Henk: a typed intermediate language

Simon Peyton Jones and Erik Meijer

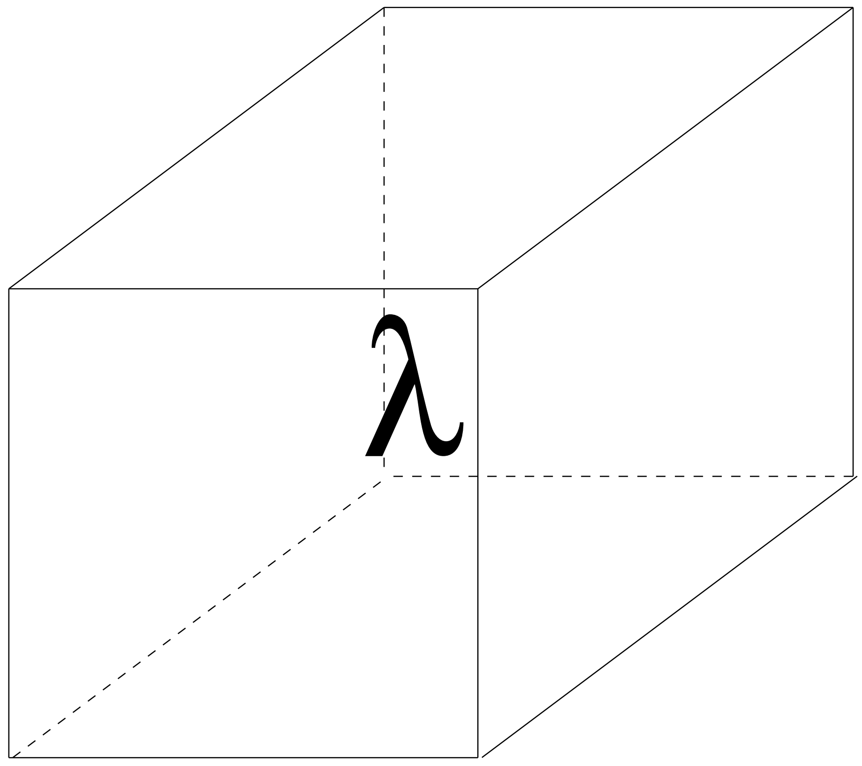
poisson

November 27, 2000

□ What is Henk?

- typed intermediate language based on the lambda-cube

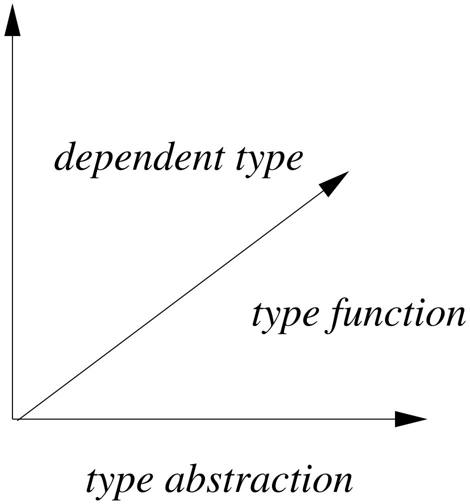
□ What is lambda-cube?

