Tuesday: Lambda Calculus & Combinatory Logic

2.1.1 Words, language, theory

Concept	Example
An *alphabet* Σ is a set of symbols (often finite)	$\Sigma_0 = \{a, b\}$
A *word over* Σ is a finite sequence of elements in Σ	
Σ^* consists of all words over Σ	$abba \in \Sigma_0^*,\ bc \notin \Sigma_0^*$

A *language* L *over* Σ is a subset $L \subseteq \Sigma^*$

$L \subseteq \Sigma^*$ chooses in some way *meaningful* strings called *sentences*

often such a language is given by a grammar

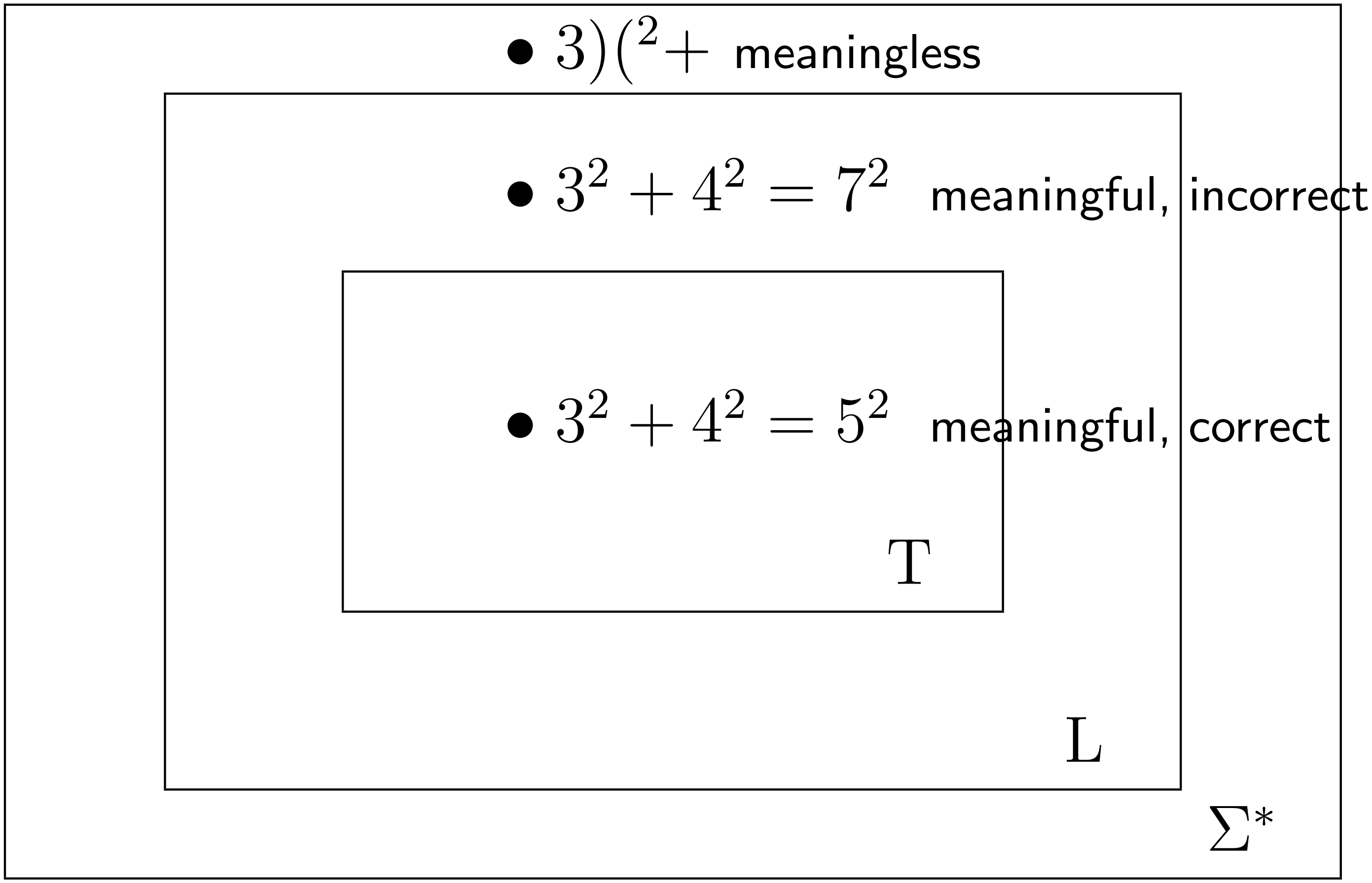
With a theory T we go one step further:

A *theory in a language* L is just a subset $T \subseteq L$

A theory selects a set of *correct* sentences

often such a theory is given by an axiomatic system

2.1.2 Words, language, theory



$$\Sigma \longmapsto \Sigma^* \longmapsto L \longmapsto T$$

$$\Sigma = \{3, 4, 5, 7, ^2, (,), +, =, \ldots\}$$

2.1.3 Combinators

$$\Sigma_{\mathbf{CL}} = \{\mathbf{I}, \mathbf{K}, \mathbf{S}, x, ',), (, =\}$$

We introduce several simple regular grammars over $\Sigma_{\mathbf{CL}}$.

