Type Theory

Lecture 1b: Simple Type Theory

H. Geuvers

Radboud University
Nijmegen, NL

OPLSS 2026 Summer 2026

Outline

Simple type theory

Theory of simple types: $\lambda\!\!\rightarrow$ or simple type theory, STT. Just arrow types

$$\mathsf{Typ} := \mathsf{TVar} \mid (\mathsf{Typ} \rightarrow \mathsf{Typ})$$

- Examples: $(\alpha\!\rightarrow\!\beta)\!\rightarrow\!\alpha$, $(\alpha\!\rightarrow\!\beta)\!\rightarrow\!((\beta\!\rightarrow\!\gamma)\!\rightarrow\!(\alpha\!\rightarrow\!\gamma))$
- Brackets associate to the right and outside brackets are omitted:
  $(\alpha\!\rightarrow\!\beta)\!\rightarrow\!(\beta\!\rightarrow\!\gamma)\!\rightarrow\!\alpha\!\rightarrow\!\gamma$
- Types are denoted by $A, B, \ldots$.

Simple type theory à la Church

Formulation with contexts to declare the free variables:

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$$

is a context, usually denoted by Γ.

Definition The derivation rules of $\lambda\!\!\rightarrow$ (à la Church) are as follows. This gives an inductive definition of the derivable judgments $\Gamma \vdash M : A$.

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A{\rightarrow}B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B} \qquad \frac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash \lambda x{:}A.P : A{\rightarrow}B}$$

$\Gamma \vdash_{\lambda\rightarrow} M : A$ if there is a derivation using these rules with conclusion $\Gamma \vdash M : A$

Examples of typable terms

$$\begin{array}{rcl} \vdash \lambda x : A.\lambda y : B.x & : & A \to B \to A \\ \vdash \lambda x : A \to B.\lambda y : B \to C.\lambda z : A.y\,(x\,z) & : & (A \to B) \to (B \to C) \to A \to C \\ \vdash \lambda x : A.\lambda y : (B \to A) \to A.y(\lambda z : B.x) & : & A \to ((B \to A) \to A) \to A \end{array}$$

Not for every type there is a closed term of that type:

$(\alpha \to \alpha) \to \alpha$ is not inhabited

That is: there is no term M such that

$$\vdash M : (\alpha \to \alpha) \to \alpha.$$

Using the typing rules bottom up

If one tries to solve a typing problem or an inhabitation problem, one reads the rules bottom up.

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A{\to}B \;\; \Gamma \vdash N : A}{\Gamma \vdash MN : B} \qquad \frac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash \lambda x{:}A.P : A{\to}B}$$

$\Gamma \vdash M : A?$	Type Checking Problem	TCP
$\Gamma \vdash M : ?$	Type Synthesis Problem	TSP
$\Gamma \vdash ? : A$	Type Inhabitation Problem	TIP

Typed Terms versus Type Assignment

- With typed terms also called typing à la Church, we have terms with type information in the λ-abstraction

$$\lambda x : A.x \ : \ A \rightarrow A$$

  - Terms have unique types,
  - The type is directly computed from the type info in the variables.

- With typed assignment also called typing à la Curry, we assign types to untyped λ-terms

$$\lambda x.x : A \rightarrow A$$

  - Terms do not have unique types,
  - A principal type can be computed using unification.

Simple Type Theory à la Church and à la Curry

$\lambda\!\!\rightarrow$ (à la Church):

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A{\rightarrow}B \;\; \Gamma \vdash N : A}{\Gamma \vdash MN : B} \qquad \frac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash \lambda x{:}A.P : A{\rightarrow}B}$$

$\lambda\!\!\rightarrow$ (à la Curry):

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A{\rightarrow}B \;\; \Gamma \vdash N : A}{\Gamma \vdash MN : B} \qquad \frac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash \lambda x.P : A{\rightarrow}B}$$

Church vs. Curry typing

- The Curry formulation is especially interesting for (functional) programming:
  - you want to write as little type information as possible;
  - let the compiler infer the types for you.
- The Church formulation is especially interesting for proof checking:
  - terms are created interactively;
  - if one adds dependent types, the type structure is so intricate that type inference is undecidable (if you start from an untyped term).

