Calculus of Lambda Terms

Church's Definition of Computable Functions

A Bit of History

- **Alonzo Church** proposed **λ-calculus** in 1930s as a part of his work in foundations of mathematics and mathematical logic.
- **Church** and his students, **Stephen C. Kleene** and **J. B. Rossers** studied the calculus and its problems in 1940s.
- **Peter Landin** in 1960s observed that **λ-calculus** may be viewed as the **core language** of many programming languages.

A Bit of History

- The semantics of **λ-calculus** was studied in 1960s and 1970s by **Dana Scott** and others.
- The study of **Lambda** and other calculi inspired by it[a], is still active areas of research in **programming language theory**.

[a] **π-calculus** of Rabin Milner and others for concurrent message passing languages; **Martin Abadi** and **Luca Cardelli's** calculus for object oriented languages.

A Bit of History

- What does it mean when we write $\mathbf{5x^2y + 9}$?
- It may be viewed as a **value of a function** for some unspecified argument.
- The other view is, that it gives the **dynamics of computation** of the function, provided we know how to **add**, **mutiply** and evaluate the **exponent**.

A Bit of History

- In a mordern programming language we write,

  5*x*x*y + 9

  for an expression and

- this expression as a function is written as,

  int calc(int x, int y) {return 5*x*x*y + 9;}

  This is called **function abstraction**.

A Bit of History

- There was no programing language in **1930s** and
- **Alonzo Church** had to invent his own notation for **function abstraction** and **function application** to its argument.

Denumerable Set of Variable Names

A **denumerable** set of **variable names** can be defined inductively as follows. These variables are called object variables. The set of alphabet is $\{x, 0\}$.

- **Basis:** x is a **variable**.
- **Induction:** If v is a **variable**, then so is $v0$.
- **Smallest Set:** Nothing else is a **variable**.

The set is $\mathbf{V} = \{\mathbf{x}, \mathbf{x0}, \mathbf{x00}, \mathbf{x000}, \cdots\}$. For brevity we shall call them as $\mathbf{V} = \{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \cdots\}$.

We shall also use $\mathbf{a}, \mathbf{b}, \mathbf{c}, \cdots, \mathbf{u}, \mathbf{v}, \mathbf{u_i}, \mathbf{v^j}$ as meta-variables, variables ranging over the variable names.

Inductive Definition of Pure λ-terms

Let $\mathbf{V}$ be a denumerable set of variables.

- Each $\mathbf{v} \in \mathbf{V}$ is a **λ-term**.
- If **u** and **v** are **λ-terms**, and **x** is a variable, then
  - $(\mathbf{uv})$, function application, and
  - $(\lambda\mathbf{x}.\mathbf{u})$, function abstraction are **λ-terms**.
- Nothing else is a **λ-term**.

Let us call the collection of **pure λ-terms** as $\mathbf{\Lambda}$. Impure **λ-terms** may have some predefined constants.

Definition of Λ by Inference Rules

- **Axiom:** $\dfrac{}{\mathbf{x} \in \mathbf{\Lambda}}$, for all $\mathbf{x} \in \mathbf{V}$, the set of variables.
- **Rule$_1$:** $\dfrac{\mathbf{u} \in \mathbf{\Lambda} \quad \mathbf{v} \in \mathbf{\Lambda}}{(\mathbf{uv}) \in \mathbf{\Lambda}}$
- **Rule$_2$:** $\dfrac{\mathbf{u} \in \mathbf{\Lambda} \quad \mathbf{x} \in \mathbf{V}}{(\lambda \mathbf{x}.\mathbf{v}) \in \mathbf{\Lambda}}$

An Alternate Definition of Λ

For each $i \in \mathbb{N}$ we define

$$\begin{aligned} \mathbf{\Lambda_0} &= \mathbf{V}, \text{ the set of variables,} \\ \mathbf{\Lambda_i} &= \mathbf{V} \cup \{(\mathbf{u}, \mathbf{v}) : \mathbf{u}, \mathbf{v} \in \mathbf{\Lambda_{i-1}}\} \cup \\ & \quad \{(\lambda \mathbf{x}.\mathbf{u} : \mathbf{u} \in \mathbf{\Lambda_{i-1}}, \mathbf{x} \in \mathbf{V}\}, \ \mathbf{i} > \mathbf{0}. \end{aligned}$$

The collection of terms $\Lambda = \bigcup_{i \in \mathbb{N}} \Lambda_i$.

There are other ways to define Λ.