(i) $\boxed{\texttt{constant} \ := \ \mathbf{I} \mid \mathbf{K} \mid \mathbf{S}}$

(ii) $\boxed{\texttt{variable} \ := \ x \mid \texttt{variable}'}$

(iii) $\boxed{\texttt{term} \ := \ \texttt{constant} \mid \texttt{variable} \mid (\texttt{term}\ \texttt{term})}$

(iv) $\boxed{\texttt{formula} \ := \ \texttt{term} = \texttt{term}}$

Intuition:

in (FA) the term F stands for a *function* and A for an *argument*

Variables $x, x', x'', \ldots$; write: $x, y, z, x_1, y_1, z_1, \ldots$

2.1.4 Combinatory Logic $\mathbf{CL}$ (Schönfinkel 1920)

Axioms

$$\begin{array}{rcll} \mathbf{I}P & = & P & \qquad (\mathbf{I}) \\ \mathbf{K}PQ & = & P & \qquad (\mathbf{K}) \\ \mathbf{S}PQR & = & PR(QR) & \qquad (\mathbf{S}) \end{array}$$

Equational deduction rules

$$\begin{array}{rcl} & & P = P \\ P = Q & \Rightarrow & Q = P \\ P = Q, Q = R & \Rightarrow & P = R \\ P = Q & \Rightarrow & PR = QR \\ P = Q & \Rightarrow & RP = RQ \end{array}$$

Here P, Q, R denote arbitrary terms

$\mathbf{I}P$ stands for $(\mathbf{I}P)$, $\mathbf{K}PQ$ for $((\mathbf{K}P)Q)$ and $\mathbf{S}PQR$ for $(((\mathbf{S}P)Q)R)$

In general $PQ_1 \ldots Q_n \;\equiv\; (..((PQ_1)Q_2) \ldots Q_n)$ (association to the left)

2.1.5 Combinatory algebras

$\mathcal{C} = \langle X, \cdot, \mathbf{K}, \mathbf{S} \rangle$

$$\boxed{\begin{array}{rcl} (\mathbf{K}x)y & = & x \\ ((\mathbf{S}x)y)z & = & (xz)(yz) \\ \mathbf{K} & \neq & \mathbf{S} \end{array}}$$

The theory $\mathbf{CL}^{\mathrm{neg}}$

Fact. The theory $\mathbf{CL}^{\mathrm{neg}}$ is

- Consistent
- Essentially incomplete
- Essentially undecidable

This means the following: $\mathbf{CL}^{\mathrm{neg}}$ does not prove every equation;

for every consistent extension T of $\mathbf{CL}^{\mathrm{neg}}$ one has

T is undecidable (there is no algorithm to determine provability)

T is incomplete (there are terms P, Q such that neither $P = Q$ nor $P \neq Q$ are in T)

2.1.6 Some magic with combinators

PROPOSITION.

(i) Let $\mathbf{D} \equiv \mathbf{SII}$. Then (doubling)

$$\mathbf{D}x =_{\mathbf{CL}} xx.$$

(ii) Let $\mathcal{B} \equiv \mathbf{S}(\mathbf{KS})\mathbf{K}$. Then (composition)

$$\mathcal{B}fgx =_{\mathbf{CL}} f(gx).$$

(iii) Let $\mathbf{L} \equiv \mathbf{D}(\mathcal{B}\mathbf{DD})$. Then (self-doubling, life!)

$$\mathbf{L} =_{\mathbf{CL}} \mathbf{LL}.$$

PROOF.

(i)
$$\begin{array}{rcl} \mathbf{D}x & \equiv & \mathbf{SII}x \\ & = & \mathbf{I}x(\mathbf{I}x) \\ & = & xx. \end{array}$$

(ii)
$$\begin{array}{rcl} \mathcal{B}fgx & \equiv & \mathbf{S}(\mathbf{KS})\mathbf{K}fgx \\ & = & \mathbf{KS}f(\mathbf{K}f)gx \\ & = & \mathbf{S}(\mathbf{K}f)gx \\ & = & \mathbf{K}fx(gx) \\ & = & f(gx). \end{array}$$

(iii)
$$\begin{array}{rcl} \mathbf{L} & \equiv & \mathbf{D}(\mathcal{B}\mathbf{DD}) \\ & = & \mathcal{B}\mathbf{DD}(\mathcal{B}\mathbf{DD}) \\ & = & \mathbf{D}(\mathbf{D}(\mathcal{B}\mathbf{DD})) \\ & \equiv & \mathbf{DL} \\ & = & \mathbf{LL}. \end{array}$$

We want to understand and preferably also to control this!

2.1.7 Lambda Calculus

The meaning of

$$\lambda x.3x$$

is the function

$$x \longmapsto 3x$$

that assigns to x the value $3x$ (3 times x)
So according to this intended meaning we have

$$(\lambda x.3x)(6) = 18.$$

The parentheses around the 6 are usually not written:

$$(\lambda x.3x)6 = 18$$

Principal axiom

$$\boxed{(\lambda x.M)N =_\beta M[x:=N]}$$

2.1.8 Language

Alphabet

$$\Sigma = \{x, ', (,), \lambda, =\}$$

Language (abstract syntax)

$$\begin{array}{rcl} \texttt{variable} & := & \texttt{x} \mid \texttt{variable}' \\ \texttt{term} & := & \texttt{variable} \mid \texttt{term}\ \texttt{term} \mid \lambda\,\texttt{variable}\ \texttt{term} \\ \texttt{formula} & := & \texttt{term} = \texttt{term} \end{array}$$