  [This lecture]

Formulas-as-Types (Curry, Howard)

There are two readings of a judgement $M : A$

1. term as algorithm/program, type as specification:
   M is a function of type A
2. type as a proposition, term as its proof:
   M is a proof of the proposition A

- There is a one-to-one correspondence (isomorphism!):

  typable terms in $\lambda\!\!\rightarrow \simeq$ derivations in minimal proposition logic (only implication)

- $x_1 : B_1, x_2 : B_2, \ldots, x_n : B_n \vdash M : A$ can be read as
  M is a proof of A from the assumptions $B_1, B_2, \ldots, B_n$.

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{[A\to B\to C]^3 \quad [A]^1}{B\to C} \qquad \dfrac{[A\to B]^2 \quad [A]^1}{B}}{C}\,1}{(A\to B)\to A\to C}\,2}{(A\to B\to C)\to(A\to B)\to A\to C}\,3 \quad \simeq$$

$$\lambda x{:}A\to B\to C.\lambda y{:}A\to B.\lambda z{:}A.x\ z\ (y\ z)$$
$$: (A\to B\to C)\to(A\to B)\to A\to C$$

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{[x : A{\to}B{\to}C]^3 \;\; [z : A]^1}{x\,z : B{\to}C} \qquad \dfrac{[y : A{\to}B]^2 \;\; [z : A]^1}{y\,z : B}}{x\,z\,(y\,z) : C}\,1}{\lambda z{:}A.x\,z\,(y\,z) : A{\to}C}\,2}{\lambda y{:}A{\to}B.\lambda z{:}A.x\,z\,(y\,z) : (A{\to}B){\to}A{\to}C}\,3$$

$$\lambda x{:}A{\to}B{\to}C.\lambda y{:}A{\to}B.\lambda z{:}A.x\,z\,(y\,z) : (A{\to}B{\to}C){\to}(A{\to}B){\to}A{\to}C$$

Exercise: Give the derivation that corresponds to

$$\lambda x{:}C{\to}E.\lambda y{:}(C{\to}E){\to}E.y(\lambda z.y\,x) :$$
$$(C{\to}E){\to}((C{\to}E){\to}E){\to}E$$

Fitch style derivations

The Fitch style (also: flag style) presentation of $\lambda\!\rightarrow$.

1	$x : A$	
2	$\ldots$	
3	$\ldots$	
4	$M : B$	
5	$\lambda x{:}A.M : A \rightarrow B$	abs, 1, 4

abs-rule

1	$\ldots$	
2	$\ldots$	
3	$M : A \rightarrow B$	
4	$\ldots$	
5	$\ldots$	
6	$N : A$	
7	$\ldots$	
8	$M\,N : B$	app, 3, 6

app-rule

Same example in Fitch style

1		$x : A \to B \to C$
2	|	$y : A \to B$
3	| |	$z : A$
4	| |	$x\ z : B \to C$
5	| |	$y\ z : B$
6	| |	$x\ z\ (y\ z) : C$
7	|	$\lambda z{:}A.x\ z\ (y\ z) : A \to C$
8		$\lambda y{:}A \to B.\lambda z{:}A.x\ z\ (y\ z) : (A \to B) \to A \to C$
9		$\lambda x{:}A \to B \to C.\lambda y{:}A \to B.\lambda z{:}A.x\ z\ (y\ z) : (A \to B \to C) \to (A \to B) \to A \to C$

Computation = Detour-elimination

- β-reduction: $(\lambda x{:}A.M)P \to_\beta M[x := P]$

Detour-elimination in minimal logic = β-reduction in $\lambda\!\to$.