Examples of λ-terms

Actual Term	*We Write*	*Name*
$(\lambda x.x)$	$\lambda x.x$	$\mathbf{I}$
$(\lambda x.(\lambda y.x))$	$\lambda xy.x$	$\mathbf{K}$
$((\lambda x.(xx))(\lambda x.(xx)))$	$(\lambda x.xx)(\lambda x.xx)$	$\mathbf{\Omega}$
$(\lambda x.(\lambda y.y))$	$\lambda xy.y$	$\mathbf{K}_*$
$(\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))$	$\lambda xyz.(xz)(yz)$	$\mathbf{S}$

Avoid Parenthesis

- There are too many **parenthesis** which can be avoided by introducing a convention.
- $(\cdots((\mathbf{u_0 u_1})\mathbf{u_2})\cdots\mathbf{u_k})$ is written as $\mathbf{u_0 u_1}\cdots\mathbf{u_k}$ - function application is *left associative*.
- The term $\lambda\mathbf{x_1}.(\lambda\mathbf{x_2}.(\cdots.(\lambda\mathbf{n}.\mathbf{u})\cdots))$ will be writen as $\lambda\mathbf{x_1 x_2}\cdots\mathbf{x_n}.\mathbf{u}$ - the scope of function abstraction goes as far as possible to right.

Arithmetic Expression and Expression Tree

Consider the following **arithmetic expression**
$\mathbf{2+3*(5+4+6)+4*3}$

The **order of evaluation** is as follows,

$$\overline{\overline{2+\overline{3*\overline{\overline{5+4}^{1}+6}^{2}}^{3}}^{4}+\overline{4*3}^{5}}^{6}$$

The **order of evaluation** can be shown more clearly in an **expression tree**.

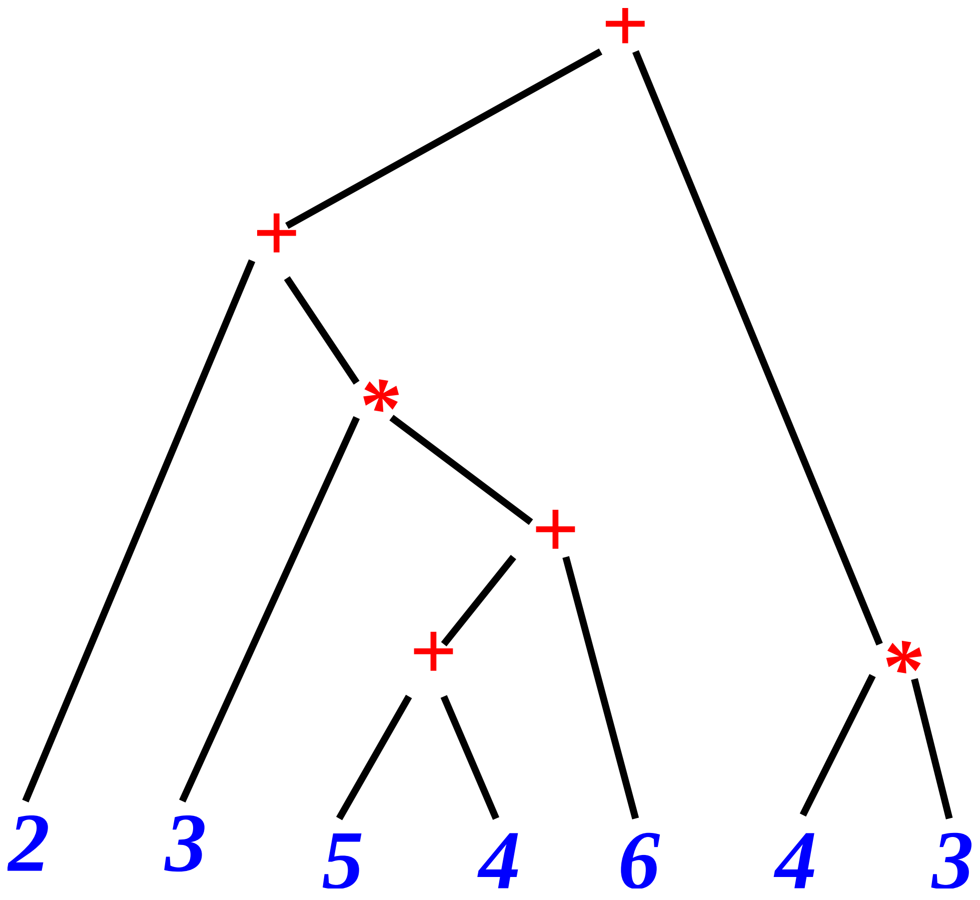

Figure 1: **Expression Tree**

Expression Tree of a λ-term

Consider the following λ-**expression**.

$$(\lambda \mathbf{x}.\lambda \mathbf{y}.\lambda \mathbf{z}.(\mathbf{xz})(\mathbf{yz}))(\lambda \mathbf{x}.\mathbf{x})$$

We shall use '**@**' for application.

