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
- *LispBasics*
  - Overview
Functional Programming in *Lisp*

- The Theory of Functions

- *LispBasics*
  - Overview
  - Data Structures (The S-expression)
Functional Programming in *Lisp*

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

- The Theory of Functions
- *LispBasics*
  - 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

- *LispBasics*
  - Overview
  - Data Structures (The S-expression)
  - Built-in Functions + Predicates
  - Evaluation (Forms)
  - Lambda-Expressions
  - Special Forms
Functional Programming in *Lisp*

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

- Issues
Issues in Functional Programming

- Issues
  - Recursion, Variables, Efficiency
Issues in Functional Programming

- Issues
  - Recursion, Variables, Efficiency
  - Funarg Problem (Scoping)
Issues in Functional Programming

- Recursion, Variables, Efficiency
- Funarg Problem (Scoping)
- Program=Data (EVAL, NLAMBD, OOP)
Issues in Functional Programming

- Recursion, Variables, Efficiency
- Funarg Problem (Scoping)
- Program=Data (EVAL, NLAMBDAA, OOP)
- Lambda Calculus
Issues in Functional Programming

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

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

- Issues
  - Recursion, Variables, Efficiency
  - Funarg Problem (Scoping)
  - Program=Data (EVAL, NLAMBD'A, 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
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 $X_1, X_2, \ldots, X_n$. 

$G: X_1 \times X_2 \times \cdots \times X_n \rightarrow B$

E.g. Perhaps: $G(\text{rolling_stones, start_me_up}) \rightarrow \text{true}$

Note each performer has many songs, each song can have multiple performers
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: $G(\text{rolling\_stones}, \text{start\_me\_up}) \rightarrow \text{true}$
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: $G(\text{rolling_stones, start_me_up}) \rightarrow \text{true}$
- 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:

    $\forall d \in D. \forall r, s \in R. [f(d) = r \land f(d) = s] \Rightarrow r = s$
Examples of Functions

- The age of a person is a function.
  - Formally: \texttt{age-of}(Mary)=15

- The address of a department is a function.
  - Formally: \texttt{birth-mother}(russ)=claire

- The square of a number is a function.
  - Formally: \(3^2=9\)
Examples of Functions

- The age of a person is a function.
  - Formally: \texttt{age-of}(Mary)=15

- The address of a department is a function.
  - Formally: \texttt{birth-mother}(russ)=claire
Examples of Functions

- The age of a person is a function.
  - Formally: \( \text{age-of}(\text{Mary})=15 \)

- The address of a department is a function.
  - Formally: \( \text{birth-mother}(\text{russ})=\text{claire} \)

- The square of a number is a function.
  - Formally: \( 3^2 = 9 \)
## Example Functions as Maps

<table>
<thead>
<tr>
<th>Function</th>
<th>Domain Set</th>
<th>Range Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>age-of</td>
<td>all persons</td>
<td>positive reals</td>
</tr>
<tr>
<td>birth-mother</td>
<td>all people</td>
<td>people</td>
</tr>
<tr>
<td>square</td>
<td>numbers</td>
<td>numbers</td>
</tr>
</tbody>
</table>
Functions and Non-functions

Are these examples functions?

<table>
<thead>
<tr>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Yes. Every domain has a range (e.g., age-of(d)).

No. Exists domain with multiple ranges (e.g., parent-of(d)).

No. Exists domain with no range (e.g., sqrt(d)).
Functions and Non-functions

Are these examples functions?

Yes. Every $d$ has an $r$ (e.g. $\text{age-of}(d)$)
Functions and Non-functions

Are these examples functions?

