Typed Lambda Calculi
Lecture Notes

Gert Smolka
Saarland University

December 4, 2015

1 Simply Typed Lambda Calculus (STLC)

STLC is a simply typed version of $\lambda\beta$. The ability to express data types and recursion is lost, and thus Turing-completeness. The basic syntactic objects are types, terms, and contexts:

$$\begin{array}{rcll} A, B & ::= & X \mid A \to B \qquad (X : \mathbb{N}) & \textbf{types} \\ s, t & ::= & x \mid st \mid \lambda x{:}A.s \qquad (x : \mathbb{N}) & \textbf{terms} \\ \Gamma & ::= & () \mid \Gamma, x : A & \textbf{contexts} \end{array}$$

Contexts must satisfy the **side condition** that there is at most one assumption per variable. This side condition is not needed in the De Bruijn representation.

Reduction $s \succ t$ is defined as for $\lambda\beta$.

$$\frac{}{(\lambda x{:}A.s)t \succ s^x_t} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'} \qquad \frac{s \succ s'}{\lambda x{:}A.s \succ \lambda x{:}A.s'}$$

Substitution is realized analogous to $\lambda\beta$. The type annotations of abstractions are ignored by substitution and β-reduction.

The typing discipline is realized with an inductive **typing predicate** $\Gamma \vdash s : A$:

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash x : A}{\Gamma, y : B \vdash x : A}$$

$$\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash st : B} \qquad \frac{\Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x{:}A.s : A \to B}$$

Fact 1 If $\vdash s : A$, then s is closed.

Theorem 2 (Confluence) Reduction $s \succ t$ is confluent.

Theorem 3 (Type Preservation) If $\Gamma \vdash s : A$ and $s \succ t$, then $\Gamma \vdash t : A$.

Theorem 4 (Strong Normalization) If $\Gamma \vdash s : A$, then s is strongly normalizing.

Fact 5 (Unique Types) If $\Gamma \vdash s : A$ and $\Gamma \vdash s : B$, then $A = B$.

Fact 6 (Decidability) $\Gamma \vdash s : A$ is computationally decidable.

Fact 7 (Canonical Form) If $\vdash s : C$ and s is normal, then $s = \lambda x{:}A.t$ and $C = A \to B$ for some x, t, A, and B.

2 T

T is an extension of STLC with numbers and primitive recursion.

$$
\begin{array}{rcll}
A, B & ::= & \mathsf{N} \mid A \to B & \textbf{types} \\
s, t, u & ::= & x \mid st \mid \lambda x{:}A.s \mid \mathsf{O} \mid \mathsf{S}s \mid \mathsf{R}stu \qquad (x : \mathbb{N}) & \textbf{terms} \\
\Gamma & ::= & () \mid \Gamma, x{:}A & \textbf{contexts}
\end{array}
$$

Reduction $s \succ t$

$$
\frac{}{(\lambda x{:}A.s)t \succ s^x_t} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'} \qquad \frac{s \succ s'}{\lambda x{:}A.s \succ \lambda x{:}A.s'}
$$

$$
\frac{}{\mathsf{RO}tu \succ t} \qquad \frac{}{\mathsf{R}(\mathsf{S}s)tu \succ us(\mathsf{R}stu)} \qquad \frac{s \succ s'}{\mathsf{S}s \succ \mathsf{S}s'} \qquad \frac{s \succ s'}{\mathsf{R}stu \succ \mathsf{R}s'tu}
$$

Substitution is realized analogous to $\lambda\beta$.

Typing $\Gamma \vdash s : A$

$$
\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash x : A}{\Gamma, y : B \vdash x : A}
$$

$$
\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash st : B} \qquad \frac{\Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x{:}A.s : A \to B}
$$

$$
\frac{}{\Gamma \vdash \mathsf{O} : \mathsf{N}} \qquad \frac{\Gamma \vdash s : \mathsf{N}}{\Gamma \vdash \mathsf{S}s : \mathsf{N}} \qquad \frac{\Gamma \vdash s : \mathsf{N} \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : \mathsf{N} \to A \to A}{\Gamma \vdash \mathsf{R}stu : A}
$$

Theorem 8 (Confluence) Reduction $s \succ t$ is confluent.

Theorem 9 (Type Preservation) If $\Gamma \vdash s : A$ and $s \succ t$, then $\Gamma \vdash t : A$.

Theorem 10 (Strong Normalization) If $\Gamma \vdash s : A$, then s is strongly normalizing.

