Automath and Pure Type Systems

Fairouz Kamareddine
Joint work with
Twan Laan and *Rob Nederpelt*
www.macs.hw.ac.uk/~fairouz/talks/talks2003/mlcauttalk03.ps

4 July 2003

Pure Type Systems: PTSs

- PTSs were introduced by Berardi and Terlow in 1988 and 1989 [1, 11].
- PTSs rules are highly influenced by rules of Automath (see van Daalen [3]).
- $\mathbb{T} ::= \mathbb{V} \mid \mathbb{C} \mid \mathbb{T}\mathbb{T} \mid \lambda\mathbb{V}{:}\mathbb{T}.\mathbb{T} \mid \Pi\mathbb{V}{:}\mathbb{T}.\mathbb{T}$.
- $(\lambda x{:}A_1.A_2)B \to_\beta A_2[x{:=}B]$
- Note, there is *no rule* $(\Pi x{:}A_1.A_2)B \to_\Pi A_2[x{:=}B]$
- A *specification* $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$:
  sorts $\boldsymbol{S} \subseteq \mathbb{C}$, *axioms* $\boldsymbol{A} \subseteq \boldsymbol{S} \times \boldsymbol{S}$ and *(Π-formation) rules* $\boldsymbol{R} \subseteq \boldsymbol{S} \times \boldsymbol{S} \times \boldsymbol{S}$.

Typing rules of PTSs

(axiom) $\langle\rangle \vdash s_1 : s_2$ $(s_1, s_2) \in \boldsymbol{A}$

(start) $$\frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$$ $x \notin \mathrm{DOM}\,(\Gamma)$

(weak) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x{:}C \vdash A : B}$$ $x \notin \mathrm{DOM}\,(\Gamma)$

(Π) $$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.B) : s_3}$$ $(s_1, s_2, s_3) \in \boldsymbol{R}$

(λ) $$\frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash (\Pi x{:}A.B) : s}{\Gamma \vdash (\lambda x{:}A.b) : (\Pi x{:}A.B)}$$

(appl) $$\frac{\Gamma \vdash F : (\Pi x{:}A.B) \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$$

(conv) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_\beta B'}{\Gamma \vdash A : B'}$$

Examples

- $\lambda\to$: $\boldsymbol{A} = (*, \Box)$ and $\boldsymbol{R} = \{(*, *, *)\}$.

(axiom) $\qquad \langle\rangle \vdash * : \Box$

(Π) $\qquad \dfrac{\Gamma \vdash A : * \qquad \Gamma, x{:}A \vdash B : *}{\Gamma \vdash (\Pi x{:}A.B) : *}$

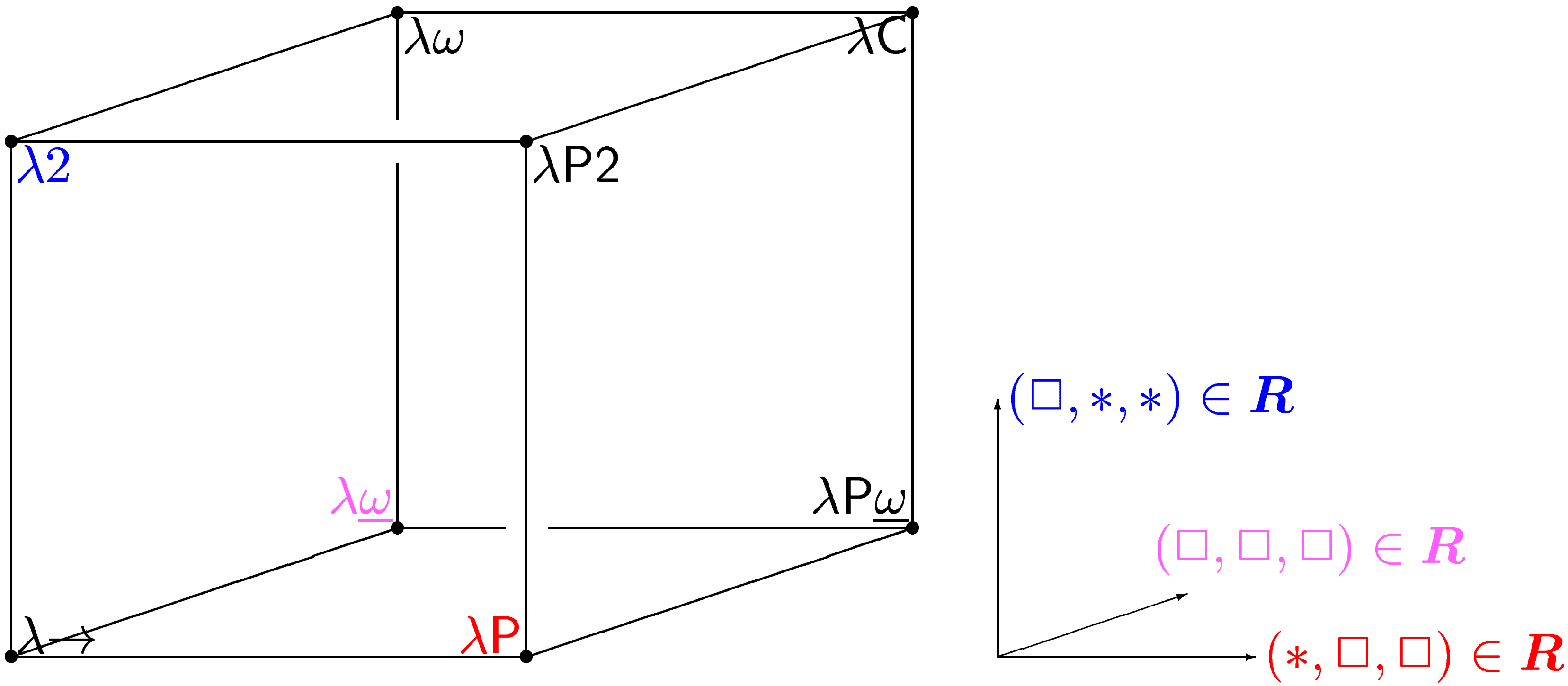
- $\lambda 2$: $\boldsymbol{A} = (*, \Box)$ and $\boldsymbol{R} = \{(*, *, *), (\Box, *, *)\}$.

