Lambda-Calculus and Type Theory

ISR 2024
Obergurgl, Austria

Herman Geuvers & Niels van der Weide

Radboud University Nijmegen, The Netherlands

Lecture 2

Simple Type theory and Formulas-as-Types and Proofs-as-terms

Simple Type Theory

Simplest system: $\lambda\to$ or simple type theory, STT. Just arrow types

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

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

Terms:

- typed variables $x_1^\sigma, x_2^\sigma, \ldots$, countably many for every σ.
- application: if $M : \sigma\to\tau$ and $N : \sigma$, then $(MN) : \tau$
- abstraction: if $P : \tau$, then $(\lambda x^\sigma.P) : \sigma\to\tau$

Examples of simply typed terms

$$\begin{array}{rcl} \lambda x^{\sigma}.\lambda y^{\tau}.x & : & \sigma\to\tau\to\sigma \\ \lambda x^{\alpha\to\beta}.\lambda y^{\beta\to\gamma}.\lambda z^{\alpha}.y(xz) & : & (\alpha\to\beta)\to(\beta\to\gamma)\to\alpha\to\gamma \\ \lambda x^{\alpha}.\lambda y^{(\beta\to\alpha)\to\alpha}.y(\lambda z^{\beta}.x) & : & \alpha\to((\beta\to\alpha)\to\alpha)\to\alpha \end{array}$$

For every type there is a term of that type:

$$x^{\sigma} : \sigma$$

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 closed term of type $(\alpha\to\alpha)\to\alpha$.]

Church' simple type theory

Church formulation of simple type theory: terms with type information.

Inductive definition of the terms:

- typed variables $x_1^\sigma, x_2^\sigma, \ldots$, countably many for every σ.
- application: if $M : \sigma \to \tau$ and $N : \sigma$, then $(MN) : \tau$
- abstraction: if $P : \tau$, then $(\lambda x^\sigma . P) : \sigma \to \tau$

Alternative: Inductive definition of the terms in rule form:

$$\frac{}{x^\sigma : \sigma} \qquad \frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau} \qquad \frac{P : \tau}{\lambda x^\sigma . P : \sigma \to \tau}$$

Advantage: We also have a derivation tree, a proof of the fact that the term has that type.

We can reason over derivations.

Simple type theory à la Church with contexts

Formulation with contexts to declare the free variables:

$$x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_n : \sigma_n$$

is a context, usually denoted by Γ.
Derivation rules of $\lambda\!\to$ (à la Church):

$$\frac{x{:}\sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash M : \sigma\to\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \qquad \frac{\Gamma, x{:}\sigma \vdash P : \tau}{\Gamma \vdash \lambda x{:}\sigma.P : \sigma\to\tau}$$

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

Reading the typing rules top down

Inductive definition of the "derivable judgments"

$$\frac{x{:}\sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash M : \sigma{\to}\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \qquad \frac{\Gamma, x{:}\sigma \vdash P : \tau}{\Gamma \vdash \lambda x{:}\sigma.P : \sigma{\to}\tau}$$

Deriving

$$\vdash \lambda x{:}\alpha.\lambda y{:}(\beta{\to}\alpha){\to}\alpha.y(\lambda z{:}\beta.x) : \alpha{\to}((\beta{\to}\alpha){\to}\alpha){\to}\alpha$$

Reading the typing rules bottom up

Trying to solve a typing problem / an inhabitation problem

$$\frac{x{:}\sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash M : \sigma{\to}\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \qquad \frac{\Gamma, x{:}\sigma \vdash P : \tau}{\Gamma \vdash \lambda x{:}\sigma.P : \sigma{\to}\tau}$$

Formulas-as-Types (Curry, Howard)

There are two readings of a judgement $M : \sigma$

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

- There is a one-to-one correspondence:

  typable terms in $\lambda\!\rightarrow$ $\simeq$ derivations in minimal proposition logic

- $x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n \vdash M : \sigma$ can be read as
  M is a proof of σ from the assumptions $\tau_1, \tau_2, \ldots, \tau_n$.