- Yes. Every $d$ has an $r$ (e.g. $\text{age-of}(d)$)

- No. Exists $d$ with multiple $r$ (e.g. $\text{parent-of}(d)$)

- No. Exists $d$ with no $r$ (e.g. $\sqrt{d}$)
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. $\text{age-of}(d)$)

- No. Exists $d$ with multiple $r$ (e.g. $\text{parent-of}(d)$)

- No. Exists $d$ with no $r$ (e.g. $\text{sqrt}(d)$)
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: \( \mathbb{Z}^+ = \{0, 1, 2, 3, \ldots\} \)
N-ary Functions

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

\[
\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z} = \begin{bmatrix}
(0,0) & (0,1) & \ldots \\
(1,0) & (1,1) \\
\vdots & \ddots
\end{bmatrix}
\]
N-ary Functions

- N-ary domain is the cross product of unary domains
  - The domain of positive integers is: \( \mathbb{Z}^+ = \{0, 1, 2, 3, \ldots\} \)
  - The domain of integer pairs is:
    \[
    \mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z} = \begin{bmatrix}
    (0, 0) & (0, 1) & \ldots \\
    (1, 0) & (1, 1) & \\
    \vdots & \vdots & \\
    \end{bmatrix}
    \]
  - Each element of \( \mathbb{Z}^2 \) is a binary tuple
N-ary Functions

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

- **range** is the cross product of unary domains
  - The range of real pairs is:
    $$\mathcal{R}^2 = \mathcal{R} \times \mathcal{R} = \begin{bmatrix} (-\infty, -\infty) & \ldots \\ \vdots & \ddots \\ \vdots & (\infty, \infty) \end{bmatrix}$$
  - 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:
    \[
    \mathcal{R}^2 = \mathcal{R} \times \mathcal{R} = \begin{bmatrix}
    (-\infty, -\infty) & \cdots \\
    \vdots & \\
    \vdots & \\
    \vdots & \\
    \end{bmatrix}
    \]
  - 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, \texttt{aspolar}, maps \( \mathcal{R}^2 \) to radius-angle space \( \mathcal{R} \times \Theta \)

\[
\text{aspolar} : \mathcal{R}^2 \to \mathcal{R} \times \Theta
\]
Vector Functions

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

  - The range of real pairs is:

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

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

- The rectangular-to-polar-coordinates function, `as_polar`, maps $\mathbb{R}^2$ to radius-angle space $\mathbb{R} \times \Theta$

  $$\text{as}_polar : \mathbb{R}^2 \rightarrow \mathbb{R} \times \Theta$$

- **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: learner : $D \rightarrow R$
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: learner : $D \rightarrow R$
- 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: learner : $D \rightarrow R$
- Sample Data point: ( (5,1) (-2, -1) (3,1) (11,1) (-9, -1) (-20,-1) )
- Sample Range value: $f(x) = \text{if } x>0 \text{ return 1 else return -1}$
We say that “\( f \) is applied to an argument of \( D \) to give a value in \( R \).”
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. \( 5 + 7 \))

  - Prex notation: function name before arguments
    General form: \( fn (a_1, a_2, \ldots) \) (e.g. \(+((5, 7), \text{distance-between(ottawa, edmonton)})\))

  - Postx notation: function name follows arguments
    General form: \( a_1, a_2, \ldots, fn \) (e.g. \((5, 7)+\) “Reverse Polish notation”)
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. $+(5, 7)$, $\text{distance-between( ottawa, edmonton)}$)
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: \( fn(a_1, a_2, ...) \)
      (e.g. \(+ (5, 7), \text{distance-between}(\text{ottawa, edmonton})\) )
  - Postfix notation: function name follows arguments
    General form: \( a_1, a_2, ... \ fn \) (e.g. \( (5, 7) + "\text{Reverse Polish notation}" \))
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:

- \((1, 2) \equiv (1 + 2) \equiv (1, 2) +\)
- \(\text{grade-of(first-name,lastname)} \equiv \text{first-name grade-of last-name} \equiv \text{first-name last-name grade-of}\)
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
  
  \[ \text{domain}(f) \supseteq \text{range}(g) \]
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
  \[ \text{domain}(f) \supseteq \text{range}(g) \]

- 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
  
  $\text{domain}(f) \supseteq \text{range}(g)$

- 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 $\in \mathbb{Z}^+$