Fact 11 (Unique Types) If $\Gamma \vdash s : A$ and $\Gamma \vdash s : B$, then $A = B$.

Fact 12 (Decidability) $\Gamma \vdash s : A$ is computationally decidable.

Fact 13 (Canonical Forms) Let s be normal.

1. If $\vdash s : \mathsf{N}$, then $s = \mathsf{S}^n\mathsf{O}$ for some n.
2. If $\vdash s : A \to B$, then $s = \lambda x{:}A.t$ for some x and t.

To have simple canonical forms as specified by the fact, it is essential that the constructor S and the recursor R are provided through full syntactic forms rather than constants.

From Coq's perspective, T extends STLC with an inductive type for numbers. Since there are only simple types, the recursor does not provide for proofs.

Exercise 14 Let D be a type. Give types A, B, C such that
$\vdash \lambda x{:}A.\lambda y{:}B.\lambda z{:}C.\mathsf{R}xyz : A \to B \to C \to D$.

3 PCF

PCF is a deterministic weak call-by-value version of STLC with a fixed point operator providing full recursion and numbers added. PCF is Turing-complete and may be seen as a simply typed version of L with numbers.

$$
\begin{array}{rcllr}
A, B & ::= & \mathsf{N} \mid A \to B & & \textbf{types} \\
s, t, u & ::= & x \mid st \mid \lambda x{:}A.s \mid \mu x{:}A.s \mid \mathsf{O} \mid \mathsf{S}s \mid \mathsf{M}stu & (x : \mathbb{N}) & \textbf{terms} \\
\Gamma & ::= & () \mid \Gamma, x{:}A & & \textbf{contexts} \\
v & ::= & \lambda x{:}A.s \mid \mathsf{O} \mid \mathsf{S}v & & \textbf{values}
\end{array}
$$

Numbers are accommodated with the constructs O and S and a match construct M. Fixed points are provided by the syntactic form starting with μ.

Reduction $s \succ t$

$$
\frac{}{(\lambda x{:}A.s)v \succ s^x_v} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{vt \succ vt'} \qquad \frac{}{\mu x{:}A.s \succ s^x_{\mu x : A.s}}
$$

$$
\frac{}{\mathsf{MO}tu \succ t} \qquad \frac{}{\mathsf{M}(\mathsf{S}v)tu \succ uv} \qquad \frac{s \succ s'}{\mathsf{S}s \succ \mathsf{S}s'} \qquad \frac{s \succ s'}{\mathsf{M}stu \succ \mathsf{M}s'tu}
$$

Substitution s^x_t is realized analogous to L. Thus reduction is only meaningful for closed terms, which suffices for programming languages.

Typing $\Gamma \vdash s : A$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash x : A}{\Gamma, y : B \vdash x : A}$$

$$\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash st : B} \qquad \frac{\Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x{:}A.s : A \to B} \qquad \frac{\Gamma, x : A \vdash s : A}{\Gamma \vdash \mu x{:}A.s : A}$$

$$\frac{}{\Gamma \vdash \mathsf{O} : \mathsf{N}} \qquad \frac{\Gamma \vdash s : \mathsf{N}}{\Gamma \vdash \mathsf{S}s : \mathsf{N}} \qquad \frac{\Gamma \vdash s : \mathsf{N} \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : \mathsf{N} \to A}{\Gamma \vdash \mathsf{M}stu : A}$$

Fact 15 (Determinism) Reduction $s \succ t$ is functional.

Theorem 16 (Type Preservation) If $\vdash s : A$ and $s \succ t$, then $\vdash t : A$.

Type preservation holds only for closed terms since naive substitution is employed.

Fact 17 (Unique Types) If $\Gamma \vdash s : A$ and $\Gamma \vdash s : B$, then $A = B$.

Fact 18 (Decidability) $\Gamma \vdash s : A$ is computationally decidable.

Fact 19 (Canonical Forms) Let s be normal.

1. If $\vdash s : \mathsf{N}$, then $s = \mathsf{S}^n\mathsf{O}$ for some n.
2. If $\vdash s : A \to B$, then $s = \lambda x{:}A.t$ for some x and t.

4 F

F extends STLC with polymorphic types $\forall X{:}\mathsf{P}.A$. We present F with a single sorted syntax, where terms include types. F is a subsystem of Coq's type theory. The term P represents Prop.

$$\begin{array}{rcll} s, t, A, B & ::= & x \mid \mathsf{P} \mid A \to B \mid \forall x.A \mid st \mid \lambda x{:}A.s \qquad (x : \mathbb{N}) & \textbf{terms} \\ \Gamma & ::= & () \mid \Gamma, x : A & \textbf{contexts} \end{array}$$

Note that $\forall$ and λ are binders. Terms that are equal up to renaming of bound variables are identified. Contexts must satisfy the side condition specified for STLC.