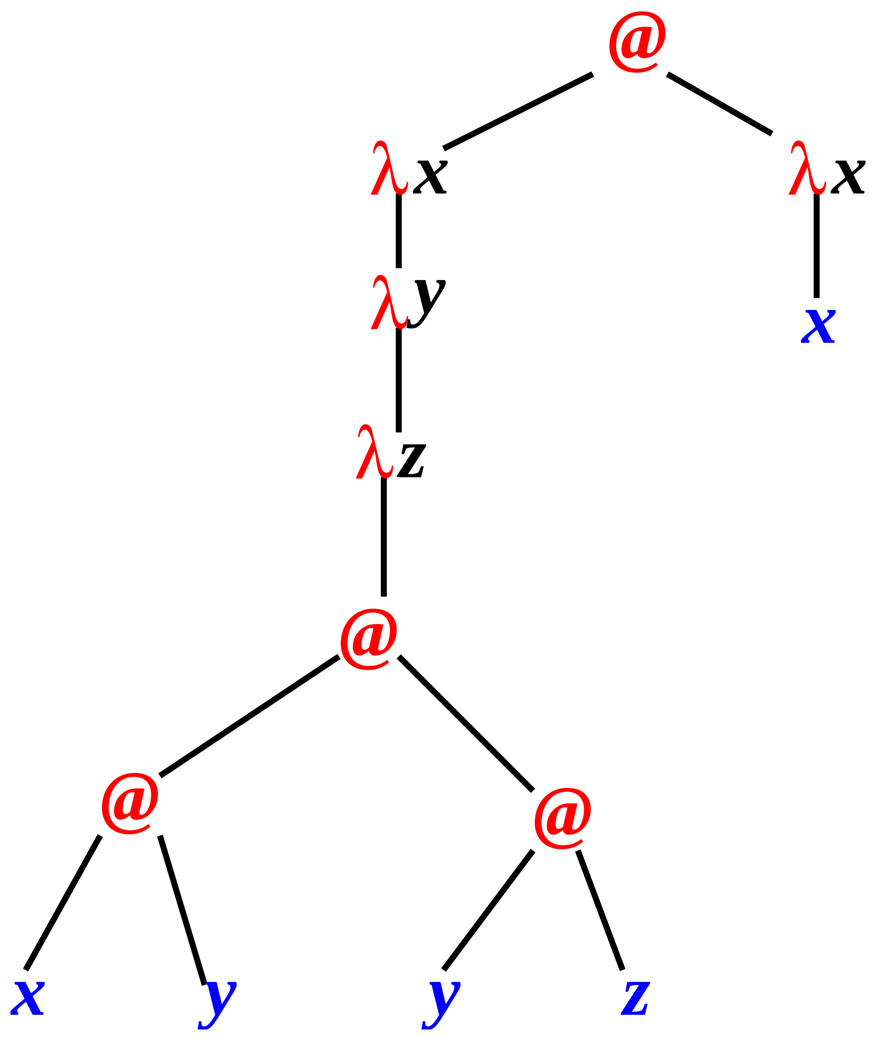

Figure 2: **λ-Expression Tree**

Free and Bound Variables

- We know $\int_0^1 \mathbf{yx^2\,dx} = \frac{\mathbf{y}}{\mathbf{3}} = \int_0^1 \mathbf{yz^2\,dz}$ - here x and z are bound variables. But **y** is a **free variable**.
- Bound variables can be **renamed** without changing the **value** of the expression. [But then a bound variable **cannot be renamed** to a free variable.]
- A bound variable is similar to the formal parameter of a function.
- In a λ-term, the '$\lambda\mathbf{x}$' binds 'x'.

Free and Bound Variables in a λ-term

Let $\mathbf{FV(u)}$ and $BV(u)$ be the set of **free** and bound variables of a λ-term u. The inductive definitions of $\mathbf{FV(u)}$ and $BV(u)$ are as follows.

