Types Summer School
Giens France September 2002

Type Systems

Herman Geuvers
Nijmegen University, NL

Lecture 1: Simple and Polymorphic Type Theory

Simplest system: $\lambda\!\rightarrow$ 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 $\sigma, \tau, \ldots$.

Terms:

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

Examples:

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

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

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 $\lambda{\rightarrow}$:

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

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

Typical problems one would like to have an algorithm for:

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

Typical problems one would like to have an algorithm for:

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

For $\lambda\!\rightarrow$, all these problems are decidable.
Remarks:

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

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

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

Example

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

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

Computation:

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

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

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

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

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 $\lambda\!\rightarrow$.

- 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.

- 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

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

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

Polymorphic λ-calculus

Why Polymorphic λ-calculus?

- Simple type theory $\lambda\!\rightarrow$ is not very expressive
- In simple type theory, we can not 'reuse' a function.
  E.g. $\lambda x{:}\alpha.x : \alpha{\rightarrow}\alpha$ and $\lambda x{:}\beta.x : \beta{\rightarrow}\beta$.

We want to define functions that can treat types polymorphically:
add types $\forall\alpha.\sigma$:
Examples

- $\forall\alpha.\alpha{\rightarrow}\alpha$
  If $M : \forall\alpha.\alpha{\rightarrow}\alpha$, then M can map any type to itself.
- $\forall\alpha.\forall\beta.\alpha{\rightarrow}\beta{\rightarrow}\alpha$
  If $M : \forall\alpha.\forall\beta.\alpha{\rightarrow}\beta{\rightarrow}\alpha$, then M can take two inputs (of arbitrary types) and return a value of the first input type.

Derivation rules of $\lambda 2$:

Full (system F-style) polymorphism:

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

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \lambda\alpha.M : \forall\alpha.\sigma}\ \alpha \notin \mathsf{FV}(\Gamma) \qquad \frac{\Gamma \vdash M : \forall\alpha.\sigma}{\Gamma \vdash M\tau : \sigma[\tau/\alpha]}\ \text{for } \tau \text{ any type}$$

Examples:

- $\lambda\alpha.\lambda\beta.\lambda x{:}\alpha.\lambda y{:}\beta.x : \forall\alpha.\forall\beta.\alpha{\to}\beta{\to}\alpha.$
- $\lambda x{:}(\forall\alpha.\alpha).\lambda y{:}\sigma.x\tau : (\forall\alpha.\alpha){\to}\sigma{\to}\tau.$
- $\lambda x{:}(\forall\alpha.\alpha).x\,(\sigma{\to}\tau)\,(x\,\sigma) : (\forall\alpha.\alpha){\to}\tau.$

Recall: Important Properties

$$\begin{array}{ll} \Gamma \vdash M : \sigma? & \text{TCP} \\ \Gamma \vdash M : ? & \text{TSP} \\ \vdash ? : \sigma & \text{TIP} \end{array}$$

Properties of $\lambda 2$

- TIP is undecidable,
- TCP and TSP are equivalent & decidable.

Formulas-as-types for $\lambda 2$:
There is a formulas-as-types isomorphism between $\lambda 2$ and second order proposition logic, PROP2

Derivation rules of PROP2:

$$\frac{\Gamma \vdash \sigma}{\Gamma \vdash \forall \alpha . \sigma} \; \alpha \notin \mathsf{FV}(\Gamma) \qquad \frac{\Gamma \vdash \forall \alpha . \sigma}{\Gamma \vdash \sigma[\tau / \alpha]}$$

NB This is constructive second order proposition logic:

$$\forall \alpha . \forall \beta . ((\alpha \to \beta) \to \alpha) \to \alpha \text{ Peirce's law}$$

is not derivable.

Definability of the other connectives:

$$
\begin{aligned}
\bot &:= \forall\alpha.\alpha \\
\sigma\wedge\tau &:= \forall\alpha.(\sigma\to\tau\to\alpha)\to\alpha \\
\sigma\vee\tau &:= \forall\alpha.(\sigma\to\alpha)\to(\tau\to\alpha)\to\alpha \\
\exists\alpha.\sigma &:= \forall\beta.(\forall\alpha.\sigma\to\beta)\to\beta
\end{aligned}
$$

and all the standard constructive derivation rules are derivable.