Substitution s^x_t
Substitution satisfies the following equations:

$$\begin{array}{rcll}
x^y_u & = & \text{if } x{=}y \text{ then } u \text{ else } x & \\
\mathsf{P}^y_u & = & \mathsf{P} & \\
(A \to B)^x_u & = & A^x_u \to B^x_u & \\
(\forall x.A)^y_u & = & \forall x.A^y_u & \text{if } x \neq y \text{ and } x \text{ not free in } u \\
st^y_u & = & s^y_u\, t^y_u & \\
(\lambda x{:}A.s)^y_u & = & \lambda x{:}A^y_u.\, s^y_u & \text{if } x \neq y \text{ and } x \text{ not free in } u
\end{array}$$

Reduction $s \succ t$

$$\frac{}{(\lambda x{:}A.s)t \succ s^x_t} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'} \qquad \frac{s \succ s'}{\lambda x{:}A.s \succ \lambda x{:}A.s'}$$

Typing $\Gamma \vdash s : A$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash x : A}{\Gamma, y : B \vdash x : A}$$

$$\frac{\Gamma \vdash A : \mathsf{P} \qquad \Gamma \vdash B : \mathsf{P}}{\Gamma \vdash A \to B : \mathsf{P}} \qquad \frac{\Gamma, x : \mathsf{P} \vdash A : \mathsf{P}}{\Gamma \vdash \forall x.A : \mathsf{P}}$$

$$\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash st : B} \qquad \frac{\Gamma \vdash s : \forall x.B \qquad \Gamma \vdash A : \mathsf{P}}{\Gamma \vdash sA : B^x_A}$$

$$\frac{\Gamma \vdash A \to B : \mathsf{P} \qquad \Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x{:}A.s : A \to B} \qquad \frac{\Gamma \vdash \forall x.A : \mathsf{P} \qquad \Gamma, x : \mathsf{P} \vdash s : A}{\Gamma \vdash \lambda x{:}\mathsf{P}.s : \forall x.A}$$

Valid contexts
Valid contexts are defined as follows:

$$\frac{}{()\ \text{valid}} \qquad \frac{\Gamma\ \text{valid}}{\Gamma, x : \mathsf{P}\ \text{valid}} \qquad \frac{\Gamma\ \text{valid} \qquad \Gamma \vdash A : \mathsf{P}}{\Gamma, x : A\ \text{valid}}$$

Theorem 20 (Confluence) Reduction $s \succ t$ is confluent.

Theorem 21 (Type Preservation) If $\Gamma \vdash s : A$ and $s \succ t$, then $\Gamma \vdash t : A$.

Theorem 22 (Strong Normalization) If $\Gamma \vdash s : A$, then s is strongly normalizing.

Fact 23 (Unique Types) If $\Gamma \vdash s : A$ and $\Gamma \vdash s : B$, then $A = B$.

Fact 24 (Propagation) If $\Gamma \vdash s : A$ and Γ is valid, then $\Gamma \vdash A : \mathsf{P}$.

Fact 25 (Decidability) $\Gamma \vdash s : A$ is computationally decidable.

Fact 26 (Canonical Forms) Let s be normal.

1. If $\vdash s : \mathsf{P}$, then s has either the form $A \to B$ or the form $\forall x.A$.
2. If $\vdash s : A \to B$, then $s = \lambda x : A.t$ for some x and t.
3. If $\vdash s : \forall x.A$, then $s = \lambda x : \mathsf{P}.t$ for some x and t.

F is a computational system subsuming T. The type of natural numbers can be represented as

$$\mathsf{N} \;:=\; \forall X.\, X \to (X \to X) \to X$$

The canonical members of this type are the Church numerals with reversed argument order:

$$\lambda X : \mathsf{P}.\, \lambda x : X.\, \lambda f : X \to X.\; f^n x$$

The argument reversal is needed since otherwise there would be an additional canonical element representing 1. All inductive data types can be represented in F.