Theory

Axiom	$(\lambda x\, M)N$	$=$	$M[x := N]$
Rules			$M = M$
	$M = N$	$\Rightarrow$	$N = M$
	$M = N, N = L$	$\Rightarrow$	$M = N$
	$M = N$	$\Rightarrow$	$ML = NL$
	$M = N$	$\Rightarrow$	$LM = LN$
	$M = N$	$\Rightarrow$	$\lambda x\, M = \lambda x\, N$

2.1.9 Bureaucracy

Substitution

M	$M[x:=N]$
x	N
y	y
PQ	$(P[x:=N])(Q[x:=N])$
$\lambda x\, P$	$\lambda x\, P$
$\lambda y\, P$	$\lambda y\,(P[x:=N])$, where $y \not\equiv x$

'Association to the left'

$$PQ_1 \ldots Q_n \equiv (..((PQ_1)Q_2) \ldots Q_n).$$

'Association to the right'

$$\lambda x_1 \ldots x_n.M \equiv (\lambda x_1(\lambda x_2(..(\lambda x_n(M))..)))).$$

Outer parentheses are often omitted. For example

$$(\lambda x.x)y \equiv ((\lambda xx)y)$$

2.1.10 Examples

$$\begin{array}{rclcrcl}
\mathsf{I} & \equiv & \lambda x.x & \Rightarrow & \mathsf{I}X & =_\beta & X \\
\mathsf{K} & \equiv & \lambda xy.x & \Rightarrow & \mathsf{K}XY & =_\beta & X \\
\mathsf{S} & \equiv & \lambda xyz.xz(yz) & \Rightarrow & \mathsf{S}XYZ & =_\beta & XZ(YZ) \\
\mathsf{D} & \equiv & \lambda x.xx & \Rightarrow & \mathsf{D}X & =_\beta & XX \\
\mathsf{B} & \equiv & \lambda xyz.x(yz) & \Rightarrow & \mathsf{B}XYZ & =_\beta & X(YZ)
\end{array}$$

Set of lambda terms: Λ

Free variables of a term

$$\boxed{\begin{array}{rcl}
\mathrm{FV}(x) & = & \{x\} \\
\mathrm{FV}(PQ) & = & \mathrm{FV}(P) \cup \mathrm{FV}(Q) \\
\mathrm{FV}(\lambda x.P) & = & \mathrm{FV}(P) - \{x\}
\end{array}}$$

$\Lambda^\emptyset = \{M \in \Lambda \mid \mathrm{FV}(M) = \emptyset\}$ the set of *closed terms* or *combinators*

2.1.11 Fixed point theorem

THEOREM. For all $F \in \Lambda$ there is an $M \in \Lambda$ such that

$$FM =_\beta M$$

PROOF. Defines $\mathsf{W} \equiv \lambda x.F(xx)$ and $M \equiv \mathsf{WW}$. Then

$$\begin{aligned} M &\equiv \mathsf{WW} \\ &\equiv (\lambda x.F(xx))\mathsf{W} \\ &= F(\mathsf{WW}) \\ &\equiv FM. \blacksquare \end{aligned}$$

COROLLARY. For any 'context' $C[\vec{x}, m]$ there exists a M such that

$$M\vec{X} = C[\vec{X}, M].$$

PROOF. M can be taken the fixed point of $\lambda m\vec{x}.C[\vec{x}, m]$.
Then $M\vec{X} = (\lambda m\vec{x}.C[\vec{x}, m])M\vec{X} = C[\vec{X}, M]$. $\blacksquare$

2.1.12 Consequences

We can construct terms $\mathsf{Y}, \mathsf{L}, \mathsf{O}, \mathsf{P}$ such that

$$\begin{array}{rcll} \mathsf{Y}f & = & f(\mathsf{Y}f) & \text{producing fixed points;} \\ \mathsf{L} & = & \mathsf{L}\mathsf{L} & \text{take } \mathsf{L} \equiv \mathsf{Y}\,\mathsf{D}; \\ \mathsf{O}x & = & \mathsf{O} & \text{take } \mathsf{O} \equiv \mathsf{Y}\,\mathsf{K}; \\ \mathsf{P} & = & \mathsf{P}x. & \end{array}$$

Defne for $n \in \mathsf{Nat}$ the Church numerals:

$$\mathbf{c}_n := \lambda f x. f^n x,$$

where $f^0 x := x, \;\; f^{n+1} x := f(f^n x)$

Note that for $A_+ := \lambda nmfx.nf(mfx)$ one has $A_+\mathbf{c}_n\mathbf{c}_m =_\beta \mathbf{c}_{n+m}$.

Similarly for $A_\times := \lambda nmfx.n(mf)x$ one has $A_\times\mathbf{c}_n\mathbf{c}_m =_\beta \mathbf{c}_{n\times m}$.

2.1.13 More Bureaucracy

$\lambda x.x$ and $\lambda y.y$ acting on M both give M

We write

$$\lambda x.x \equiv_\alpha \lambda y.y$$

"Names of *bound variables* may be changed".

