Lambda-Calculus and Type Theory

ISR 2024
Obergurgl, Austria

Herman Geuvers & Niels van der Weide

Radboud University Nijmegen, The Netherlands

Lecture 8

Meta theory of type systems and type checking algorithm

What do we want to prove about type systems?

The Meta Theory of type theory. Central result that we want:

- Decidability of Typing, which comes in two forms:
  - Type Checking: Given Γ, M, A, is it the case that $\Gamma \vdash M : A$?
  - Type Synthesis: Given Γ, M, can we compute an A such that $\Gamma \vdash M : A$ and otherwise decide that there is no such A?

These problems are equivalent.

Meta theory of type systems

More basic properties we want (and some are needed for Decidability of typing)

- Subject Reduction (or Closure, or Preservation of typing)
  If $\Gamma \vdash M : A$ and $M \to_\beta N$, then $\Gamma \vdash N : A$
- Church-Rosser (next lecture: for β-reduction)
  If $M \twoheadrightarrow_\beta P_1$ and $M \twoheadrightarrow_\beta P_2$, then $\exists Q(P_1 \twoheadrightarrow_\beta Q \wedge P_2 \twoheadrightarrow_\beta Q)$.
- Normalization (two lectures ahead)
  - Weak Normalization, WN, a term M is WN if $\exists P \in \mathrm{NF}(M \twoheadrightarrow_\beta P)$.
    NB. NF is the set of normal forms, terms that cannot be reduced.
  - Strong Normalization, SN, a term M is SN if
    $\neg\exists(P_i)_{i\in\mathbb{N}}(M = P_0 \to_\beta P_1 \to_\beta P_2 \to_\beta \ldots)$.
- Progress
  If $\vdash M : A$, then either $\exists P(M \to_\beta P)$ or M is a value

Subject Reduction

LEMMA If $\Gamma \vdash M : A$ and $M \to_\beta N$, then $\Gamma \vdash N : A$

PROOF By induction on M. The base case is when $M = (\lambda x{:}B.P)Q \to_\beta P[x := Q] = N$. This is also the only interesting case. It goes roughly as follows

$$\frac{\dfrac{\Gamma, x{:}B \vdash P : C}{\Gamma \vdash \lambda x{:}B.P : \Pi x{:}B.C} \qquad \Gamma \vdash Q : B}{\Gamma \vdash (\lambda x{:}B.P)Q : C[x := Q]}$$

And we need to prove that $\Gamma \vdash P[x := Q] : C[x := Q]$.

This is proved by proving a Substitution Lemma:
SUBSTITUTION LEMMA: If $\Gamma, x : B \vdash P : C$ and $\Gamma \vdash Q : B$, then $\Gamma \vdash P[x := Q] : C[x := Q]$.

Substitution Lemma

SUBSTITUTION LEMMA: If $\Gamma, x : B \vdash P : C$ and $\Gamma \vdash Q : B$, then
$\Gamma \vdash P[x := Q] : C[x := Q]$.
PROOF By induction on the derivation of $\Gamma, x : B \vdash P : C$.

But that doesn't work: one has to prove something slightly more general.
SUBSTITUTION LEMMA: If $\Gamma, x : B\, \Delta \vdash P : C$ and $\Gamma \vdash Q : B$, then
$\Gamma, \Delta[x := Q] \vdash P[x := Q] : C[x := Q]$.
PROOF By induction on the derivation of $\Gamma, x : B, \Delta \vdash P : C$.

Type Checking for λP

Define algorithms $\mathrm{Ok}(-)$ and $\mathrm{Type}_{\_}(-)$ simultaneously:

- $\mathrm{Ok}(-)$ takes a context and returns 'true' or 'false'
- $\mathrm{Type}_{\_}(-)$ takes a context and a term and returns a term or 'false'.