- $\mathbf{FV(x)} = \{\mathbf{x}\}$ and $BV(x) = \emptyset$, if x is a variable.
- $\mathbf{FV(uv)} = \mathbf{FV(u)} \cup \mathbf{FV(v)}$,
  $BV(uv) = BV(u) \cup BV(v)$,
- $\mathbf{FV(\lambda x.u)} = \mathbf{FV(u)} \setminus \{\mathbf{x}\}$,
  $$BV(\lambda x.u) = \begin{cases} BV(u) \cup \{x\} & \text{if } x \in FV(u), \\ BV(u) & \text{if } x \notin FV(u). \end{cases}$$

Examples of Free and Bound Variables

Consider the λ-term : $\lambda xy.x(y\lambda y.xy)(\lambda x.yx(\lambda y.yx)y)$

Term	*FV*	*BV*
$\lambda y.yx$	$\{x\}$	$\{y\}$
$\lambda x.yx(\lambda y.yx)y)$	$\{y\}$	$\{x,y\}$
$\lambda xy.x(y\lambda y.xy)(\lambda x.yx(\lambda y.yx)y)$	$\{\}$	$\{x,y\}$

Substitution in a λ-term

Let **u** and **v** be λ-terms and x be a variable. We use the notation $\mathbf{u}[\mathbf{x} = \mathbf{v}]$ for **simultaneous substitution** of all **free occurrences** of x in **u** by **v**.

- **Basis:** $\mathbf{x}[\mathbf{x} = \mathbf{v}]$ is $\mathbf{v}$ and $\mathbf{y}[\mathbf{x} = \mathbf{v}]$ is $\mathbf{y}$, where $\mathbf{x}, \mathbf{y} \in \mathbf{V}$.
- **Induction$_1$:** $(\lambda\mathbf{x}.\mathbf{u})[\mathbf{x} = \mathbf{v}]$ is $\lambda\mathbf{x}.\mathbf{u}$ as $\mathbf{x}$ is not **free** in $\mathbf{u}$.
- **Induction$_2$:** $(\lambda\mathbf{y}.\mathbf{u})[\mathbf{x} = \mathbf{v}]$ is $\lambda\mathbf{y}.\mathbf{u}[\mathbf{x} = \mathbf{v}]$, provided $\mathbf{y}$ is not **free** in $\mathbf{v}$.

Substitution in a λ-term

- **Induction$_3$:** If $\mathbf{y}$ is **free** in $\mathbf{v}$ and $\mathbf{z}$ is not **free** in both $\mathbf{u}$ as well as $\mathbf{v}$, then $(\lambda\mathbf{y}.\mathbf{u})[\mathbf{x} = \mathbf{v}]$ is $\lambda\mathbf{z}.(\mathbf{u}[\mathbf{y} = \mathbf{z}])[\mathbf{x} = \mathbf{v}]$.
- **Induction$_4$:** $(\mathbf{uv})[\mathbf{x} = \mathbf{w}]$ is $((\mathbf{u}[\mathbf{x} = \mathbf{w}])(\mathbf{v}[\mathbf{x} = \mathbf{w}]))$.

Equality of Terms

Consider the collection of all expressions over $\mathbb{N}$ with operator symbols $+, \times$. Let us call them $\mathcal{E}$.

$$\mathcal{E} = \left\{ \begin{array}{l} 0, 1, 2, 3, 4, 5, \cdots \\ 0+1, 1+2, 2+3, \cdots \\ \cdots 2+4\times 5, 3\times 7+9, 8\times 2+9, \cdots \\ \ldots \end{array} \right\}$$

Equality of Terms

Some of these terms have the **same value** i.e. they are **equivalent**.

$\{\mathbf{0}, \mathbf{0}+\mathbf{0}, \mathbf{0}+\mathbf{0}\times\mathbf{0}, \cdots\}, \{\mathbf{1}, \mathbf{0}+\mathbf{1}, \mathbf{1}+\mathbf{0}, \mathbf{1}+\mathbf{5}*\mathbf{0}, \cdots\},$
$\cdots\{\mathbf{5}, \mathbf{1}+\mathbf{4}, \mathbf{1}+\mathbf{2}\times\mathbf{2}, \mathbf{2}+\mathbf{3}\times\mathbf{1}, \cdots\}, \cdots$

Equivalence Relation

A **binary relation** $\mathbf{R}$ over a set $\mathbf{A}$, is called an **equivalence relation** if it satisfies the following three conditions.