NB (Hilbert and McCarthy did it wrong; von Neumann found the bug)

$$\begin{array}{rcll} \mathsf{K}MN & \equiv & (\lambda xy.x)MN & \\ & \equiv & (((\lambda x(\lambda y\ x))M)N) & \\ & = & ((\lambda yM)N) & \\ & = & M & \text{assuming that } y \text{ not in } M. \end{array}$$

But

$$\begin{array}{rcl} \mathsf{K}yz & \equiv & (((\lambda x(\lambda y\ x))y)z) \\ & =_? & ((\lambda y\ y)z) \\ & = & z?? \end{array} \quad \text{better:} \quad \begin{array}{rcll} \mathsf{K}yz & \equiv & (((\lambda x'(\lambda y'\ x'))y)z) & \\ & = & (\lambda y'\ y)z & \\ & = & y & \text{as it should.} \end{array}$$

2.1.14 Böhm's Theorem

Let M, N be two λ-terms with different $\beta\eta$-nf. Then there exists an $F{\in}\Lambda$ such that

$$\begin{aligned} FM &=_\beta \lambda xy.x \\ FN &=_\beta \lambda xy.y \end{aligned}$$

In that case $\lambda + M{=}N$ becomes *inconsistent*

Representing computable functions

2.2.1 Two examples of data types: natural numbers and trees

Natural numbers:

$$\begin{array}{rcl} \texttt{Nat} & := & \texttt{zero | suc Nat} \\ \texttt{Tree} & := & \texttt{leaf | pair Tree Tree} \end{array}$$

Equivalently, as a context-free grammar

$$\begin{array}{rcl} \texttt{Nat} & \longrightarrow & \texttt{z | (s Nat)} \\ \texttt{Tree} & \longrightarrow & \texttt{l | (p Tree Tree)} \end{array}$$

We know what belongs to it

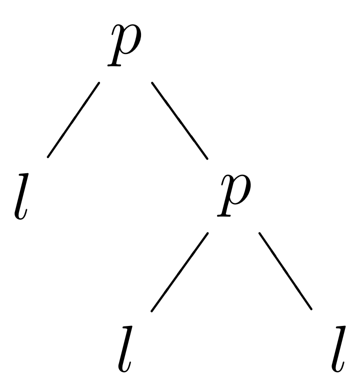
$$\texttt{Nat = }\{\texttt{z,(sz),(s(sz)),(s(s(sz))),...}\}\texttt{ = }\{\texttt{s}^n\texttt{z | } n\in\mathbb{N}\}$$

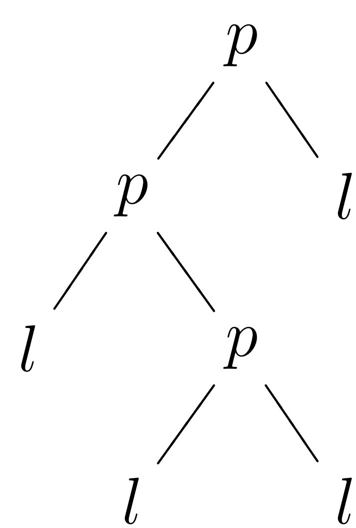
2.2.2 Trees

```
Tree  :=  l | (p Tree Tree)
```

Examples of elements of (language defined by) `Tree`

`(pl(pll))` and `(p(pl(pll))l)`





2.2.3 Translating data into lambda terms (Böhm-Berarducci)

$\texttt{Nat:} \quad \texttt{t} \rightsquigarrow \ulcorner \texttt{t} \urcorner := \lambda \texttt{sz.t}$

For example

$$\ulcorner(\texttt{s}(\texttt{s}(\texttt{sz})))\urcorner := \lambda \texttt{sz.}(\texttt{s}(\texttt{s}(\texttt{sz}))) \equiv_\alpha \lambda f x. f^3 x =: \mathbf{c}_3$$

$\texttt{Tree:} \quad \texttt{t} \rightsquigarrow \ulcorner \texttt{t} \urcorner := \lambda \texttt{pl.t}$

For example

$$\ulcorner(\texttt{pl}(\texttt{pll}))\urcorner = \lambda \texttt{pl.}(\texttt{pl}(\texttt{pll}))$$

2.2.4 Operating on data after representing them

For `Nat` we could operate on the codes to 'λ-define' functions:

$$
\begin{aligned}
\mathtt{A}_+ \ulcorner \mathtt{n} \urcorner\, \ulcorner \mathtt{m} \urcorner &=_\beta \ulcorner \mathtt{n} + \mathtt{m} \urcorner \\
\mathtt{A}_\times \ulcorner \mathtt{n} \urcorner\, \ulcorner \mathtt{m} \urcorner &=_\beta \ulcorner \mathtt{n} \times \mathtt{m} \urcorner
\end{aligned}
$$

We can do this for all computable functions

Define on `Trees` the operation of mirroring:

```
      Mirror (l)  =  l
 Mirror (p t1 t2)  =  (p (Mirror t2) (Mirror t1))
```

We will construct a λ-term A_M such that

$$A_M \ulcorner t \urcorner =_\beta \ulcorner \mathtt{Mirror(t)} \urcorner$$

2.2.5 The computable functions

A (k-ary) *numeric function* is a $\varphi : \mathbb{N}^k \to \mathbb{N}$

The *initial numeric functions* are defined by

$$\begin{array}{rcl} Z(n) & = & 0 \\ S^+(n) & = & n+1 \\ U_i^k(n_1,\ldots,n_k) & = & n_i \end{array}$$

Let $\mathcal{A}$ be a class of numeric functions.

