Deadline: 31.10.2023 23:59:59

- Recommended book: Types and Programming Languages by B. C. Pierce; chapters 5 (untyped lambda calculus) and 9 (simply typed lambda calculus).
- Great paper on Reduction Strategies: Demonstrating Lambda Calculus Reduction by P.Sestoft

1 Simply typed lambda calculus (STLC, $\lambda_{\to}$)

1.1 Syntax (Pure Calculus extended with Base Types α_i)

$$\begin{array}{rll} M, N ::= & x \mid \lambda x{:}\, \tau . N \mid MN & \text{terms} \\ \tau, \sigma ::= & \alpha_i \mid \tau \to \sigma & \text{simple types} \end{array}$$

1.2 Typing Rules

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \text{Var/Ax} \qquad \frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x{:}\, \tau . M : \tau \to \sigma}\ \text{Abs}/\to \text{I} \qquad \frac{\Gamma \vdash N : \tau \quad \Gamma \vdash M : \tau \to \sigma}{\Gamma \vdash MN : \sigma}\ \text{App}/\to \text{E}$$

1.3 Reduction (Evaluation) Rules

$$\frac{M \longrightarrow M'}{M\ N \longrightarrow M'\ N}\ EApp_1 \qquad \frac{N \longrightarrow N'}{v\ N \longrightarrow v\ N'}\ EApp_2 \qquad \frac{}{(\lambda x : \tau . M)\ v \longrightarrow M[x/v]}\ EBeta$$

Note:

1. substitution always assumed to be capture-avoiding;
2. v is a value ($\lambda x : \tau . M$);
3. corresponds to call-by-value.

1.4 Examples

à la Curry	à la Church
$\lambda x.x\ :\ \alpha \to \alpha \qquad \lambda x.x\ :\ (\alpha \to \beta) \to \alpha \to \beta$ $\lambda xy.x\ :\ \alpha \to \beta \to \alpha$	$\lambda x^{\alpha}.x\ :\ \alpha \to \alpha \qquad \lambda x^{\alpha \to \beta}.x\ :\ (\alpha \to \beta) \to \alpha \to \beta$ $\lambda x^{\alpha} y^{\beta}.x\ :\ \alpha \to \beta \to \alpha$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f : \beta \to \gamma,\ g : \alpha \to \beta,\ x : \alpha \vdash f\ :\ \beta \to \gamma}\ Ax \quad \dfrac{\dfrac{}{\cdots, g : \alpha \to \beta, \cdots \vdash g\ :\ \alpha \to \beta}\ Ax \quad \dfrac{}{\cdots,\ x : \alpha \vdash x\ :\ \beta}\ Ax}{f : \beta \to \gamma,\ g : \alpha \to \beta,\ x : \alpha \vdash g\ x\ :\ \beta} \to E}{f : \beta \to \gamma,\ g : \alpha \to \beta,\ x : \alpha \vdash f\ (g\ x)\ :\ \gamma} \to E}{f : \beta \to \gamma,\ g : \alpha \to \beta \vdash \lambda\ x^{\alpha}.\ f\ (g\ x)\ :\ \alpha \to \gamma} \to I}{f : \beta \to \gamma \vdash \lambda\ g^{(\alpha \to \beta)} x^{\alpha}.\ f\ (g\ x)\ :\ (\alpha \to \beta) \to \alpha \to \gamma} \to I}{\vdash \lambda\ f^{(\beta \to \gamma)} g^{(\alpha \to \beta)} x^{\alpha}.\ f\ (g\ x)\ :\ (\beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma} \to I$$

$$\dfrac{\dfrac{\dfrac{x : Bool\ \in\ \{x : Bool\}}{x : Bool \vdash x : Bool}\ \text{Var}}{\vdash \lambda x : Bool.x\ :\ Bool \to Bool}\ \text{Abs} \quad \dfrac{}{\vdash true\ :\ Bool}\ \text{T-True}}{\vdash (\lambda x : Bool.x)\ true\ :\ Bool}\ \text{App}$$