Example

$$\dfrac{\dfrac{\dfrac{[\alpha\to\beta\to\gamma]^3 \quad [\alpha]^1}{\beta\to\gamma} \qquad \dfrac{[\alpha\to\beta]^2 \quad [\alpha]^1}{\beta}}{\dfrac{\gamma}{\alpha\to\gamma}\,1}}{\dfrac{(\alpha\to\beta)\to\alpha\to\gamma}{(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma}\,3}\,2$$

$\simeq$

$\lambda x{:}\alpha\to\beta\to\gamma.\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.xz(yz)$
$: (\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma$

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[x : \alpha\to\beta\to\gamma]^3 \quad [z : \alpha]^1}{xz : \beta\to\gamma} \qquad \dfrac{[y : \alpha\to\beta]^2 \quad [z : \alpha]^1}{yz : \beta}}{xz(yz) : \gamma}}{\lambda z{:}\alpha.xz(yz) : \alpha\to\gamma}\,1}{\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.xz(yz) : (\alpha\to\beta)\to\alpha\to\gamma}\,2}{\lambda x{:}\alpha\to\beta\to\gamma.\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.xz(yz) : (\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma}\,3$$

Exercise: Give the derivation that corresponds to

$$\lambda x{:}\gamma\to\varepsilon.\lambda y{:}(\gamma\to\varepsilon)\to\varepsilon.y(\lambda z.y\,x) : (\gamma\to\varepsilon)\to((\gamma\to\varepsilon)\to\varepsilon)\to\varepsilon$$

Flag style deductions

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

1	$\quad x : \sigma$	
2	$\quad \ldots$	
3	$\quad \ldots$	
4	$\quad M : \tau$	
5	$\lambda x{:}\sigma.M : \sigma \to \tau$	abs, 1, 4

abs-rule

1	$\ldots$	
2	$\ldots$	
3	$M : \sigma \to \tau$	
4	$\ldots$	
5	$\ldots$	
6	$N : \sigma$	
7	$\ldots$	
8	$M\,N : \tau$	app, 3, 6

app-rule

Example

1 $x : \alpha\to\beta\to\gamma$

2 $y : \alpha\to\beta$

3 $z : \alpha$

4 $x\,z : \beta\to\gamma$

5 $y\,z : \beta$

6 $x\,z(y\,z) : \gamma$

7 $\lambda z{:}\alpha.x\,z(y\,z) : \alpha\to\gamma$

8 $\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.x\,z(y\,z) : (\alpha\to\beta)\to\alpha\to\gamma$

9 $\lambda x{:}\alpha\to\beta\to\gamma.\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.x\,z(y\,z) : (\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma$

Computation

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

Cut-elimination

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

$$\dfrac{\dfrac{\begin{matrix}[\sigma]^1 \\ \mathcal{D}_1 \\ \tau\end{matrix}}{\sigma\to\tau}\,1 \qquad \dfrac{\mathcal{D}_2}{\sigma}}{\tau} \quad\longrightarrow\quad \begin{matrix}\mathcal{D}_2 \\ \sigma \\ \mathcal{D}_1 \\ \tau\end{matrix}$$

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

Example

Proof of $A\to A\to B, (A\to B)\to A \vdash B$

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

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

Example ctd

Proof of $A \to A \to B, (A \to B) \to A \vdash B$ after reduction

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

Example ctd

Proof of $A \to A \to B, (A \to B) \to A \vdash B$ with term information.

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

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

Example ctd

Reduced proof of $A \to A \to B, (A \to B) \to A \vdash B$ with term info.

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

Extension with other connectives

STT with product types $\times$ (proposition logic with conjunction $\wedge$)
Extend the types with $\sigma \times \tau$. Extend the terms with pairing and projection.

$$\frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \pi_1 M : \sigma} \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \pi_2 M : \tau} \qquad \frac{\Gamma \vdash P : \sigma \quad \Gamma \vdash Q : \tau}{\Gamma \vdash \langle P, Q \rangle : \sigma \times \tau}$$

With reduction rules

$$\begin{array}{lcl} \pi_1 \langle P, Q \rangle & \to & P \\ \pi_2 \langle P, Q \rangle & \to & Q \end{array}$$

Why do we want types?