(i) $\mathcal{A}$ is *closed under composition* if for all $\chi, \psi_1, \ldots, \psi_m \in \mathcal{A}$

$$\varphi = \lambda\!\!\lambda \vec{n}.\chi(\psi_1(\vec{n}),\ldots,\psi_m(\vec{n})) \;\Rightarrow\; \varphi \in \mathcal{A}$$

(ii) $\mathcal{A}$ is *closed under primitive recursion* if for all $\psi, \chi \in \mathcal{A}$ and φ defined by

$$\left.\begin{array}{rcl} \varphi(0,\vec{n}) & = & \chi(\vec{n}) \\ \varphi(k+1,\vec{n}) & = & \psi(\varphi(k,\vec{n}),k,\vec{n}) \end{array}\right\} \;\Rightarrow\; \varphi \in \mathcal{A}$$

(iii) $\mathcal{A}$ is *closed under minimalization* if for all $\chi \in \mathcal{A}$ and φ defined by

$$\varphi = \lambda\!\!\lambda \vec{n}.\mu m[\chi(\vec{n},m)=0] \;\Rightarrow\; \varphi \in \mathcal{A},$$

with χ such that $\forall \vec{n} \exists m.\chi(\vec{n},m)=0$.

2.2.6 The computable functions and their λ-definability

The *computable functions* are the smallest class $\mathcal{C}$
that contains the inital functions and
is closed under composition, primitive recursion and minimalization

A numeric function φ is λ-*definable* if there is an $F_\varphi \in \Lambda^{\varnothing}$ such that

$$\forall n_1 \dots n_k \in \mathbb{N} \, . F_\varphi \mathbf{c}_{n_1} \dots \mathbf{c}_{n_k} =_\beta \mathbf{c}_{\varphi(n_1,\dots,n_k)}$$

PROPOSITION. The initial functions are λ-definable

PROOF. Take $F_Z \equiv \lambda x.\mathbf{c}_0$, $F_{S^+} \equiv \lambda nfx.f(nfx)$, $F_{U_i^k} \equiv \lambda x_1 \dots x_k.x_i$.
One has e.g.

$$\begin{aligned} F_{S^+}\mathbf{c}_n &=_\beta \lambda fx.f(\mathbf{c}_n f x) \\ &=_\beta \lambda fx.f(f^n x) \\ &=_\beta \lambda fx.f^{n+1}x \\ &\equiv \mathbf{c}_{n+1} \quad \blacksquare \end{aligned}$$

2.2.7 λ-defining primitive recursion

S.C. Kleene invented the method to λ-define the predecessor:

$$\begin{array}{rcl} P^-(0) & = & 0 \\ P^-(n+1) & = & n \end{array}$$

Pairing

Define $[M,N] \equiv \lambda z.zMN$. Then $[M_1,M_2](\lambda x_1x_2.x_i) = M_i$

Kleene wanted to represent the informal $n \longmapsto [P^-(n),n]$:

$[0,0],[0,1],[1,2],[2,3],\ldots$

$$T : [P^-(n),n] \longmapsto [P^-(n+1),n+1]?$$

Take $F_T \equiv \lambda p.[p(\lambda xy.y), S^+(p(\lambda xy.y))]$. Then

$${F_T}^n[\mathbf{c}_0,\mathbf{c}_0] = [\mathbf{c}_{P^-(n)},\mathbf{c}_n]$$

Hence, $F_{P^-} \equiv \lambda n.nF_T[\mathbf{c}_0,\mathbf{c}_0]$ works as λ-definition of P^-

2.2.8 Representing the basic operation on Tree

LEMMA. There exists a $P \in \Lambda$ such that

$$P\ulcorner \mathtt{t}_1 \urcorner \ulcorner \mathtt{t}_2 \urcorner = \ulcorner \mathtt{p}\mathtt{t}_1\mathtt{t}_2 \urcorner \tag{1}$$

PROOF. Taking $P := \lambda \mathtt{t}_1\mathtt{t}_2\mathtt{pl}.\mathtt{p}(\mathtt{t}_1\mathtt{pl})(\mathtt{t}_2\mathtt{pl})$ we claim that (1) holds.

Note that $\mathtt{t} \in \mathtt{Tree}$ can be considered as a λ-term: $\mathtt{Tree} \subseteq \Lambda$

Since $\ulcorner \mathtt{t} \urcorner = \lambda \mathtt{pl}.\mathtt{t}$ one has $\ulcorner \mathtt{t} \urcorner \mathtt{pl} =_\beta \mathtt{t}$. Hence

$$\begin{aligned} P\ulcorner \mathtt{t}_1 \urcorner \ulcorner \mathtt{t}_2 \urcorner &= (\lambda \mathtt{t}_1\mathtt{t}_2\mathtt{pl}.\mathtt{p}(\mathtt{t}_1\mathtt{pl})(\mathtt{t}_2\mathtt{pl}))\ulcorner \mathtt{t}_1 \urcorner \ulcorner \mathtt{t}_2 \urcorner \\ &= \lambda \mathtt{pl}.\mathtt{p}(\ulcorner \mathtt{t}_1 \urcorner \mathtt{pl})(\ulcorner \mathtt{t}_2 \urcorner \mathtt{pl}) \\ &= \lambda \mathtt{pl}.\mathtt{p}\mathtt{t}_1\mathtt{t}_2 \\ &= \ulcorner \mathtt{p}\mathtt{t}_1\mathtt{t}_2 \urcorner. \blacksquare \end{aligned}$$

2.2.9 Representing mirroring in Λ

PROPOSITION. There exists an $A_M \in \Lambda$ such that for all $\mathtt{t} \in \mathtt{Tree}$

$$A_M \ulcorner \mathtt{t} \urcorner =_\beta \ulcorner \mathtt{Mirror(t)} \urcorner \tag{2}$$

PROOF. Take $A_M = \lambda \mathtt{tpl.tp'l}$, where $\mathtt{p}' = \lambda \mathtt{ab.pba}$.
We claim by induction that (2) holds. Note that $A_M \ulcorner \mathtt{t} \urcorner \mathtt{pl} = \ulcorner \mathtt{t} \urcorner \mathtt{p'l}$.

