Proofs as Programs Summer School
Eugene Oregon June - July 2002

Type Systems

Herman Geuvers
Nijmegen University, NL

Lecture 1: Simple Type Theory

Simplest system: $\lambda\!\to$ 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:

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

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

Typed Terms versus Type Assignment:

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

  - $\alpha\to((\beta\to\alpha)\to\alpha)\to\alpha$
  - $\alpha\to((\beta\to\alpha)\to\gamma)\to\gamma$
  - $(\alpha\to\alpha)\to((\beta\to\alpha\to\alpha)\to\gamma)\to\gamma$
  - ...

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

Example of computing a principle type:

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

Example of computing a principle type:

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

1. Assign a type variable to all variables: $x : \alpha, y : \beta, z : \gamma$.

Example of computing a principle type:

$$\lambda x^{\alpha}.\lambda y^{\beta}.y^{\beta}(\lambda z^{\gamma}.y^{\beta}x^{\alpha})$$

1. Assign a type variable to all variables: $x : \alpha, y : \beta, z : \gamma$.
2. Generate equations between types, necessary for the term to be typable:

Example of computing a principle type:

$$\lambda x^{\alpha}.\lambda y^{\beta}.y^{\beta}(\lambda z^{\gamma}.y^{\beta}x^{\alpha})$$

1. Assign a type variable to all variables: $x:\alpha, y:\beta, z:\gamma$.
2. Generate equations between types, necessary for the term to be typable:
   - $\beta = \alpha \to \delta$

Example of computing a principle type:

$$\lambda x^{\alpha}.\lambda y^{\beta}.y^{\beta}(\lambda z^{\gamma}.y^{\beta}x^{\alpha})$$

1. Assign a type variable to all variables: $x : \alpha, y : \beta, z : \gamma$.
2. Generate equations between types, necessary for the term to be typable:
   - $\beta = \alpha{\to}\delta$
   - $\beta = (\gamma{\to}\delta){\to}\epsilon$

Example of computing a principle type:

$$\lambda x^{\alpha}.\lambda y^{\beta}.y^{\beta}(\lambda z^{\gamma}.y^{\beta}x^{\alpha})$$

1. Assign a type variable to all variables: $x : \alpha, y : \beta, z : \gamma$.
2. Generate equations between types, necessary for the term to be typable:
   - $\beta = \alpha \to \delta$
   - $\beta = (\gamma \to \delta) \to \epsilon$
3. Find a most general substitution for the type variables that solves the equations:

$$\alpha := \gamma \to \delta, \beta := (\gamma \to \delta) \to \epsilon, \delta := \epsilon$$

Example of computing a principle type:

$$\lambda x^{\alpha}.\lambda y^{\beta}.y^{\beta}(\lambda z^{\gamma}.y^{\beta}x^{\alpha})$$

1. Assign a type variable to all variables: $x:\alpha, y:\beta, z:\gamma$.
2. Generate equations between types, necessary for the term to be typable:
   - $\beta = \alpha \rightarrow \delta$
   - $\beta = (\gamma \rightarrow \delta) \rightarrow \epsilon$
3. Find a most general substitution for the type variables that solves the equation:

$$\alpha := \gamma \rightarrow \delta, \beta := (\gamma \rightarrow \delta) \rightarrow \epsilon, \delta := \epsilon$$

4. The principal type of $\lambda x.\lambda y.y(\lambda z.yx)$ is now

$$(\gamma \rightarrow \epsilon) \rightarrow ((\gamma \rightarrow \epsilon) \rightarrow \epsilon) \rightarrow \epsilon$$

Typical problems one would like to have an algorithm for:

$M : \sigma?$	Type Checking Problem	TCP
$M : ?$	Type Synthesis Problem	TSP
$? : \sigma$	Type Inhabitation Problem (by a closed term)	TIP

Typical problems one would like to have an algorithm for:

$M : \sigma?$	Type Checking Problem	TCP
$M : ?$	Type Synthesis Problem	TSP
$? : \sigma$	Type Inhabitation Problem (by a closed term)	TIP

For $\lambda\!\!\rightarrow$, all these problems are decidable,
both for the Curry style as for the Church style presentation.

Typical problems one would like to have an algorithm for:

$M : \sigma?$ Type Checking Problem		TCP
$M : ?$ Type Synthesis Problem		TSP
$? : \sigma$ Type Inhabitation Problem (by a closed term)		TIP

For $\lambda\!\rightarrow$, all these problems are decidable,
both for the Curry style as for the Church style presentation.
Remarks:

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

From now on we focus on the Church formulation of simple type theory:

terms with type information.