$$\text{(axiom)} \qquad \langle\rangle \vdash * : \Box$$

$$(\Pi) \qquad \frac{\Gamma \vdash A : * \quad \Gamma, x{:}A \vdash B : *}{\Gamma \vdash (\Pi x{:}A.B) : *}$$

$$(\Pi) \qquad \frac{\Gamma \vdash A : \Box \quad \Gamma, x{:}A \vdash B : *}{\Gamma \vdash (\Pi x{:}A.B) : *}$$

The Barendregt Cube



Typing Polymorphic identity needs $(\Box, *)$

- $$\frac{y:*\vdash y:* \qquad y:*, x{:}y \vdash y:*}{y:*\vdash \Pi x{:}y.y:*}$$ by (Π) $(*, *)$

- $$\frac{y:*, x:y\vdash x:y \qquad y:*\vdash \Pi x{:}y.y:*}{y:*\vdash \lambda x:y.x:\Pi x{:}y.y}$$ by (λ)

- $$\frac{\vdash *:\Box \qquad y:*\vdash \Pi x{:}y.y:*}{\vdash \Pi y:*.\Pi x{:}y.y:*}$$ by (Π) $(\Box, *)$

- $$\frac{y:*\vdash \lambda x:y.x:\Pi x{:}y.y \qquad \vdash \Pi y:*.\Pi x{:}y.y:*}{\vdash \lambda y:*.\lambda x:y.x:\Pi y:*.\Pi x{:}y.y}$$ by (λ)

$\mathrm{A}\textsc{ut}$-68

- $\mathcal{E} ::= \quad \mathcal{V} \quad | \quad a(\underbrace{\mathcal{E}, \ldots, \mathcal{E}}) \quad | \quad [\mathcal{V} : \mathcal{E}^+]\mathcal{E} \quad | \quad \langle\mathcal{E}\rangle\mathcal{E}.$

  $a \in \mathcal{C}$, $\qquad \mathcal{V} \cap \mathcal{C} = \varnothing \qquad \mathcal{E}^+ \stackrel{\text{def}}{=} \mathcal{E} \cup \{\mathtt{type}\}.$

- Contexts $\Gamma ::= \quad \langle\rangle \quad | \quad \Gamma, \mathcal{V} : \mathcal{E}$ where variables are declared at most once.

