PROOF101, Spring 2026, Lecture Slides 03

American University of Beirut

Simply-Typed Lambda-Calculus

Assaf Kfoury

March 2026

Outline of Lecture Slides 03

- Main reference for these lecture slides is the book:

  Lectures on the Curry-Howard Isomorphim

  by Sorensen and Urzyczyn – henceforth denoted [LCHI]

- Most details are from [LCHI, Chapter 3]

- Headings in this set of slides (a little different from [LCHI, Chapt 3]) :

  - Simply-typed λ-calculus *à la* Curry [LCHI, Sect 3.1]
  - Type reconstruction, type inhabitation, type checking [LCHI, Sect 3.2]
  - Simply-typed λ-calculus *à la* Church [LCHI, Sect 3.3]
  - Church typing vs. Curry typing [LCHI, Sect 3.4]
  - Strong Normalization and Confluence [LCHI, Sects 3.5, 3.6]
  - Expressibility [LCHI, Sect 3.7]

Outline of Lecture Slides 03

Important things to remember from [LCHI, Chapter 3] :

- *Typing* à la *Curry versus Typing* à la *Church*, slides 4– 11 ,
- *Subject Reduction*, slides 6 , 10 ,
- *Strong Normalization* and *Confluence*, slide 13 .

If you care to know more about the simply-typed λ-calculus beyond [LCHI] :

- Roger Hindley's book: *Basic Simple Type Theory* .
- Wikipedia page *Simply Typed Lambda Calculus* and the references therein .

One important topic omitted in [LCHI] as well as in these slides is the *denotational semantics* of the simply-typed λ-calculus. It necessitates a preliminary introduction to *domain theory* and *category theory*, which would both take us far afield and away from the main focus of this course.

Simply-Typed λ-Calculus *à la* Curry

- Recall from [LCHI, Chapter 1] :

  Λ = the set of terms of the *pure λ-calculus* :

  $$M, N \in \Lambda \ ::= \ x \ \big| \ (M\ N) \ \big| \ (\lambda x\, M)$$

- Recall from [LCHI, Chapter 2] :

  PV $= \{p, q, r, \ldots\}$ is the set of *propositional (or type) variables* .
  Φ = the set of *propositional wff's (or types)* over PV .
  $\Phi_{\to}$ = the set of *implicational propositional wff's (or types)* over PV:

  $$\varphi, \psi \in \Phi_{\to} \ ::= \ p \ \big| \ \varphi \to \psi$$

- $\Phi_{\to}$ = the set of *simple types* = the set of *implicational prop. wff's*
  (these are the types of the "simplest" simply-typed λ-calculus) .

Simply-Typed λ-Calculus *à la* Curry

The simply-typed λ-Calculus *à la* Curry, henceforth **$\boldsymbol{\lambda_{\rightarrow}}$-Curry** :

- *Typing rules* (or *type-assignment rules*) for $\boldsymbol{\lambda_{\rightarrow}}$-Curry:

$$\text{(Var)} \qquad \Gamma,\ x : \tau \ \vdash\ x : \tau$$

$$\text{(Abs)} \qquad \frac{\Gamma,\ x : \sigma \ \vdash\ M : \tau}{\Gamma \ \vdash\ (\lambda x.\ M) : \sigma \rightarrow \tau}$$

$$\text{(App)} \qquad \frac{\Gamma \ \vdash\ M : \sigma \rightarrow \tau \qquad\qquad \Gamma \ \vdash\ N : \sigma}{\Gamma \ \vdash\ (M\ N) : \tau}$$

where

Γ is an *environment* (of *type assumptions*) $\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$,

$M, N \in \Lambda$,

$\sigma,\ \tau\ \in \Phi_{\rightarrow}$.

Simply-Typed λ-Calculus *à la* Curry

Correctness of the type-assignment rules for $\boldsymbol{\lambda}_{\rightarrow}$-Curry is confirmed by:

- **THEOREM** (Subject Reduction for $\boldsymbol{\lambda}_{\rightarrow}$-Curry)*
  If $\Gamma \vdash M : \sigma$ *and* $M \twoheadrightarrow_{\beta} N$, *then* $\Gamma \vdash N : \sigma$.

  "Types are preserved under β-reduction."

  Proof and preliminary lemmas are in [LCHI, Sect 3.1, pp58-59] .

In some situations they call *values* terms in Λ that are λ-abstractions (*call-by-value* evaluation, not considered in [LCHI]), because no evaluation is permitted under a 'λ'.

- **THEOREM** (Progress for $\boldsymbol{\lambda}_{\rightarrow}$-Curry)
  If M is a closed well-typed term, i.e., there is a derivation according to the rules of $\boldsymbol{\lambda}_{\rightarrow}$-Curry *whose last judgement is* $\vdash M : \sigma$, *then either M is a* ***value*** *or M can be evaluated to another term N (which, by subject reduction, has type* σ*)* .
- The combination *subject reduction* + *progress* is often said to show that the typing system is *type-sound* or also that it is *type-safe* .