F is also a logical system subsuming intuitionistic propositional logic:

$$\begin{aligned}
\bot \;&:=\; \forall Z.Z \\
A \wedge B \;&:=\; \forall Z.\, (A \to B \to Z) \to Z \\
A \vee B \;&:=\; \forall Z.\, (A \to Z) \to (B \to Z) \to Z \\
\exists X.\, A \;&:=\; \forall Z.\, (\forall X.\, A \to Z) \to Z
\end{aligned}$$

5 Calculus of Constructions

The basic type theory underlying Coq is known as calculus of constructions. We consider CC_ω, a version of the calculus of construction with an infinite cumulative hierarchy of universes.

$$\begin{array}{rcllr}
u & ::= & \mathsf{U}_n & & \textbf{universes} \\
s, t, A, B & = & x \mid u \mid \forall x : A.B \mid st \mid \lambda x : A.s & (x : \mathbb{N}) & \textbf{terms} \\
\Gamma & ::= & () \mid \Gamma, x : A & & \textbf{contexts}
\end{array}$$

Note that $\forall$ and λ are binders. Terms that are equal up to renaming of bound variables are identified. Contexts must satisfy the side condition specified for STLC.

Reduction $s \succ t$ is obtained with the β-rule $(\lambda x : A.s)t \succ s^x_t$ that can be applied everywhere. **Equivalence** $s \equiv t$ is defined as the equivalence closure of β-reduction.

Subtyping $A \preceq B$ is defined as follows:

$$\frac{}{A \preceq A} \qquad \frac{m < n}{\mathsf{U}_m \preceq \mathsf{U}_n} \qquad \frac{B \preceq B'}{\forall x{:}A.\,B \preceq \forall x{:}A.\,B'}$$

Typing $\Gamma \vdash s : A$ is defined as follows:

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash x : A}{\Gamma, y : B \vdash x : A}$$

$$\frac{}{\Gamma \vdash \mathsf{U}_n : \mathsf{U}_{n+1}}$$

$$\frac{\Gamma \vdash A : u \qquad \Gamma, x : A \vdash B : u}{\Gamma \vdash \forall x{:}A.\,B : u}$$

$$\frac{\Gamma \vdash s : \forall x{:}A.\,B \qquad \Gamma \vdash t : A}{\Gamma \vdash st : B^x_t}$$

$$\frac{\Gamma \vdash A : u \qquad \Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x{:}A.s : \forall x{:}A.\,B}$$

$$\frac{\Gamma \vdash s : A \qquad \Gamma \vdash B : u}{\Gamma \vdash s : B}\; A \equiv_\beta B$$

$$\frac{\Gamma \vdash s : A}{\Gamma \vdash s : B}\; A \preceq B$$

$$\frac{\Gamma \vdash A : u \qquad \Gamma, x : A \vdash B : \mathsf{U}_0}{\Gamma \vdash \forall x{:}A.\,B : \mathsf{U}_0}$$

One says that the last rule makes U_0 **impredicative**. It turns out that U_0 is the only universe that can be made impredicative without losing consistency [Harper and Pollak, 1991].

Valid contexts are defined as follows:

$$\frac{}{\emptyset \text{ valid}} \qquad \frac{\Gamma \text{ valid} \qquad \Gamma \vdash A : u}{\Gamma, x : A \text{ valid}}\; x \notin \Gamma$$

Theorem 27 (Confluence) Reduction $s \succ t$ is confluent.

Theorem 28 (Type Preservation) If $\Gamma \vdash s : A$ and $s \succ t$, then $\Gamma \vdash t : A$.

Theorem 29 (Strong Normalization) If $\Gamma \vdash s : A$, then s is strongly normalizing.

Fact 30 (Propagation) If $\Gamma \vdash s : A$ and Γ is valid, then $\Gamma \vdash A : u$ for some universe u.

Fact 31 (Decidability) $\Gamma \vdash s : A$ is computationally decidable.

Fact 32 (Canonical Forms) Let s be normal.

1. If $\vdash s : u$, then s is either a universe or a function type $\forall x{:}A.B$.
2. If $\vdash s : \forall x.A$, then s has the form $\lambda x{:}B.t$.

6 Notes

Two textbooks covering typed lambda calculi and the calculus of abstractions are Sørensen and Urzyczyn [4] and Nederpelt and Geuvers [3]. Luo [2] presents an extension of CC_ω with a strong normalization proof. A presentation of PCF can be found in Harper [1].

References

[1] Robert Harper. *Practical foundations for programming languages.* Cambridge University Press, 2013.

[2] Zhaohui Luo. *Computation and reasoning: a type theory for computer science.* Oxford University Press, Inc., 1994.

[3] Rob Nederpelt and Herman Geuvers. *Type Theory and Formal Proof, An Introduction.* Cambridge University Press, 2014.

[4] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism.* Elsevier, 2006.