1.5 Example: STLC extended with Let-bindings

$$
\begin{array}{rcll}
M, N & ::= & x \mid \lambda x.N \mid MN \mid \text{let } x = N \text{ in } M & \text{terms} \\
\tau, \sigma & ::= & \alpha_i \mid \tau \to \sigma & \text{types} \\
v & ::= & \lambda x.M & \text{value}
\end{array}
$$

new typing rule:

$$\frac{\Gamma \vdash N : \tau_1 \quad \Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \text{let } x = N \text{ in } M \; : \; \tau_2} \; let$$

new evaluation rules:

$$\text{let } x = v \text{ in } M \longrightarrow M[x/v] \quad LetV \qquad \frac{N \longrightarrow N'}{\text{let } x = N \text{ in } M \longrightarrow \text{let } x = N' \text{ in } M} \; Let$$

1.6 $\text{Int}_{\to}$ (Implicative part of Propositional Intuitionistic Logic)

$$\frac{}{\Gamma, \alpha \vdash \alpha} \; Ax \qquad \frac{\Gamma \vdash \alpha \to \beta \quad \Gamma \vdash \alpha}{\Gamma \vdash \beta} \to E \qquad \frac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \to \beta} \to I$$

Modus ponens

Hilbert's Deduction Theorem

Example:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \beta \to \gamma} \; Ax \quad \dfrac{\dfrac{}{\Gamma \vdash \alpha \to \beta} \; Ax \quad \dfrac{}{\Gamma \vdash \beta} \; Ax}{\Gamma \vdash \beta} \to E}{\Gamma \equiv \beta \to \gamma, \; \alpha \to \beta, \; \alpha \vdash \gamma} \to E}{\beta \to \gamma, \; \alpha \to \beta \vdash \alpha \to \gamma} \to I}{\beta \to \gamma \vdash (\alpha \to \beta) \to \alpha \to \gamma} \to I}{\vdash (\beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma} \to I$$

1.7 Encodings

Booleans

$T = true = \lambda t.\lambda f.\ t$
$F = false = \lambda t.\lambda f.\ f$
$test = \lambda l.\lambda m.\lambda n.\ l\ m\ n$ ($\sim$ if then else)
$and = \lambda l.\lambda m.\lambda n.\ l\ m\ n$

Pairs

$pair = \lambda f.\lambda s.\lambda b.\ b\ f\ s$
$fst = \lambda p.\ p\ T$
$snd = \lambda p.\ p\ F$

Church Numerals

$c_0 = \lambda s.\lambda z.\ z$
$c_1 = \lambda s.\lambda z.\ s\ z$
$c_i = \lambda s.\lambda z.\ s^i\ z$
$succ = \lambda n.\lambda s.\lambda z.\ s\ (n\ s\ z)$
$plus = \lambda m.\lambda n.\lambda s.\lambda z.\ m\ s\ (n\ s\ z)$
$times = \lambda m.\lambda n.\lambda s.\lambda z.\ m\ (n\ s)\ z$

$iszero = \lambda m.\ m\ (\lambda x.\ F)\ T$
$pred = \lambda n.\lambda s.\lambda z.\ n\ (\lambda f.\lambda h.\ f\ (g\ f))\ (\lambda u.z)\ (\lambda v.v)$
$subt = \lambda m.\lambda n.\ n\ pred\ m$

2 Recursion

Recursion in untyped lambda calculus can be expressed with so-called fix-point combinators.
Theorem. Forall term F there exists term V such that $V =_\beta F\ V$.
Proof: $V = (\lambda x.\ F\ (x\ x))(\lambda x.\ F\ (x\ ))$ Qed.
Theorem. $\exists \mathbb{Y}:\ \forall F.\ \mathbb{Y}\ F\ \ \to_\beta F\ (\mathbb{Y}\ F)$.
Proofs:

1. $\mathbb{Y} = \lambda f.\ (\lambda x.\ f\ (x\ x))\ (\lambda x.\ f\ (x\ x))$
2. $A = \lambda x.\lambda y.\ y\ (x\ x\ y);\ \Theta = A\ A$
3. ...