Example ($\wedge$-elimination):

$$
\dfrac{\dfrac{\forall\alpha.(\sigma\to\tau\to\alpha)\to\alpha}{(\sigma\to\tau\to\sigma)\to\sigma} \qquad \dfrac{\dfrac{[\sigma]^1}{\tau\to\sigma}}{\sigma\to\tau\to\sigma}\,1}{\sigma}
$$

Idea:
The definition of a connective is an encoding of the elimination rule.

Data types in $\lambda 2$

$$\mathsf{Nat} := \forall \alpha . \alpha \to (\alpha \to \alpha) \to \alpha$$

This type can be used as the type of natural numbers, using the encoding of IN as Church numerals in the λ-calculus.

$$n \mapsto \lambda x . \lambda f . f(\ldots(f x)) \quad n\text{-times } f$$

- $0 := \lambda \alpha . \lambda x{:}\alpha . \lambda f{:}\alpha \to \alpha . x$
- $S := \lambda n{:}\mathsf{Nat} . \lambda \alpha . \lambda x{:}\alpha . \lambda f{:}\alpha \to \alpha . f(n \alpha x f)$
- Iteration: if $c{:}\sigma$ and $g{:}\sigma \to \sigma$, then define $\mathsf{It}\, c\, g : \mathsf{Nat} \to \sigma$ as

  $$\lambda n{:}\mathsf{Nat} . n\, \sigma\, c\, g$$

  Then

  $$\mathsf{It}\, c\, g\, n = g(\ldots(g c)) \;\; (n \text{ times } g)$$

  $\Rightarrow$ Define $+$, $\times$, $\ldots$ using iteration.

Properties of $\lambda 2$.

- 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.

Strong Normalization of β for $\lambda 2$.

Note:

- There are two kinds of β-reductions
  - $(\lambda x{:}\sigma.M)P \longrightarrow_\beta M[P/x]$
  - $(\lambda\alpha.M)\tau \longrightarrow_\beta M[\tau/\alpha]$
- The second doesn't do any harm: we can just look at the underlying untyped λ-terms

Recall the proof for $\lambda{\rightarrow}$:

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

Question:

How to define $[\![\forall\alpha.\sigma]\!]$??

$$[\![\forall\alpha.\sigma]\!] := \Pi_{X \in U}[\![\sigma]\!]_{\alpha:=X}??$$

Strong Normalization of β for $\lambda 2$.

Question:
How to define $[\![\forall\alpha.\sigma]\!]$??

$$[\![\forall\alpha.\sigma]\!] := \Pi_{X\in U}[\![\sigma]\!]_{\alpha:=X}??$$

- What is U?
  The collection of all 'possible' interpretations of types (?)
- $\Pi_{X\in U}[\![\sigma]\!]_{\alpha:=X}$ may get very (too?) big.

Girard:

- $[\![\forall\alpha.\sigma]\!]$ should be small

$$\bigcap_{X\in U}[\![\sigma]\!]_{\alpha:=X}$$

- Characterization of U.

U := SAT, the collection of saturated sets of (untyped) λ-terms.
$X \subset \Lambda$ is saturated if

- $xP_1 \dots P_n \in X$ (for all $x \in$ Var, $P_1, \dots, P_n \in$ SN)
- $X \subseteq$ SN
- If $M[N/x] \in X$ and $N \in$ SN, then $(\lambda x.M)N \in X$.

Let ρ : TVar $\rightarrow$ SAT be a valuation of type variables.
Define the interpretation of types $[\![\sigma]\!]_\rho$ as follows.

- $[\![\alpha]\!]_\rho := \rho(\alpha)$
- $[\![\sigma \rightarrow \tau]\!]_\rho := \{M | \forall N \in [\![\sigma]\!]_\rho (MN \in [\![\tau]\!]_\rho)\}$
- $[\![\forall \alpha.\sigma]\!]_\rho := \cap_{X \in \mathsf{SAT}} [\![\sigma]\!]_{\rho, \alpha := X}$

Proposition

$$x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M : \sigma \Rightarrow M[P_1/x_1, \ldots, P_n/x_n] \in [\![\sigma]\!]_\rho$$

for all valuations ρ and $P_1 \in [\![\tau_1]\!]_\rho, \ldots, P_n \in [\![\tau_n]\!]_\rho$

Corollary $\lambda 2$ is SN

(Proof: take P_1 to be x_1, ..., P_n to be x_n.)