unary domains
  - The range of real pairs is:

$$\mathcal{R}^2 = \mathcal{R} \times \mathcal{R} = \begin{bmatrix} (-\infty, -\infty) & & \dots \\ & \ddots & \\ \vdots & & (+\infty, +\infty) \end{bmatrix}$$

  - Each element of $\mathcal{R}^2$ is a binary tuple (r_1, r_2) where $r_1, r_2 \in \mathcal{R}$
- The rectangular-to-polar-coordinates function, $\mathrm{aspolar}$, maps $\mathcal{R}^2$ to radius-angle space $\mathcal{R} \times \Theta$

$$\mathrm{aspolar} : \mathcal{R}^2 \rightarrow \mathcal{R} \times \Theta$$

- **NOTE: for each element of the domain a vector function returns a exactly one tuple**

Function Functions

- N-ary domain of objects

Function Functions

- N-ary domain of objects
- Range is the space of functions

Function Functions

- N-ary domain of objects
- Range is the space of functions
- Example: Machine Learning Neural Network
  - Domain: labeled set of examples and learning algorithm
  - Range: a function f that can be used to predict labels of unseen data
  - Mapping: $\text{learner} : \mathcal{D} \rightarrow \mathcal{R}$

Function Functions

- N-ary domain of objects
- Range is the space of functions
- Example: Machine Learning Neural Network
  - Domain: labeled set of examples and learning algorithm
  - Range: a function f that can be used to predict labels of unseen data
  - Mapping: $\text{learner} : \mathcal{D} \rightarrow \mathcal{R}$
- Sample Data point: ((5,1) (-2, -1) (3,1) (11,1) (-9, -1) (-20,-1))

Function Functions

- N-ary domain of objects
- Range is the space of functions
- Example: Machine Learning Neural Network
  - Domain: labeled set of examples and learning algorithm
  - Range: a function f that can be used to predict labels of unseen data
  - Mapping: $\text{learner} : \mathcal{D} \rightarrow \mathcal{R}$
- Sample Data point: ((5,1) (-2, -1) (3,1) (11,1) (-9, -1) (-20,-1))
- Sample Range value: f(x) = if x>0 return 1 else return -1

Function Application Notation

- We say that "f is applied to an argument of D to give a value in R."

Function Application Notation

- We say that "f is applied to an argument of D to give a value in R."
- Ways of indicating function application:

Function Application Notation

- We say that "f is applied to an argument of D to give a value in R."
- Ways of indicating function application:
  - Infix notation: function name between arguments
    General form: $a_1\, fn\, a_2$ (e.g. $5 + 7$)

Function Application Notation

- We say that "f is applied to an argument of D to give a value in R."
- Ways of indicating function application:
  - Infix notation: function name between arguments
    General form: $a_1\, fn\, a_2$ (e.g. $5 + 7$)
  - Prefix notation: function name before arguments
    General form: $fn(a_1, a_2, \ldots)$
    (e.g.$+(5, 7)$, `distance-between( ottawa, edmonton)`)

Function Application Notation

- We say that "f is applied to an argument of D to give a value in R."
- Ways of indicating function application:
  - Infix notation: function name between arguments
    General form: $a_1\, fn\, a_2$ (e.g. $5 + 7$)
  - Prefix notation: function name before arguments
    General form: $fn(a_1, a_2, \ldots)$
    (e.g.$+(5, 7)$, `distance-between( ottawa, edmonton)`)
  - Postfix notation: function name follows arguments
    General form: $a_1, a_2, \ldots\ fn$ (e.g. $(5, 7)+$"Reverse Polish notation")

Equivalence of Notations

- Notations are equivalent, though there are preferred conventions:

Equivalence of Notations

- Notations are equivalent, though there are preferred conventions:
  - $+(1,2) \equiv (1+2) \equiv (1,2)+$

Equivalence of Notations

- Notations are equivalent, though there are preferred conventions:
  - $+(1,2) \equiv (1+2) \equiv (1,2)+$
  - grade-of(first-name,lastname)
    $\equiv$first-name grade-of last-name
    $\equiv$first-name last-name grade-of

Image and Preimage

- Def'n: image of $d \in D$ under function f is the result $r \in R$

Image and Preimage

- Def'n: image of $d \in D$ under function f is the result $r \in R$
- Def'n: preimage or inverse of $r \in R$ under function f is the element(s) of $d \in D$ that result in r

Composing Functions

- Def'n: The application of a function to result of another function

Composing Functions

- Def'n: The application of a function to result of another function
- Notation: $f \circ g$, means f applied to result of g

Composing Functions

- Def'n: The application of a function to result of another function
- Notation: $f \circ g$, means f applied to result of g
- Note: Domain of outer function must accept result of inner function
  $\mathrm{domain(f)} \supseteq \mathrm{range(g)}$

Composing Functions

- Def'n: The application of a function to result of another function
- Notation: $f \circ g$, means f applied to result of g
- Note: Domain of outer function must accept result of inner function
  $\mathrm{domain(f)} \supseteq \mathrm{range(g)}$
- Given the factorial function ! and the sum function $+$, their composition is: $! \circ +$

Composing Functions

- Def'n: The application of a function to result of another function
- Notation: $f \circ g$, means f applied to result of g
- Note: Domain of outer function must accept result of inner function
  $\mathrm{domain(f)} \supseteq \mathrm{range(g)}$
- Given the factorial function ! and the sum function +, their composition is: $! \circ +$
- Example: $! \circ +(2,3) \ = ![+(2,3)] \ = ![5] \ = \ 120$ $! \circ +$ is a function
  taking two real numbers, and
  returning factorial if their sum $\in \mathcal{Z}^+$