---

* The name "subject reduction" is a little mysterious, but it does have a convoluted explanation. Look for it on the Web. I prefer "type preservation".

Type Reconstruction, Type Inhabitation, Type Checking

- Three related problems [LCHI, Sect 3.2, pp 60-63] :
  1. *Type reconstruction* (also called *typability*) :

     $$? \vdash M : ? \qquad \text{or, if } \mathrm{FV}(M) = \varnothing, \qquad \vdash M : ?$$

  2. *Type inhabitation* :

     $$\Gamma \vdash ? : \tau \qquad \text{or, if } \mathrm{FV}(M) = \varnothing, \qquad \vdash ? : \tau$$

  3. *Type checking* :

     "Is $\Gamma \vdash M : \tau$ a derivable judgment?"

     or, if $\mathrm{FV}(M) = \varnothing$, "is $\vdash M : \tau$ a derivable judgment?"

- **THEOREM**
  - *Type reconstruction and type checking are logarithmic-space equivalent and* PTIME*-complete .*
    [LCHI, Thm 3.2.7, p 62]
  - *Type inhabitation is* PSPACE*-complete .*
    [LCHI, Thm 4.2.5, p 81, Thm 2.4.12, p 43]

Simply-Typed λ-Calculus *à la* Church

- $\Phi_{\to}$ = the set of *simple types*,
  the same for both **$\lambda_{\to}$-Curry** and **$\lambda_{\to}$-Church** .
- $\Lambda(\Phi_{\to})$ = the set of $\Phi_{\to}$*-annotated* (*i.e.*, *annotated-with-simple-types*) λ-terms:

$$M, N \in \Lambda(\Phi_{\to}) \ ::= \ x \mid (M\ N) \mid (\lambda x : \sigma\ M)$$

  where x ranges over the set of λ-variables,
  and σ ranges over the set $\Phi_{\to}$ of simple types.

- **Note** the difference with the set Λ of pure λ-terms:
  Every λ-binding in $\Lambda(\Phi_{\to})$ is annotated with a type, as in "$\lambda x : \sigma$" .
- **Note** the (non-essential) difference with [LCHI, Sect 3.3, pp 63-65] , where types are inserted as superscripts as in "$(\lambda x^{\sigma}\ M)$" rather than "$(\lambda x : \sigma\ M)$" .[†]

[†] Barendregt points out disadvantages of inserting types as superscripts, Example 3.2.13 in his *Lambda Calculi with Types* in *Handbook of Logic in Computer Science, Vol. II* , ed. Abramsky *et al.* Read also [LCHI, Convention 3.4.2, p 66] .

Simply-Typed λ-Calculus *à la* Church

- *Typing rules* (or *type-assignment rules*) for $\boldsymbol{\lambda}_{\rightarrow}$-Church:

$$\text{(Var)} \qquad \Gamma,\ x : \tau \ \vdash\ x : \tau$$

$$\text{(Abs)} \qquad \frac{\Gamma,\ x : \sigma \ \vdash\ M : \tau}{\Gamma \ \vdash\ \big(\,\lambda x : \sigma\,.\,M\big) : \sigma \rightarrow \tau}$$

$$\text{(App)} \qquad \frac{\Gamma \ \vdash\ M : \sigma \rightarrow \tau \qquad\qquad \Gamma \ \vdash\ N : \sigma}{\Gamma \ \vdash\ \big(M\ N\big) : \tau}$$

where

Γ is an *environment* (of *type assumptions*) $\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$,

$M, N \in \Lambda(\Phi_{\rightarrow})$, and $\sigma,\ \tau\ \in \Phi_{\rightarrow}$.

- **Note** the difference between the *(Abs) rule for* $\boldsymbol{\lambda}_{\rightarrow}$-Curry and the *(Abs) rule for* $\boldsymbol{\lambda}_{\rightarrow}$-Church . In the latter, a type assumption $x : \sigma$ is discharged together with its type to produce the binding $\lambda x : \sigma$.

Simply-Typed λ-Calculus *à la* Church

Correctness of the type-assignment rules for $\boldsymbol{\lambda}_{\rightarrow}$-Church is confirmed by:

- **THEOREM** (Subject Reduction for $\boldsymbol{\lambda}_{\rightarrow}$-Church)[‡]
  If $\Gamma \vdash M : \sigma$ *and* $M \twoheadrightarrow_{\beta} N$, *then* $\Gamma \vdash N : \sigma$.
- Proof and preliminary lemmas in [LCHI, Sect 3.3, p 65] .

[‡] See footnote ∗.