Qed.
Theorem (First Recursion Theorem). $\forall M.\exists F:\ F =_\beta M[f/F]$.
Proofs: $F = \mathbb{Y}\ (\lambda f.M)$ Qed.

Note, in $\lambda_\to$ recursion is impossible. One can extend $\lambda_\to$ with explicit fixed-point combinator by defining corresponding new term expression, typing and evaluation rules. The correct form of fixed-point combinator depends on evaluation strategy, for example $\mathbb{Y}$ is standard combinator for call-by-name, while Θ is standard call-by-value fixed-point combinator.
Example: factorial function $fact$ on church numerals
$f = \lambda f.\lambda n.$ if $iszero\ n$ then c_1 else $times\ n\ (f\ (pred\ n))$
$fact = fix\ g$
where fix is the corresponding fixed-point combinator.

2.1 Exercises

1. Prove that the following statements are derivable in STLC (provide type derivation)

   (a) $f : Bool \to Bool \vdash f\ (if\ false\ then\ true\ else\ false) : Bool$

$$\dfrac{\dfrac{}{f : Bool \to Bool \vdash f : Bool \to Bool}\,\text{Ax} \qquad \dfrac{\dfrac{}{\vdash false : Bool}\,\text{T-False} \quad \dfrac{}{\vdash true : Bool}\,\text{T-True} \quad \dfrac{}{\vdash false : Bool}\,\text{T-False}}{\vdash if\ false\ then\ true\ else\ false : Bool}\,\text{T-if}}{f : Bool \to Bool \vdash f\ (if\ false\ then\ true\ else\ false)\ :\ Bool}\,\to\text{E}$$

   (b) $f : Bool \to Bool \vdash \lambda x : Bool.\ f\ (if\ x\ then\ false\ else\ x)\ :\ Bool \to Bool$

$$\dfrac{\dfrac{\dfrac{}{f : Bool \to Bool \vdash f : Bool \to Bool}\,\text{Ax} \qquad \dfrac{\dfrac{}{x : Bool \vdash x : Bool}\,\text{Ax} \quad \dfrac{}{\vdash false : Bool}\,\text{T-False} \quad \dfrac{}{x : Bool \vdash x : Bool}\,\text{Ax}}{x : Bool \vdash if\ x\ then\ false\ else\ x : Bool}\,\text{T-if}}{f : Bool \to Bool, x : Bool \vdash f\ (if\ x\ then\ false\ else\ x)\ :\ Bool}\,\to\text{E}}{f : Bool \to Bool \vdash \lambda x : Bool.\ f\ (if\ x\ then\ false\ else\ x)\ :\ Bool}\,\to\text{I}$$

assuming

$$\frac{\Gamma \vdash e \ : \ Bool \quad \Gamma \vdash v \ : \ \tau \quad \Gamma \vdash u \ : \ \tau}{\Gamma \vdash if \ \ e \ then \ v \ else \ u \ : \ \tau}\ \text{T-If}$$

2. Find all inhabitants (closed terms) of the following types (both in à la Curch and à la Curry):

   (a) $(\alpha \to \beta) \to (\beta \to \gamma) \to \alpha \to \gamma$ - $\lambda f^{\alpha\to\beta} g^{\beta\to\gamma} x^{\alpha}.f(gx)$

   (b) $\alpha \to \beta \to (\alpha \to \beta \to \gamma) \to \gamma$ - $\lambda x^{\alpha} y^{\beta} f^{\alpha\to\beta\to\gamma}.fxy$