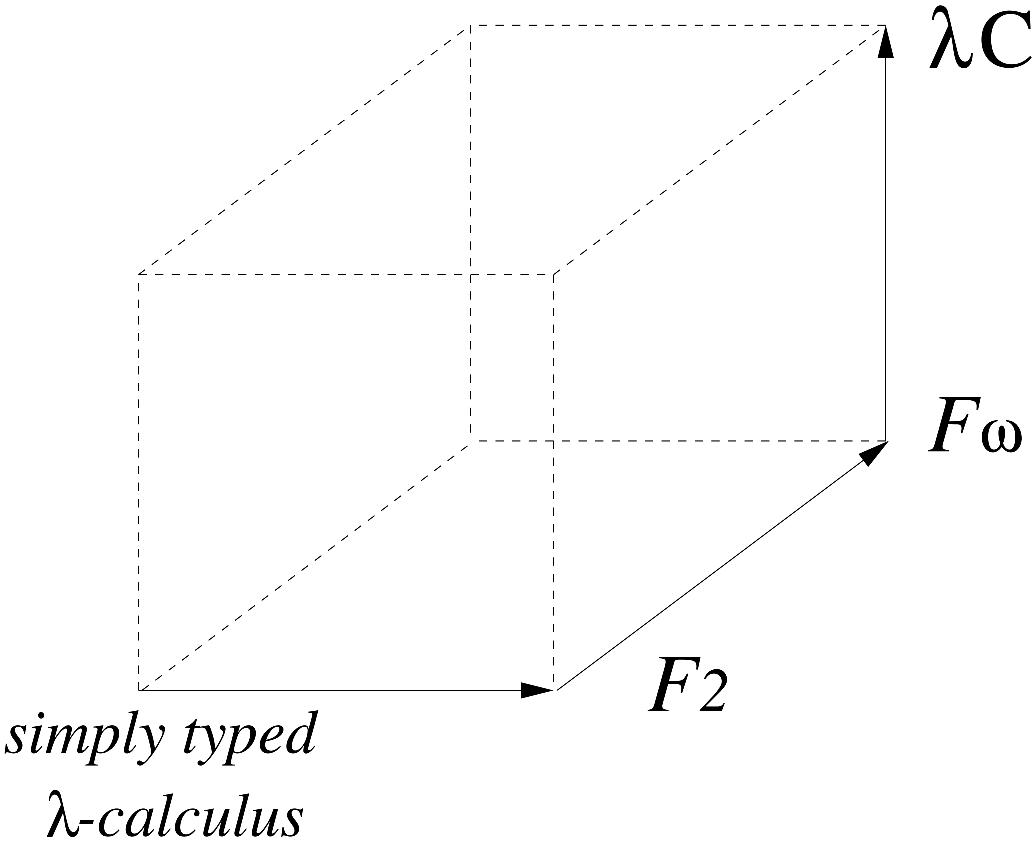


□ 4 kinds of abstraction

- terms can depend on terms - abstraction - simply-typed λ
  - abstract a term variable over a term expression
- terms can depned on types - type abstraction - F_2
  - abstract a type variable over a term expression
- types can depend on types - type function - F_ω
- types can depend on terms - dependent type - λ_C

□ What is lambda-cube?





□ Pure Type System, Barendregt[1992]

$$
\begin{array}{rcll}
E & ::= & K & \text{Constant} \\
 & | & x & \text{Variable} \\
 & | & E\ E & \text{Application} \\
 & | & \lambda x : E.E & \text{Abstraction} \\
 & | & \Pi x : E.E & \text{Quantification}
\end{array}
$$

- *lambda-cube* is an instance of PTS

□ Mixing terms and types

- types have the very same syntax as terms

- F_2
  - $\Lambda\alpha.\lambda\mathtt{x} : \alpha.\mathtt{id}\ [\alpha]\ \mathtt{x}$

- *lambda-cube*
  - $\lambda\alpha : \star.\lambda\mathtt{x} : \alpha.\mathtt{id}\ \alpha\ \mathtt{x}$

□ Quntification

- dependent product Π
  - $\Pi x : A.B$ is the type of functions from values of type A to values of type B, in which the result type B may perhaps depend on the value of the argument x.
- Π subsumes $\rightarrow$
  - $A \rightarrow B$ is an abbreviation for $\Pi \_ : A.B$
- Π subsumes $\forall$
  - $\forall \alpha.B$ is an abbreviation for $\Pi \alpha : \star.B$

□ Type rules for the lambda-cube

$$\frac{}{\vdash \star : \Box} \qquad \text{(STAR)}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad \text{(VAR)}$$

$$\frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B} \qquad \text{(WEAK)}$$

□ Type rules for the lambda-cube

$$\frac{\Gamma \vdash f : (\Pi x : A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash f\ a : B[a/x]} \qquad \text{(APP)}$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A.B) : t}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)} \qquad \text{(LAM)}$$

$$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : t \quad \vdash s \leadsto t}{\Gamma \vdash (\Pi x : A.B) : t} \qquad \text{(PI)}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash B : s \quad A =_\beta B}{\Gamma \vdash a : B} \qquad \text{(CONV)}$$