Case $\mathtt{t=l}$. Then

$$A_M \ulcorner \mathtt{l} \urcorner = \lambda \mathtt{pl.}(\lambda \mathtt{pl.l})\mathtt{p'l} = \lambda \mathtt{pl.l} = \ulcorner \mathtt{l} \urcorner = \mathtt{Mirror}(\ulcorner \mathtt{l} \urcorner).$$

Case $\mathtt{t} = \mathtt{pt_1t_2}$. Then

$$\begin{array}{rcll}
A_M \ulcorner \mathtt{pt_1t_2} \urcorner & = & \lambda \mathtt{pl.} \ulcorner \mathtt{pt_1t_2} \urcorner \mathtt{p'l} & \\
& = & \lambda \mathtt{pl.P} \ulcorner \mathtt{t_1} \urcorner \ulcorner \mathtt{t_2} \urcorner \mathtt{p'l} & \\
& = & \lambda \mathtt{pl.p'}(\ulcorner \mathtt{t_1} \urcorner \mathtt{p'l})(\ulcorner \mathtt{t_2} \urcorner \mathtt{p'l}) & \\
& = & \lambda \mathtt{pl.p}(\ulcorner \mathtt{t_2} \urcorner \mathtt{p'l})(\ulcorner \mathtt{t_1} \urcorner \mathtt{p'l}) & \\
& = & \lambda \mathtt{pl.p}(A_M \ulcorner \mathtt{t_2} \urcorner \mathtt{pl})(A_M \ulcorner \mathtt{t_1} \urcorner \mathtt{pl}) & \\
& = & \lambda \mathtt{pl.p}(\ulcorner \mathtt{Mirror(t_2)} \urcorner \mathtt{pl})(\ulcorner \mathtt{Mirror(t_1)} \urcorner \mathtt{pl}), & \text{by the IH,} \\
& = & \ulcorner \mathtt{p(Mirror(t_2))(Mirror(t_1))} \urcorner & \\
& = & \ulcorner \mathtt{Mirror(pt_1t_2)} \urcorner. \blacksquare &
\end{array}$$

Reduction in $\mathbf{CL}$ and λ

2.3.1 $\beta\eta$-reduction

$$\begin{array}{|rcl|}
\hline
\mathbf{I}P & \to_w & P \\
\mathbf{K}PQ & \to_w & P \\
\mathbf{S}PQR & \to_w & PR(QR) \\
\hline
(\lambda x.M)N & \to_\beta & M[x := N] \\
\lambda x.Mx & \to_\eta & M \\
\hline
\end{array}$$

Def

$$G_R(a) = \langle \{b \in \mathcal{A} \mid a \twoheadrightarrow_R b\}, \to_R \rangle$$

Exercise. Draw $G_\beta(M)$ with

$$\begin{array}{lcll}
M & := & WWW & W := \lambda xy.xyy \\
M & := & TT & T := \lambda x.\mathsf{I}xx \\
M & := & VV & V := \lambda x.\mathsf{I}(xx)
\end{array}$$

THEOREM β-reduction, η-reduction, and $\beta\eta$-reduction are CR

We have

$$
\begin{tikzcd}
\cdot \arrow[r, two heads, "\beta"] \arrow[d, "\beta"'] & \cdot \arrow[d, dotted, two heads, "\beta"] \\
\cdot \arrow[r, dotted, two heads, "\beta"'] & \cdot
\end{tikzcd}
\quad
\begin{tikzcd}
\cdot \arrow[r, "\eta"] \arrow[d, "\eta"'] & \cdot \arrow[d, dotted, "\eta", "\equiv"'] \\
\cdot \arrow[r, dotted, "\eta"', "\equiv"] & \cdot
\end{tikzcd}
\quad
\begin{tikzcd}
\cdot \arrow[r, "\eta"] \arrow[d, "\beta"'] & \cdot \arrow[d, dotted, "\beta", "\equiv"'] \\
\cdot \arrow[r, dotted, two heads, "\eta"'] & \cdot
\end{tikzcd}
$$

, hence

$$
\begin{tikzcd}
\cdot \arrow[r, two heads, "\beta"] \arrow[d, two heads, "\beta"'] & \cdot \arrow[d, dotted, two heads, "\beta"] \\
\cdot \arrow[r, dotted, two heads, "\beta"'] & \cdot
\end{tikzcd}
\quad
\begin{tikzcd}
\cdot \arrow[r, two heads, "\eta"] \arrow[d, two heads, "\eta"'] & \cdot \arrow[d, dotted, two heads, "\eta"] \\
\cdot \arrow[r, dotted, two heads, "\eta"'] & \cdot
\end{tikzcd}
\quad
\begin{tikzcd}
\cdot \arrow[r, two heads, "\eta"] \arrow[d, two heads, "\beta"'] & \cdot \arrow[d, dotted, two heads, "\beta"] \\
\cdot \arrow[r, dotted, two heads, "\eta"'] & \cdot
\end{tikzcd}
$$

