Functional Programming and Scheme

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016

Functional Language

$$\text{Output} = f\ (\text{Input})$$

Functional: program is a mathematical function

History

What is computation?

- Combinatory logic (1920’s)
- ***Lambda calculus (1930’s)***
- Turing machine (1930’s)
- …

First electronic general-purpose computer

- ENIAC (1946)

Mathematical Function

A mathematical function is a mapping of members of one set, called the **domain set**, to another set, called the **range set (image)**.

Notations:

Domain Range

$$+ : (\mathbb{Z}, \mathbb{Z}) \longmapsto \mathbb{Z}$$

$$\text{isNegtive} : \mathbb{Z} \longmapsto \{\text{True}, \text{False}\}$$

Infix form

Function application

$+\ 1\ 2 = 3$ (same as 1+2=3)

$$\text{isNegtive}\ 1 = \ \text{False}$$

The λ-Calculus

Alonzo Church, 1930s

A pure λ-term is defined inductively as follows:

- Any variable x is a λ-term
- If t is a λ-term, so is $\lambda x.\, t$ (abstraction)
- If t_1, t_2 are λ-terms, so is $t_1\, t_2$ (application)

Analogy in C:
Abstraction: `int f (int x) {return x+1}`
Application : `f(2)`

λ-Term Examples

Identity function: $\lambda x.\, x$

Application: $(\lambda x.\, x)\ y$ [apply identity function to parameter y]

How to parse a λ-term

1. λ binding extends to the rightmost part

   $\lambda x.\, x\ \lambda y.\, y\ z$ is parsed as $\lambda x.\, (x\ (\lambda y.\, (y\ z)))$

2. Applications are left-associative

   $t_1\ t_2\ t_3$ is parsed as $(t_1\ t_2)\ t_3$

Number of Parameters

In the pure λ-calculus, λ only bind one parameter

For convince, we write

$\lambda x\ y.\, t$ as a ***shorthand*** for $\lambda x.\, (\lambda y.\, t)$

This process of removing parameters is called ***currying***, which we will cover later in this course.

Bound vs. Free Variables

In $(\lambda x. t)$, the variable x is ***bound*** in t

Otherwise, a variable is ***free***

A variable is bound to the closest λ

Example

$\lambda x. \lambda x. x$ is a function that takes a parameter, and returns the identity function (i.e., the inner-most x is bound to the second λ)

More Formally …

In $(\lambda x.\, t)$, the variable x is ***bound*** in t

Otherwise, a variable is ***free***

Systematically, we define free variables as follows:

- $\mathrm{FV}(x) = \{x\}$
- $\mathrm{FV}\,(t_1\; t_2) = \mathrm{FV}(t_1) \cup \mathrm{FV}(t_2)$
- $\mathrm{FV}\,(\lambda x.\, t) = \mathrm{FV}\,(t) - \{x\}$

Let Var(t) be all variables used in t, the bound variables in t (written $\mathrm{BD}(t)$) are

$$\mathrm{BD}(t) = \mathrm{Var}(t) - \mathrm{FV}(t)$$

Free Variables: Example

$$
\begin{aligned}
\mathsf{FV}(\lambda x.\,(x\;x)) &= \mathrm{FV}(x\;x) - \{x\} \\
&= \big(\mathrm{FV}(x) \cup \mathrm{FV}(x)\big) - \{x\} \\
&= (\{x\} \cup \{x\}) - \{x\} \\
&= \{\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{FV}(\lambda y.\,(x\;y)) &= \mathrm{FV}(x\;y) - \{y\} \\
&= \big(\mathrm{FV}(x) \cup \mathrm{FV}(y)\big) - \{y\} \\
&= (\{x\} \cup \{y\}) - \{y\} \\
&= \{x\}
\end{aligned}
$$

λ-Calculus Evaluation (Informal)

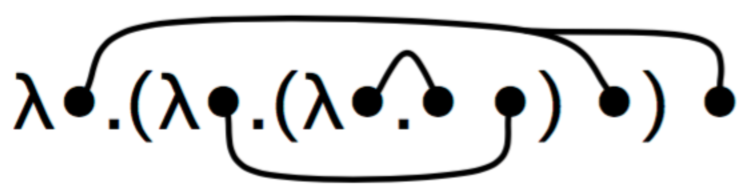
Identity function: $\lambda x.\, x$

is the same as $\lambda y.\, y$ and $\lambda z.\, z$ etc.

Observation: the name of a parameter is irrelevant in λ-calculus

Stoy diagrams

$$\lambda x.\, (\lambda y.\, (\lambda x.\, x\ y)\ x)\ x)$$



α-Reduction

Replacing all bound variables gives the same term

$(\lambda x.\, x) = (\lambda y.\, y)$

$(\lambda x.\, x\ x) = (\lambda y.\, y\ y)$

$(\lambda x.\, x\ x) \neq (\lambda y.\, x\ y)$

$x \neq y$

More formally:

$\lambda x.\, t = \lambda y.\, t\{y/x\}$ when $y \notin \mathrm{FV}(t)$

where $t\{y/x\}$ means substitute ***all*** x in t with y