- Types give a (partial) specification
- Typed terms can't go wrong (Milner)
  Subject Reduction property: If $M : A$ and $M \twoheadrightarrow_\beta N$, then $N : A$.
- Typed terms always terminate
- The type checking algorithm detects (simple) mistakes

But:

- The compiler should compute the type information for us!
  (Why would the programmer have to type all that?)
- This is called a type assignment system, or also typing à la Curry:
- For M an untyped term, the type system assigns a type σ to M (or not)

Simple Type Theory à la Church and à la Curry

$\lambda\!\rightarrow$ (à la Church):

$$\frac{x{:}\sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash M : \sigma{\rightarrow}\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \qquad \frac{\Gamma, x{:}\sigma \vdash P : \tau}{\Gamma \vdash \lambda x{:}\sigma.P : \sigma{\rightarrow}\tau}$$

$\lambda\!\rightarrow$ (à la Curry):

$$\frac{x{:}\sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash M : \sigma{\rightarrow}\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \qquad \frac{\Gamma, x{:}\sigma \vdash P : \tau}{\Gamma \vdash \lambda x.P : \sigma{\rightarrow}\tau}$$

Typed Terms versus Type Assignment:

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

$$\lambda x : \alpha . x : \alpha \rightarrow \alpha$$

  As a consequence:

  - 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

$$\lambda x . x : \alpha \rightarrow \alpha$$

  As a consequence:

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

Examples

- Typed Terms:

$$\lambda x : \alpha . \lambda y : (\beta \to \alpha) \to \alpha . y (\lambda z : \beta . x)$$

  has only the type $\alpha \to ((\beta \to \alpha) \to \alpha) \to \alpha$

- Type Assignment:

$$\lambda x . \lambda y . y (\lambda z . x)$$

  can be assigned the types
  - $\alpha \to ((\beta \to \alpha) \to \alpha) \to \alpha$
  - $(\alpha \to \alpha) \to ((\beta \to \alpha \to \alpha) \to \gamma) \to \gamma$
  - ...

  with $\alpha \to ((\beta \to \alpha) \to \gamma) \to \gamma$ being the principal type

Example derivation

$\lambda x.\lambda y.y(\lambda z.x)$ can be assigned the type
$(\alpha{\to}\alpha){\to}((\beta{\to}\alpha{\to}\alpha){\to}\gamma){\to}\gamma$ in $\lambda{\to}$ a la Curry.

Connection between Church and Curry typed $\lambda\!\to$

Definition The erasure map $|-|$ from $\lambda\!\to$ à la Church to $\lambda\!\to$ à la Curry is defined by erasing all type information.

$$\begin{aligned} |x| &:= x \\ |M\,N| &:= |M|\,|N| \\ |\lambda x:\sigma.M| &:= \lambda x.|M| \end{aligned}$$

So, e.g.

$$|\lambda x:\alpha.\lambda y:(\beta\to\alpha)\to\alpha.y(\lambda z:\beta.x)| = \lambda x.\lambda y.y(\lambda z.x)$$

Theorem If $\Gamma \vdash M:\sigma$ in $\lambda\!\to$ à la Church, then $\Gamma \vdash |M|:\sigma$ in $\lambda\!\to$ à la Curry.

Theorem If $\Gamma \vdash P:\sigma$ in $\lambda\!\to$ à la Curry, then there is an M such that $|M| \equiv P$ and $\Gamma \vdash M:\sigma$ in $\lambda\!\to$ à la Church.

Connection between Church and Curry typed $\lambda\!\!\to$

Definition The erasure map $|-|$ from $\lambda\!\!\to$ à la Church to $\lambda\!\!\to$ à la Curry is defined by erasing all type information.

$$
\begin{array}{rcl}
|x| & := & x \\
|M\,N| & := & |M|\,|N| \\
|\lambda x:\sigma.M| & := & \lambda x.|M|
\end{array}
$$

Theorem If $\Gamma \vdash P : \sigma$ in $\lambda\!\!\to$ à la Curry, then there is an M such that $|M| \equiv P$ and $\Gamma \vdash M : \sigma$ in $\lambda\!\!\to$ à la Church.
Proof: by induction on derivations.

$$
\frac{x{:}\sigma \in \Gamma}{\Gamma \vdash x : \sigma}
\qquad
\frac{\Gamma \vdash M : \sigma{\to}\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}
\qquad
\frac{\Gamma, x{:}\sigma \vdash P : \tau}{\Gamma \vdash \lambda x.P : \sigma{\to}\tau}
$$