- **Reflexivity:** $(\mathbf{a}, \mathbf{a}) \in \mathbf{R}$, for all $\mathbf{a} \in \mathbf{A}$.
- **Symmetry:** If $(\mathbf{a}, \mathbf{b}) \in \mathbf{R}$, then $(\mathbf{b}, \mathbf{a}) \in \mathbf{R}$, for all $\mathbf{a}, \mathbf{b} \in \mathbf{A}$.
- **Transitivity:** If $(\mathbf{a}, \mathbf{b}), (\mathbf{b}, \mathbf{c}) \in \mathbf{R}$, then $(\mathbf{a}, \mathbf{c}) \in \mathbf{R}$, for all $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbf{A}$.

The **equality relation** is an **equivalence relation**.

Inference Rules of Equivalence Relation

Following are a few **inference rules** of **terms** in $\mathcal{E}$. We have $s, t, r, \cdots \in \mathcal{E}$.

- Usual rules of **equivalence** :

$$\frac{}{\mathbf{s} = \mathbf{s}}; \quad \frac{\mathbf{s} = \mathbf{t}}{\mathbf{t} = \mathbf{s}}; \quad \frac{\mathbf{s} = \mathbf{t} \quad \mathbf{t} = \mathbf{r}}{\mathbf{s} = \mathbf{r}};$$

  for all $\mathbf{s}, \mathbf{t}, \mathbf{r} \in \mathcal{E}$.

Inference Rules of $+$ and $\times$

Some more **inference rules** of **terms** in $\mathcal{E}$. We have $s, t, r \cdots \in \mathcal{E}$.

- $$\frac{}{\mathbf{r} + \mathbf{s} = \mathbf{s} + \mathbf{r}}; \quad \frac{}{\mathbf{r} \times (\mathbf{s} + \mathbf{t}) = (\mathbf{r} \times \mathbf{s}) + (\mathbf{r} \times \mathbf{t})};$$

- $$\frac{\mathbf{s} = \mathbf{t}}{\mathbf{r} + \mathbf{s} = \mathbf{r} + \mathbf{t}}; \quad \frac{\mathbf{s} = \mathbf{t}}{\mathbf{r} \times \mathbf{s} = \mathbf{r} \times \mathbf{t}}$$

Equality over Λ

- Usual rules of **equivalence** :

$$\frac{}{\mathbf{u} = \mathbf{u}}; \quad \frac{\mathbf{u} = \mathbf{v}}{\mathbf{v} = \mathbf{u}}; \quad \frac{\mathbf{u} = \mathbf{v} \quad \mathbf{v} = \mathbf{w}}{\mathbf{u} = \mathbf{w}};$$

-

$$\frac{\mathbf{u} = \mathbf{v}}{(\mathbf{uw}) = (\mathbf{vw})}; \quad \frac{\mathbf{u} = \mathbf{v}}{(\mathbf{wu}) = (\mathbf{wu})}; \quad \frac{\mathbf{u} = \mathbf{v}}{\lambda\mathbf{x.u} = \lambda\mathbf{x.v}}(\xi - \mathbf{rule}).$$

Equality over Λ

- **α-equivalence:** Two λ-terms u and v are equal if one is obtained from the other by **renaming the bound variables**.
- **β-equivalence:** The λ-term $(\lambda\mathbf{x}.\mathbf{u})\mathbf{v}$ is equivalent to $\mathbf{u}[\mathbf{x} = \mathbf{v}]$.

Examples of Equality

- $\lambda \mathbf{x}.\mathbf{x} = \lambda \mathbf{a}.\mathbf{a}$ - α-equivalence.
- $(\lambda \mathbf{x}.\mathbf{x})\mathbf{u} = \mathbf{x}[\mathbf{x} = \mathbf{u}] = \mathbf{u}$, for all $u \in \Lambda$ i.e. $\lambda x.x$ and its equivalent terms behave like **identity function**.

Examples of Equality

$$
\begin{aligned}
Kuv &= (\lambda xy.x)uv, \\
&= ((\lambda x.(\lambda y.x))u)v \\
&= ((\lambda y.x)[x = u]) \\
&= (\lambda y.x[x = u])v, \\
&= (\lambda y.u)v, \; y \notin FV(u), \\
&= u
\end{aligned}
$$

The **combinator** **K** selects the 1st of the two arguments. Simiarly the **combinator** $\mathbf{k}_* \equiv \lambda\mathbf{xy.y}$ selects the second argument.