Church Typing vs. Curry Typing

Many intimate connections between **$\lambda_{\to}$-Curry** and **$\lambda_{\to}$-Church**:

- The *erasing*, or *type-erasure*, map $\left|\cdot\right| : \Lambda(\Phi_{\to}) \to \Lambda$ from Church-style to Curry-style:

$$\begin{aligned} \left|x\right| &\triangleq x \\ \left|(M\,N)\right| &\triangleq (\left|M\right|\ \left|N\right|) \\ \left|(\lambda x : \sigma.\ M)\right| &\triangleq (\lambda x.\ \left|M\right|) \end{aligned}$$

- **PROPOSITION.** *For all Church-style terms $M, N \in \Lambda(\Phi_{\to})$:*
  1. If $M \to_{\beta} N$ then $\left|M\right| \to_{\beta} \left|N\right|$.
  2. If $\Gamma \vdash_{\text{Church}} M : \sigma$ then $\Gamma \vdash_{\text{Curry}} \left|M\right| : \sigma$.
- Proofs, and further details and (easy) facts relating $\boldsymbol{\lambda}_{\to}$-Curry and $\boldsymbol{\lambda}_{\to}$-Church are in [LCHI, Sect 3.5, pp 65-67] .

Church Typing vs. Curry Typing (an Aside)

A characterization of the difference between **$\lambda_{\to}$-Curry** and **$\lambda_{\to}$-Church** by J.C. Reynolds in *Theories of Programming Languages*, [Sects 15.4, 15.5] :

- Curry style is the *extrinsic theory of types*:

  "The meaning of a [λ-term] is the same as in an untyped language, while the meaning of a type describes a set of results. When a [λ-term] has a type, it evaluates to a member of the set described by the type."

- Church style is the *intrinsic theory of types*:

  "A [λ-term] only has a meaning when it satisfies a typing judgement, the kind of meaning depends on the judgement, and a [λ-term] satisfying several judgements will have several meanings."

- Details in *intrinsic and extrinsic views of typing* or in intrinsic vs. extrinsic .

- To confuse you a little more, there is also:
  - something called *intensional type theory* (different from *intrinsic theory*)
  - something called *extensional type theory* (different from *extrinsic theory*)

The distinction *intensional* vs. *extensional* will play a role when we later consider the *Calculus of Inductive Constructions* and *Homotopy Type Theory*.

Strong Normalization and Confluence

In addition to *Subject Reduction* theorems, slides 6 and 10,
the following are fundamental properties of the simply-typed λ-calculus:

- **THEOREM** (Strong Normalization)
  All terms of $\boldsymbol{\lambda_{\rightarrow}}$**-Curry** *and* $\boldsymbol{\lambda_{\rightarrow}}$**-Church** *are strongly normalizing .*

  There are many very different proofs of this theorem. One particularly elegant proof is in [LCHI, Sect 3.5, pp 67-70] .

- **THEOREM** (Confluence / Church-Rosser Property)
  For all terms $M_1, M_2, M_3 \in \boldsymbol{\lambda_{\rightarrow}}$**-Curry** *(or* $M_1, M_2, M_3 \in \boldsymbol{\lambda_{\rightarrow}}$**-Church***)*

  $$\textit{if} \quad M_1 \twoheadrightarrow_\beta M_2 \quad \textit{and} \quad M_1 \twoheadrightarrow_\beta M_3$$

  then there is $M_4 \in \boldsymbol{\lambda_{\rightarrow}}$**-Curry** *(or, resp.,* $M_4 \in \boldsymbol{\lambda_{\rightarrow}}$**-Church***) such that*

  $$M_2 \twoheadrightarrow_\beta M_4 \quad \textit{and} \quad M_3 \twoheadrightarrow_\beta M_4.$$

  A proof based on *Newman's Lemma* is in [LCHI, Sect 3.6, pp 71-72] .

Expressibility

Below are two results on the expressive power of the simply-typed λ-calculus.

For these, we do not need to separate between **$\boldsymbol{\lambda_{\rightarrow}}$-Curry** and **$\boldsymbol{\lambda_{\rightarrow}}$-Church** and we write $\boldsymbol{\lambda_{\rightarrow}}$ to refer indistinguishably to both.

- **THEOREM** (Complexity of β-Equality)
  It is decidable whether two terms $M, N \in \boldsymbol{\lambda_{\rightarrow}}$ are β-equal, but the complexity is non-elementary .

- **THEOREM** (Definability in $\boldsymbol{\lambda_{\rightarrow}}$)
  The $\boldsymbol{\lambda_{\rightarrow}}$-definable functions are exactly the extended polynomials .

- Details and supporting definitions for both theorems are in [LCHI, Sect 3.7, pp 72-73] .

(THIS SLIDE INTENTIONALLY LEFT BLANK)