, and

$$
\begin{tikzcd}
\cdot \arrow[r, two heads, "\beta\eta"] \arrow[d, two heads, "\beta\eta"'] & \cdot \arrow[d, dotted, two heads, "\beta\eta"] \\
\cdot \arrow[r, dotted, two heads, "\beta\eta"'] & \cdot
\end{tikzcd}
$$

2.3.2 Corollaries of the CR theorem

DEF. An equation $M = N$ is called inconsistent, notation $M \# N$, if $\lambda + M{=}N$ proves every equation, otherwise consistent

For example $\lambda xy.x \# \lambda xy.y$. These terms are called 'true, false'

PROP. λ is consistent, i.e.does not prove $x = y$. By the CR theorem.

COR. $M \# N \ \Rightarrow \ M \neq_\beta N$.

The converse is not true. Let $\Omega := (\lambda x.xx)(\lambda x.xx)$. Then $\Omega\mathsf{true} \neq_\beta \Omega\mathsf{false}$, but $\Omega\mathsf{true} = \Omega\mathsf{false}$ is consistent

COROLLARY. There are no terms P_1, P_2 such that
$P_1(xy) = x$ or $P_2(xy) = y$.

If P_1 exists, we can apply it twice to both sides of $\mathsf{true}\,\mathsf{I}\,\mathsf{I} = \mathsf{false}\,\mathsf{I}\,\mathsf{I}$.
If P_2 exists, we can apply it once to $\mathsf{K}\,\mathsf{I}\,\mathsf{true} = \mathsf{K}\,\mathsf{I}\,\mathsf{false}$. ■

2.4.1 Reflection in lambda calculus

Data types.

`nat`	$\longrightarrow$	`z	s(nat)`
`tree`	$\longrightarrow$	`b	P tree tree`
`ltree`	$\longrightarrow$	`L var	P ltree ltree	!ltree`
`var`	$\longrightarrow$	`x	var`$'$

Böhm-Berarducci (BB) representation of first two data types.

$\lambda sz.s^n z$ (Church numerals); $\lambda bP.Pb(Pbb)$, $\lambda bP.P(Pbb)(Pbb)$.

We have seen the representation of addition on `nat`.

Mirroring on `tree`: $F_{\mathrm{mirror}} \equiv \lambda tbP.tbP'$, where $P' \equiv \lambda ab.Pba$.

Then $F_{\mathrm{mirror}}(\lambda bP.Pb(Pbb)) =_\lambda \lambda bP.P(Pbb)b$.

2.4.2 Reflection in lambda calculus (2)

Tuples and projections:

$$\begin{array}{lcl} \langle M_1, \ldots, M_n \rangle & \equiv & \lambda z.z M_1 \ldots M_n. \\ U_i^n & \equiv & \lambda x_1 \ldots x_n.x_i. \end{array}$$

Then $\langle M_1, \ldots, M_n \rangle U_i^n =_\lambda M_i$.

Böhm-Piperno-Guerrini (BPG) representation of third data type.

Define

$$\begin{array}{lcl} F_L & \equiv & \lambda xe.eU_1^3 xe; \\ F_P & \equiv & \lambda xye.eU_2^3 xye; \\ F_! & \equiv & \lambda xe.eU_3^3 xe. \end{array}$$

or more mnemonical

$$\begin{array}{lcl} F_L x & =_\lambda & \lambda e.eU_1^3 xe; \\ F_P xy & =_\lambda & \lambda e.eU_2^3 xye; \\ F_! x & =_\lambda & \lambda e.eU_3^3 xe. \end{array}$$

Now define

$$\begin{array}{lcl} \ulcorner Lx \urcorner & =_\lambda & F_L x; \\ \ulcorner Pt_1 t_2 \urcorner & =_\lambda & F_P \ulcorner t_1 \urcorner \ulcorner t_2 \urcorner; \\ \ulcorner !t \urcorner & =_\lambda & F_! \ulcorner t \urcorner. \end{array}$$

or in nf

$$\begin{array}{lcl} \ulcorner Lx \urcorner & \equiv & \lambda e.eU_1^3 xe; \\ \ulcorner Pt_1 t_2 \urcorner & \equiv & \lambda e.eU_2^3 \ulcorner t_1 \urcorner \ulcorner t_2 \urcorner e; \\ \ulcorner !t \urcorner & \equiv & \lambda e.eU_3^3 \ulcorner t \urcorner e. \end{array}$$

PROPOSITION. Let A_1, A_2, A_3 be given lambda terms. Then there exists a H such that

$$\begin{array}{lcl} H(F_L x) & =_\lambda & A_1 x H; \\ H(F_P xy) & =_\lambda & A_2 xy H; \\ H(F_! x) & =_\lambda & A_3 x H. \end{array}$$

Hint. Try $H \equiv \langle \langle B_1, B_2, B_3 \rangle \rangle$.