The type synthesis algorithm $\mathrm{Type}_{\_}(-)$ is sound if (for all Γ and M)

$$\mathrm{Type}_{\Gamma}(M) = A \quad \Longrightarrow \quad \Gamma \vdash M : A$$

The type synthesis algorithm $\mathrm{Type}_{\_}(-)$ is complete if (for all Γ, M and A)

$$\Gamma \vdash M : A \quad \Longrightarrow \quad \mathrm{Type}_{\Gamma}(M) =_{\beta} A$$

- A proof assistant like Coq is based on a type checking algorithm.
- The type checking algorithm is the trusted kernel of Coq

$$
\begin{array}{rcl}
\mathrm{Ok}(<>) & = & \text{'true'} \\
\mathrm{Ok}(\Gamma, x{:}A) & = & \mathrm{Type}_\Gamma(A) \in \{*, \Box\}, \\
\mathrm{Type}_\Gamma(x) & = & \text{if } \mathrm{Ok}(\Gamma) \text{ and } x{:}A \in \Gamma \text{ then } A \text{ else 'false'}, \\
\mathrm{Type}_\Gamma(*) & = & \text{if } \mathrm{Ok}(\Gamma) \text{then } \Box \text{ else 'false'}, \\
\mathrm{Type}_\Gamma(MN) & = & \text{if } \mathrm{Type}_\Gamma(M) = C \text{ and } \mathrm{Type}_\Gamma(N) = D \\
& & \quad \text{then} \quad \text{if } C \twoheadrightarrow_\beta \Pi x{:}A.B \text{ and } A =_\beta D \\
& & \qquad\qquad \text{then } B[x := N] \text{ else 'false'} \\
& & \quad \text{else} \quad \text{'false'},
\end{array}
$$

$$
\begin{aligned}
\mathrm{Type}_{\Gamma}(\lambda x{:}A.M) \quad &= \quad \text{if } \mathrm{Type}_{\Gamma,x:A}(M) = B \\
&\qquad \text{then} \qquad \text{if } \mathrm{Type}_{\Gamma}(\Pi x{:}A.B) \in \{*, \Box\} \\
&\qquad\qquad\qquad \text{then } \Pi x{:}A.B \text{ else `false'} \\
&\qquad \text{else `false'}, \\
\mathrm{Type}_{\Gamma}(\Pi x{:}A.B) \quad &= \quad \text{if } \mathrm{Type}_{\Gamma}(A) = * \text{ and } \mathrm{Type}_{\Gamma,x:A}(B) = s \\
&\qquad \text{then } s \text{ else `false'}
\end{aligned}
$$

Soundness and Completeness

Soundness

$$\mathrm{Type}_\Gamma(M) = A \quad \Longrightarrow \quad \Gamma \vdash M : A$$

Completeness

$$\Gamma \vdash M : A \quad \Longrightarrow \quad \mathrm{Type}_\Gamma(M) =_\beta A$$

As a consequence:

$$\mathrm{Type}_\Gamma(M) = \text{'false'} \quad \Longrightarrow \quad M \text{ is not typable in } \Gamma$$

NB 1. Completeness implies that Type terminates on all well-typed terms. We want that Type terminates on all pseudo terms.
NB 2. Completeness only makes sense if we have uniqueness of types
(Otherwise: let $\mathrm{Type}_{\_}(-)$ generate a set of possible types)

Termination

We want $\mathrm{Type}_{\_}(-)$ to terminate on all inputs.
Interesting cases: λ-abstraction and application:

$$\mathrm{Type}_\Gamma(\lambda x{:}A.M) \quad = \quad \text{if } \mathrm{Type}_{\Gamma,x:A}(M) = B$$
$$\text{then} \quad \text{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{*, \Box\} \text{ then } \Pi x{:}A.B \text{ else 'false'}$$
$$\text{else 'false'},$$

! Recursive call is not on a smaller term!
Replace the side condition

$$\text{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{*, \Box\}$$

by

$$\text{if } \mathrm{Type}_\Gamma(A) \in \{*\}$$

Termination

We want $\mathrm{Type}_{\_}(-)$ to terminate on all inputs.
Interesting cases: λ-abstraction and application:

$$\mathrm{Type}_\Gamma(MN) \;=\; \text{if } \mathrm{Type}_\Gamma(M) = C \text{ and } \mathrm{Type}_\Gamma(N) = D$$
$$\text{then} \quad \text{if } C \twoheadrightarrow_\beta \Pi x{:}A.B \text{ and } A =_\beta D \text{ then } B[x := N] \text{ else `false'}$$
$$\text{else} \quad \text{`false'},$$

! Need to decide β-reduction and β-equality!
For this case, termination follows from:

- Soundness of Type and
- Decidability of equality on well-typed terms.

This decidability of equality follows from SN (strong normalization) and CR (Church-Rosser property) – to be discussed in later lectures.