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 $\lambda{\to}$:

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

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

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

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 between
  - typable terms in $\lambda\!\to$
  - derivations in minimal proposition logic
- The judgement

$$x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n \vdash M : \sigma$$

can be read as

M is a proof of σ from the assumptions $\tau_1, \tau_2, \ldots, \tau_n$.

Example

$$
\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{[\alpha\to\beta\to\gamma]^3 \quad [\alpha]^1}{\beta\to\gamma} \qquad \cfrac{[\alpha\to\beta]^2 \quad [\alpha]^1}{\beta}}{\gamma}}{\alpha\to\gamma}\, 1}{(\alpha\to\beta)\to\alpha\to\gamma}\, 2}{(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma}\, 3
\quad \simeq \quad
\begin{array}{l}
\lambda x{:}\alpha\to\beta\to\gamma.\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha..xz(yz) \\
\quad : (\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma
\end{array}
$$

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\alpha\to\beta\to\gamma]^3 \quad [\alpha]^1}{\beta\to\gamma} \qquad \dfrac{[\alpha\to\beta]^2 \quad [\alpha]^1}{\beta}}{\gamma}}{\alpha\to\gamma}\;1}{(\alpha\to\beta)\to\alpha\to\gamma}\;2}{(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma}\;3 \quad \simeq \quad \begin{array}{c} \lambda x{:}\alpha\to\beta\to\gamma.\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.xz(yz) \\ : (\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma \end{array}$$

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

Computation:

- β-reduction: $(\lambda x{:}\sigma.M)P \longrightarrow_\beta M[P/x]$
- η-reduction: $\lambda x{:}\sigma.Mx \longrightarrow_\eta M$ if $x \notin$ FV(M)

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

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

$$\dfrac{\dfrac{\begin{array}{c}[x : \sigma]^1\\ \mathcal{D}_1\\ M : \tau\end{array}}{\lambda x{:}\sigma.M : \sigma\to\tau}\,1 \qquad \dfrac{\mathcal{D}_2}{P : \sigma}}{(\lambda x{:}\sigma.M)P : \tau} \quad\rightsquigarrow\quad \begin{array}{c}\mathcal{D}_2\\ P : \sigma\\ \mathcal{D}_1\\ M[P/x] : \tau\end{array}$$

Properties of $\lambda\!\rightarrow$.

- Uniqueness of types
  If $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M : \tau$, then $\sigma = \tau$.
- Subject Reduction
  If $\Gamma \vdash M : \sigma$ and $M \longrightarrow_{\beta\eta} N$, then $\Gamma \vdash N : \sigma$.
- Strong Normalization
  If $\Gamma \vdash M : \sigma$, then all $\beta\eta$-reductions from M terminate.

Properties of $\lambda\!\to$.

- Uniqueness of types
  If $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M : \tau$, then $\sigma = \tau$.
- Subject Reduction
  If $\Gamma \vdash M : \sigma$ and $M \longrightarrow_{\beta\eta} N$, then $\Gamma \vdash N : \sigma$.
- Strong Normalization
  If $\Gamma \vdash M : \sigma$, then all $\beta\eta$-reductions from M terminate.
- Substitution property
  If $\Gamma, x : \tau, \Delta \vdash M : \sigma$, $\Gamma \vdash P : \tau$, then $\Gamma, \Delta \vdash M[P/x] : \sigma$.
- Thinning
  If $\Gamma \vdash M : \sigma$ and $\Gamma \subseteq \Delta$, then $\Delta \vdash M : \sigma$.
- Strengthening
  If $\Gamma, x : \tau \vdash M : \sigma$ and $x \notin \mathsf{FV}(M)$, then $\Gamma \vdash M : \sigma$.

Strong Normalization of β for $\lambda\!\to$.

Note:

- Terms may get larger under reduction
- Redexes may get multiplied under reduction.

Definition

- $[\![\alpha]\!] := \mathsf{Term}(\alpha) \cap \mathsf{SN}$.
- $[\![\sigma\!\to\!\tau]\!] := \{M : \sigma\!\to\!\tau | \forall N \in [\![\sigma]\!](M N \in [\![\tau]\!])\}$.

Lemma (all by induction on σ)

- $[\![\sigma]\!] \subseteq \mathsf{SN}$
- $x^{\sigma} \in [\![\sigma]\!]$
- If $M \in [\![\sigma]\!]$, $N \in [\![\tau]\!]$, $M[N/x] \in \mathsf{SN}$, then $M[N/x] \in [\![\sigma]\!]$.

Proposition

$$M : \sigma \Rightarrow M \in [\![\sigma]\!]$$

Corollary $\lambda\!\to$ is SN