APPLICATION. There exists an H that erases the !'s in an `ltree`.

$$\begin{array}{rcll} H(F_L x) & =_\lambda & F_L x, & \text{take } A_1 \equiv \lambda xh.F_L x; \\ H(F_P xy) & =_\lambda & F_P(Hx)(Hy), & \text{take } A_2 \equiv \lambda xyh.F_P(hx)(hy); \\ H(F_! x) & =_\lambda & Hx, & \text{take } A_3 \equiv \lambda xh.hx. \end{array}$$

2.4.3 Reflection in lambda calculus (3)

Coding lambda terms as other lambda terms in nf (Mogensen).

$$\begin{array}{rcll} \ulcorner x \urcorner & \equiv & \lambda e.eU_1^3 xe & =_\lambda \quad F_L x; \\ \ulcorner MN \urcorner & \equiv & \lambda e.eU_2^3 \ulcorner M \urcorner \ulcorner N \urcorner e & =_\lambda \quad F_P \ulcorner M \urcorner \ulcorner N \urcorner; \\ \ulcorner \lambda x.M \urcorner & \equiv & \lambda e.eU_3^3 (\lambda x.\ulcorner M \urcorner) e & =_\lambda \quad F_! (\lambda x.\ulcorner M \urcorner). \end{array}$$

By the above proposition there exists a lambda term E (self-interpreter) such that

$$\begin{array}{rcl} \mathsf{E} \ulcorner x \urcorner & =_\lambda & x; \\ \mathsf{E} \ulcorner MN \urcorner & =_\lambda & \mathsf{E} \ulcorner M \urcorner (\mathsf{E} \ulcorner N \urcorner); \\ \mathsf{E} \ulcorner \lambda x.M \urcorner & =_\lambda & \lambda x.(\mathsf{E} \ulcorner M \urcorner). \end{array}$$

Hence for all lambda terms M one has

$$\mathsf{E} \ulcorner M \urcorner =_\lambda M.$$

Following the construction one can take $\mathsf{E} \equiv \langle\langle \mathsf{K}, \mathsf{S}, \mathbb{C} \rangle\rangle$.

There exists lambda terms P_1, P_2 such that

$$P_1 \ulcorner MN \urcorner =_\lambda \ulcorner M \urcorner \text{ and } P_2 \ulcorner MN \urcorner =_\lambda \ulcorner N \urcorner.$$

There exists a lambda term Q such that

$$Q \ulcorner MN \urcorner \ulcorner L \urcorner =_\lambda \ulcorner ML \urcorner.$$

2.4.4 Reflection revisited

The last slide shows that reflection gives power. We can select from the code of a term (but not from the term itself) or we can replace part of it by the code of another term.

THEOREM. For all lambda terms F there is a lambda term X such that

$$F\ulcorner X\urcorner =_\lambda X.$$

APPLICATION. There is a term H such that

$$\begin{array}{lcll} H\,\mathsf{c}_n & = & \mathsf{c}_{3n} & \text{if } n \text{ is even;} \\ H\,\mathsf{c}_n & = & G\ulcorner H\urcorner\mathsf{c}_n & \text{else.} \end{array}$$

Typical use of reflection actually happens during translation (so called compiling) of higher programming languages into machine code. Often the compiler of the higher programming language is written in that language itself. In order to run that compiler the first time, one needs an older (usually less efficient) compiler in another language.

Typed lambda calculi

4.1.1 Simply typed lambda calculus $\lambda_{\to}^{\mathbb{A}}$ (Curry version)

Let $\mathbb{A}$ be a set of symbols. Types over $\mathbb{A}$, notation $\mathbb{T} = \mathbb{T}_{\to}^{\mathbb{A}}$:

$$\boxed{\mathbb{T} \quad = \quad \mathbb{A} \mid \mathbb{T} \to \mathbb{T}}$$

Type assignment

(axiom) $\quad \Gamma \vdash x : A$, if $(x{:}A) \in \Gamma$

$$(\to\mathrm{E}) \quad \frac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B} \qquad (\to\mathrm{I}) \quad \frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \to B)}$$

Examples

$$\left.\begin{array}{lll} \vdash \mathsf{I} & : & (A \to A) \\ \vdash \mathsf{K} & : & (A \to B \to A) \\ \vdash \mathsf{S} & : & (A \to B \to C) \to (A \to B) \to (A \to C) \end{array}\right\} \text{for all } A, B, C \in \mathbb{T}$$

Theorem. $\vdash M : A \;\Rightarrow\; M \in \mathrm{SN}$ (typable terms are strongly normalizing)

Theorem. Type checking is decidable; type reconstruction is computable

Theorem. $\vdash M : A \;\&\; M \twoheadrightarrow M' \;\Rightarrow\; \vdash M' : A$ (type checking only at compile time)

4.1.2 $\lambda_{\rightarrow}^{\mathbb{A}}$ (Church version)

$$\begin{array}{rcl} A\in\mathbb{A} & \Rightarrow & x^A\in\Lambda^{\mathbb{A}}(A) \\ M\in\Lambda^{\mathbb{A}}(A\to B),\ N\in\Lambda^{\mathbb{A}}(A) & \Rightarrow & (MN)\in\Lambda^{\mathbb{A}}(B) \\ M\in\Lambda^{\mathbb{A}}(B) & \Rightarrow & (\lambda x^A.M)\in\Lambda^{\mathbb{A}}(A\to B) \end{array}$$

Given $M\in\Lambda^{\mathbb{A}}(A)$, define $|M|\in\Lambda$ and Γ_M

M	$\|M\|$	Γ_M
x^A	x	$x : A$