   (c) $((\alpha \to \beta \to \alpha) \to \alpha) \to \alpha$ - $\lambda f^{(\alpha\to\beta\to\alpha)\to\alpha}.f\left(\lambda x^{\alpha} y^{\beta}.x\right)$

   (d) $\beta \to ((\alpha \to \beta) \to \gamma) \to \gamma$ - $\lambda y^{\beta} f^{(\alpha\to\beta)\to\gamma}.f\,(\lambda x^{\alpha}.y)$

   (e) $\alpha \to (\alpha \to \alpha) \to \alpha$ - $\{\lambda x^{\alpha} f^{\alpha\to\alpha}.x, \lambda x^{\alpha} f^{\alpha\to\alpha}.fx, \lambda x^{\alpha} f^{\alpha\to\alpha}.f(fx), \ldots\}$

3. Compute the most general (principal) type of the following terms

   (a) $S = \lambda x\ y\ z.\ x\ z\ (y\ z)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \alpha \to \beta \to \gamma \vdash x : \alpha \to \beta \to \gamma}\ \text{Ax} \quad \dfrac{}{z : \alpha \vdash z : \alpha}\ \text{Ax}}{x : \alpha \to \beta \to \gamma, z : \alpha \vdash x\ z : \beta \to \gamma}\ \to \text{E} \quad \dfrac{\dfrac{}{y : \alpha \to \beta \vdash y : \alpha \to \beta}\ \text{Ax} \quad \dfrac{}{z : \alpha \vdash z : \alpha}\ \text{Ax}}{y : \alpha \to \beta, z : \alpha \vdash y\ z : \beta}\ \to \text{E}}{x : \alpha \to \beta \to \gamma, y : \alpha \to \beta, z : \alpha \vdash x\ z\ \ (y\ z) : \gamma}\ \to \text{E}}{x : \alpha \to \beta \to \gamma, y : \alpha \to \beta \vdash \lambda z.\ x\ z\ \ (y\ z) : \alpha \to \gamma}\ \to \text{I}}{x : \alpha \to \beta \to \gamma \vdash \lambda y\ z.\ x\ z\ \ (y\ z) : (\alpha \to \beta) \to \alpha \to \gamma}\ \to \text{I}}{\vdash \lambda x\ y\ z.\ x\ z\ \ (y\ z) : (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma}\ \to \text{I}$$

   (b) $K = \lambda x\ y.\ x$

$$\dfrac{\dfrac{\dfrac{}{x : \alpha, y : \beta \vdash x : \alpha}\ \text{Ax}}{x : \alpha \vdash \lambda y.\ x : \beta \to \alpha}\ \to \text{I}}{\vdash \lambda x\ y.\ x : \alpha \to \beta \to \alpha}\ \to \text{I}$$

   (c) SKK

$$\dfrac{\dfrac{\dfrac{}{\vdash S\ : (\alpha \to (\beta \to \alpha) \to \alpha) \to (\alpha \to \beta \to \alpha) \to \alpha \to \alpha}\ (a) \quad \dfrac{}{\vdash K : \alpha \to (\beta \to \alpha) \to \alpha}\ (b)}{\vdash SK : (\alpha \to \beta \to \alpha) \to \alpha \to \alpha}\ \to \text{E} \quad \dfrac{}{\vdash K : \alpha \to \beta \to \alpha}\ (b)}{\vdash SKK : \alpha \to \alpha}\ \to \text{E}$$

   (d) $I = \lambda x.\ x$

$$\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}\ \text{Ax}}{\vdash \lambda x.\ x : \alpha \to \alpha}\ \to \text{I}$$