$$\dfrac{\dfrac{\begin{array}{c}[A]^1\\ \mathcal{D}_1\\ B\end{array}}{A\to B}\,1 \qquad \dfrac{\mathcal{D}_2}{A}}{B} \quad\longrightarrow\quad \begin{array}{c}\mathcal{D}_2\\ A\\ \mathcal{D}_1\\ B\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}[x:A]^1\\ \mathcal{D}_1\\ M:B\end{array}}{\lambda x{:}A.M : A\to B}\,1 \qquad \dfrac{\mathcal{D}_2}{P:A}}{(\lambda x{:}A.M)P : B} \quad\longrightarrow_\beta\quad \begin{array}{c}\mathcal{D}_2\\ P:A\\ \mathcal{D}_1\\ M[x:=P]:B\end{array}$$

Example of a detour

Proof of $A \to A \to B, (A \to B) \to A \vdash B$ with a detour.

$$\dfrac{\dfrac{\dfrac{A \to A \to B \quad [A]^1}{A \to B} \qquad [A]^1}{B} \qquad (A \to B) \to A}{A} \qquad \dfrac{\dfrac{A \to A \to B \quad [A]^1}{A \to B} \qquad [A]^1}{B}}{A \to B}}{B}$$

It contains a detour: a $\to$-i directly followed by an $\to$-e.

Example proof with term information

$$\dfrac{\dfrac{\dfrac{p : A{\to}A{\to}B \quad [y : A]^1}{p\,y : A{\to}B} \qquad [y : A]^1}{p\,y\,y : B}}{\lambda y{:}A.p\,y\,y : A{\to}B} \qquad \dfrac{q : (A{\to}B){\to}A \qquad \dfrac{\dfrac{p : A{\to}A{\to}B \quad [x : A]^1}{p\,x : A{\to}B} \qquad [x : A]^1}{p\,x\,x : B}}{\lambda x{:}A.p\,x\,x : A{\to}B}}{(\lambda y{:}A.p\,y\,y)(q(\lambda x{:}A.p\,x\,x)) : B}$$

Term contains a β-redex: $(\lambda x{:}A.p\,x\,x)\,(q(\lambda x{:}A.p\,x\,x))$

Typical problems one would like to have an algorithm for

$\Gamma \vdash M : A?$	Type Checking Problem	TCP
$\Gamma \vdash M : ?$	Type Synthesis Problem	TSP
$\Gamma \vdash ? : A$	Type Inhabitation Problem	TIP

For $\lambda\!\!\rightarrow$, all these problems are decidable,
both for the Curry style and for the Church style presentation.
For Curry style TSP is usually called the Type Assignment Problem, TAP.

- TCP and TSP are (as usual) equivalent: To solve $M\,N : A$, one has to solve $N :?$ (and if this gives answer B, solve $M : B\!\rightarrow\!A$).
- For Curry systems, TCP and TSP soon become undecidable beyond $\lambda\!\!\rightarrow$.
- TIP is undecidable for most extensions of $\lambda\!\!\rightarrow$, as it corresponds to provability in some logic. For $\lambda\!\!\rightarrow$, TIP is PSPACE complete.

Meta-theory for $\lambda\!\rightarrow$

- For $\lambda\!\rightarrow$ à la Church: Uniqueness of types
  If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A = B$.
- Subject Reduction or Closure
  If $\Gamma \vdash M : A$ and $M \rightarrow_{\beta\eta} N$, then $\Gamma \vdash N : A$.
- Strong Normalization
  If $\Gamma \vdash M : A$, then all $\beta\eta$-reductions from M terminate.

Extension with other connectives

Adding product types $\times$ to $\lambda\!\!\rightarrow$. (Proposition logic with conjunction $\wedge$.)

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_1 M : A} \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_2 M : B} \qquad \frac{\Gamma \vdash P : A \;\; \Gamma \vdash Q : B}{\Gamma \vdash \langle P, Q \rangle : A \times B}$$

With reduction rules

$$\begin{aligned} \pi_1 \langle P, Q \rangle &\rightarrow P \\ \pi_2 \langle P, Q \rangle &\rightarrow Q \end{aligned}$$

Similar rules can be given for sum-types $A + B$, corresponding to disjunction $A \vee B$.

Questions?