$\square \leadsto$

$$\frac{}{\vdash \star \leadsto \star} \quad (\star \leadsto \star)$$

$$\frac{}{\vdash \square \leadsto \star} \quad (\square \leadsto \star)$$

$$\frac{}{\vdash \square \leadsto \square} \quad (\square \leadsto \square)$$

$$\frac{}{\vdash \star \leadsto \square} \quad (\star \leadsto \square)$$

$$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : t \quad \vdash s \leadsto t}{\Gamma \vdash (\Pi x : A.B) : t} \quad (\mathsf{PI})$$

□ Instances of the lambda-cube

System	Corresponding subset of $\rightsquigarrow$
Simply typed λ-calculus	$\{\star \rightsquigarrow \star\}$
$F2$	$\{\star \rightsquigarrow \star, \Box \rightsquigarrow \star\}$
F_ω	$\{\star \rightsquigarrow \star, \Box \rightsquigarrow \star, \Box \rightsquigarrow \Box\}$
Calculus of constructions(λC)	$\{\star \rightsquigarrow \star, \Box \rightsquigarrow \star, \Box \rightsquigarrow \Box, \star \rightsquigarrow \Box\}$

□ Syntax Directed rules

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} \quad \text{(VAR)}$$

$$\frac{\Gamma \vdash f :\twoheadrightarrow (\Pi x : A.B) \quad \Gamma \vdash a : A' \quad A =_\beta A'}{\Gamma \vdash f \ a : B[a/x]} \quad \text{(APP)}$$

$$\frac{\Gamma \vdash A :\twoheadrightarrow s \quad \Gamma, x : A \vdash B :\twoheadrightarrow t \quad \vdash s \leadsto t}{\Gamma \vdash (\Pi x : A.B) : t} \quad \text{(PI)}$$

$$\frac{\Gamma \vdash a : A \quad A \twoheadrightarrow^{wh} B}{\Gamma \vdash a :\twoheadrightarrow B} \quad \text{(RED)}$$

□ Predicative variant

- impredicative
  - type variables can range over universally-quantified types.
  - `f` $(\forall\beta.\beta \to \beta)$
- predicative
  - type variables can not range over polytypes.
  - Harper and Mitchell* shows that predicative F_ω is sufficient for the interpretation of ML-style polymorphism.

*R.Harper and J.C.Mitchell, On the type structure of Standard ML, TOPLAS '93

□ λ^{ML} - a predicative variant of F_ω

$$
\begin{array}{lcll}
(kinds) & \kappa & ::= & \Omega \mid \kappa_1 \rightarrow \kappa_2 \\
(con's) & \mu & ::= & \alpha \mid \texttt{Int} \mid \lambda\alpha :: \kappa.\mu \mid \mu_1[\mu_2] \mid ... \\
(types) & \sigma & ::= & T(\mu) \mid \forall\alpha :: \kappa.\sigma \mid ... \\
(temrs) & e & ::= & x \mid \lambda x : \sigma.e \mid e_1\ e_2 \mid ... \mid e[\mu] \mid ...
\end{array}
$$

□ Predicative variant of lambda-cube

$\star$	is the kind of monotypes
$\star\star$	is the kind of polytypes

$$\frac{}{\Gamma \vdash \star\star : \Box\Box} \qquad \text{(STAR2)}$$

$$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : t \quad \vdash s \leadsto t : u}{\Gamma \vdash (\Pi x : A.B) : u} \qquad \text{(PI')}$$

□ Predicative variant of lambda-cube

$$\frac{}{\vdash \star \leadsto \star : \star} \qquad \frac{}{\vdash \Box \leadsto \star : \star\star}$$

$$\frac{}{\vdash \star \leadsto \star\star : \star\star} \qquad \frac{}{\vdash \Box \leadsto \star\star : \star\star}$$

$$\frac{}{\vdash \star\star \leadsto \star : \star\star} \qquad \frac{}{\vdash \Box \leadsto \Box : \Box}$$

$$\frac{}{\vdash \star\star \leadsto \star\star : \star\star}$$

□ Henk is

- elegantly parameterised (F_2, F_ω, λC, ...)

- very small
  - a single set of rules
  - a single data type
  - a single set of utility functions (prase, print, type-check, ...)