4. Construct a derivation of type $((\alpha \to \beta) \to \gamma) \to \beta \to \gamma$ and the associated typed λ-term

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f : (\alpha \to \beta) \to \gamma \vdash f : (\alpha \to \beta) \to \gamma}\ \text{Ax} \quad \dfrac{\dfrac{}{x : \beta \vdash x : \beta}\ \text{Ax}}{x : \beta \vdash \lambda y^{\alpha}.x : \alpha \to \beta}\ \to \text{I}}{f : (\alpha \to \beta) \to \gamma, x : \beta \vdash f\,(\lambda y^{\alpha}.x) : \gamma}\ \to \text{E}}{f : (\alpha \to \beta) \to \gamma \vdash \lambda x^{\beta}.f\,(\lambda y^{\alpha}.x) : \beta \to \gamma}\ \to \text{I}}{\vdash \lambda f^{(\alpha\to\beta)\to\gamma}\ x^{\beta}.f\,(\lambda y^{\alpha}.x) : ((\alpha \to \beta) \to \gamma) \to \beta \to \gamma}\ \to \text{I}$$

5. Add product types to $\lambda_{\rightarrow}$, that is add $\sigma \times \tau$ to the types and

   (a) Add the appropriate term constructor and projections

$$
\begin{array}{rll}
M, N ::= & x \mid \lambda x{:}\,\tau.N \mid MN \mid (M, N) \mid \langle N| \mid |N\rangle & \text{terms} \\
\tau, \sigma ::= & \alpha_i \mid \tau \to \sigma \mid \sigma \times \tau & \text{simple types}
\end{array}
$$

   (b) Define typing rules

$$
\dfrac{\Gamma \vdash x : \sigma \quad \Gamma \vdash y : \tau}{\Gamma \vdash (x, y) : \sigma \times \tau}\ \text{Prd} \qquad \dfrac{\Gamma \vdash x : \sigma \times \tau}{\Gamma \vdash \langle x| : \sigma}\ \text{Fst} \qquad \dfrac{\Gamma \vdash x : \sigma \times \tau}{\Gamma \vdash |x\rangle : \tau}\ \text{Snd}
$$

   (c) Define reduction rules for the new term constructors

$$
\dfrac{M \longrightarrow M'}{(M, N) \longrightarrow (M', N)}\ EPrd_1 \qquad \dfrac{N \longrightarrow N'}{(v, N) \longrightarrow (v, N')}\ EPrd_2
$$

$$
\dfrac{}{\langle (M, N)| \longrightarrow M}\ EFst \qquad \dfrac{}{|(M, N)\rangle \longrightarrow N}\ ESnd
$$

   (d) Define (bii)map function for pairs and provide derivation for it

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f : \alpha \to \alpha' : f : \alpha \to \alpha'}\ \text{Ax} \quad \dfrac{\dfrac{}{x : \alpha \times \beta \vdash x : \alpha \times \beta}\ \text{Ax}}{x : \alpha \times \beta \vdash \langle x| : \alpha}\ \text{Fst}}{f : \alpha \to \alpha', x : \alpha \times \beta \vdash f\langle x| : \alpha'}\ \to \text{E} \quad \dfrac{\dfrac{}{g : \beta \to \beta' \vdash g : \beta \to \beta'}\ \text{Ax} \quad \dfrac{\dfrac{}{x : \alpha \times \beta \vdash x : \alpha \times \beta}\ \text{Ax}}{x : \alpha \times \beta \vdash |x\rangle : \beta}\ \text{Snd}}{g : \beta \to \beta', x : \alpha \times \beta \vdash g|x\rangle : \beta'}\ \to \text{E}}{f : \alpha \to \alpha', g : \beta \to \beta', x : \alpha \times \beta \vdash (f\langle x|, g|x\rangle) : \alpha' \times \beta'}\ \text{Prd}}{f : \alpha \to \alpha', g : \beta \to \beta' \vdash \lambda x^{\alpha\times\beta}.\ (f\langle x|, g|x\rangle) : \alpha \times \beta \to \alpha' \times \beta'}\ \to \text{I}}{f : \alpha \to \alpha' \vdash \lambda g^{\beta\to\beta'}\ x^{\alpha\times\beta}.\ (f\langle x|, g|x\rangle) : (\beta \to \beta') \to \alpha \times \beta \to \alpha' \times \beta'}\ \to \text{I}}{\vdash \lambda f^{\alpha\to\alpha'}\ g^{\beta\to\beta'}\ x^{\alpha\times\beta}.\ (f\langle x|, g|x\rangle) : (\alpha \to \alpha') \to (\beta \to \beta') \to \alpha \times \beta \to \alpha' \times \beta'}\ \to \text{I}
$$

6. Besides β-equivalence there exists another form of equivalence on lambda terms called η-equivalence or η-coercion (denoted $=_\eta$; η-expansion $\to_\eta$ and η-reduction $\leftarrow_\eta$). It is defined by $M =_\eta \lambda x.\ M\ x$. Note, it can't be expressed via β-reductions. Also, note that in untyped calculus a term can be η-expanded an arbitrary number of times, while in simply typed lambda calculus η-expansion is obviously limited by the term's type. Term of $\lambda_{\rightarrow}$ is said to be in η-long form if it is fully η-expanded; formally, it can be defined with the following grammar where $m \in \mathbb{N}$, $n, p \in \mathbb{N}_0$:

$$
\begin{array}{ll}
\Lambda^{lf}_{\text{odd}} & ::= \lambda x : \tau_1 \ldots x : \tau_p.\ \Lambda^{lf}_{\text{even}} \\
\Lambda^{lf}_{\text{odd}} & ::= x\ {\Lambda_1}^{lf}_{\text{odd}}\ \ldots\ {\Lambda_n}^{lf}_{\text{odd}} \mid {\Lambda_0}^{lf}_{\text{odd}}\ @_{\text{long}}\ {\Lambda_1}^{lf}_{\text{odd}}\ \ldots\ {\Lambda_m}^{lf}_{\text{odd}}
\end{array}
$$

   In other words, being viewed as a tree, all odd level nodes are abstractions over an arbitrary number of variables, while even level nodes are applications. Find the η-long form of the following term:

$$
test\ (mult\ c_3\ c_2)\ (snd\ (pair\ (and\ T\ F)\ c_1))
$$

7. Provide step-by-step evaluation of term $fact\ c_3$ with both call-by-name and call-by-value reduc-

tion strategies.

$$
\begin{aligned}
\mathit{fact}\ c_3 &\rightarrow_\beta f\ \mathit{fact}\ c_3 \\
&\rightarrow_\beta \text{if } \mathit{iszero}\ c_3 \text{ then } c_1 \text{ else } \mathit{times}\ c_3\ \ (\mathit{fact}\ c_2) \\
&\rightarrow_\beta \mathit{times}\ c_3\ \ (\mathit{fact}\ c_2) \\
&\rightarrow_\beta \mathit{times}\ c_3\ \ (\text{if } \mathit{iszero}\ c_2 \text{ then } c_1 \text{ else } \mathit{times}\ c_2\ \ (\mathit{fact}\ c_1)) \\
&\rightarrow_\beta \mathit{times}\ c_3\ \ (\mathit{times}\ c_2\ \ (\mathit{times}\ c_1\ \ (\mathit{fact}\ c_0))) \\
&\rightarrow_\beta \mathit{times}\ c_3\ \ (\mathit{times}\ c_2\ \ (\mathit{times}\ c_1\ \ (\text{if } \mathit{iszero}\ c_0 \text{ then } c_1 \text{ else } \mathit{times}\ c_0\ \ (\mathit{fact}\ \ (\mathit{pred}\ c_0))))) \\
&\rightarrow_\beta \mathit{times}\ c_3\ \ (\mathit{timrs}\ c_2\ \ (\mathit{times}\ c_1\ c_1)) \\
&\rightarrow_\beta c_6
\end{aligned}
$$