- Lines $l ::= \quad \Gamma; \mathcal{V}; -; \mathcal{E}^+ \quad | \quad \Gamma; \mathcal{C}; \mathrm{PN}\ ; \mathcal{E}^+ \quad | \quad \Gamma; \mathcal{C}; \mathcal{E}; \mathcal{E}^+$

- Books $\mathfrak{B} \quad ::= \quad \emptyset \quad | \quad \mathfrak{B}, l.$

Example of an AUTOMATH-book

∅	prop	PN	type	(1)
∅	x	—	prop	(2)
x	y	—	prop	(3)
x,y	and	PN	prop	(4)
x	proof	PN	type	(5)
x,y	px	—	proof(x)	(6)
x,y,px	py	—	proof(y)	(7)
x,y,px,py	and-I	PN	proof(and)	(8)
x,y	pxy	—	proof(and)	(9)
x,y,pxy	and-O1	PN	proof(x)	(10)
x,y,pxy	and-O2	PN	proof(y)	(11)
x	prx	—	proof(x)	(12)
x,prx	and-R	and-I(x,x,prx,prx)	proof(and(x,x))	(13)
x,y,pxy	and-S	and-I(y,x,and-O2,and-O1)	proof(and(y,x))	(14)

Notions of correctness and of typing

- See D. van Daalen 1980 [4].
- $\mathfrak{B}; \emptyset \vdash \text{OK}$ indicates that book $\mathfrak{B}$ is correct.
- $\mathfrak{B}; \Gamma \vdash \text{OK}$ indicates that the context Γ is correct with respect to the (correct) book $\mathfrak{B}$.
- $\mathfrak{B}; \Gamma \vdash \Sigma_1 : \Sigma_2$ indicates that Σ_1 is a correct expression of type Σ_2 with respect to $\mathfrak{B}$ and Γ.
- We also say: $\Sigma_1 : \Sigma_2$ is a correct *statement* with respect to $\mathfrak{B}$ and Γ.
- The Automath book given earlier is correct.

Correct books and contexts

(axiom)

$$\varnothing;\varnothing \vdash \text{OK}$$

(context ext.)

$$\frac{\mathfrak{B}_1, (\Gamma; x; \text{—}; \alpha), \mathfrak{B}_2; \Gamma \vdash \text{OK}}{\mathfrak{B}_1, (\Gamma; x; \text{—}; \alpha), \mathfrak{B}_2; \Gamma, x{:}\alpha \vdash \text{OK}}$$

(book ext.: var1)

$$\frac{\mathfrak{B}; \Gamma \vdash \text{OK}}{\mathfrak{B}, (\Gamma; x; \text{—}; \texttt{type}); \varnothing \vdash \text{OK}}$$

(book ext.: var2)

$$\frac{\mathfrak{B}; \Gamma \vdash \Sigma_2 : \texttt{type}}{\mathfrak{B}, (\Gamma; x; \text{—}; \Sigma_2); \varnothing \vdash \text{OK}}$$

(book ext.: pn1)

$$\frac{\mathfrak{B}; \Gamma \vdash \text{OK}}{\mathfrak{B}, (\Gamma; k; \text{PN}; \texttt{type}); \varnothing \vdash \text{OK}}$$

(book ext.: pn2)

$$\frac{\mathfrak{B}; \Gamma \vdash \Sigma_2 : \texttt{type}}{\mathfrak{B}, (\Gamma; k; \text{PN}; \Sigma_2); \varnothing \vdash \text{OK}}$$

(book ext.: def1)

$$\frac{\mathfrak{B}; \Gamma \vdash \Sigma_1 : \texttt{type}}{\mathfrak{B}, (\Gamma; k; \Sigma_1; \texttt{type}); \varnothing \vdash \text{OK}}$$

(book ext.: def2)

$$\frac{\mathfrak{B}; \Gamma \vdash \Sigma_2 : \texttt{type} \; \mathfrak{B}; \Gamma \vdash \Sigma_1 : \Sigma_2' \; \mathfrak{B}; \Gamma \vdash \Sigma_2 =_{\beta \text{d}} \Sigma_2'}{\mathfrak{B}, (\Gamma; k; \Sigma_1; \Sigma_2); \varnothing \vdash \text{OK}}$$

For rules (book ext.) we assume $x \in \mathcal{V}$ and $k \in \mathcal{C}$ do not occur in $\mathfrak{B}$ or Γ.

Correct statements

(start)
$$\frac{\mathfrak{B};\Gamma_1,x{:}\alpha,\Gamma_2 \vdash \text{OK}}{\mathfrak{B};\Gamma_1,x{:}\alpha,\Gamma_2 \vdash x{:}\alpha}$$

(parameters)
$$\frac{\begin{array}{c}\mathfrak{B} \equiv \mathfrak{B}_1,(x_1{:}\alpha_1,\ldots,x_n{:}\alpha_n;b;\Omega_1;\Omega_2),\mathfrak{B}_2 \\ \mathfrak{B};\Gamma \vdash \Sigma_i{:}\alpha_i[x_1,\ldots,x_{i-1}{:=}\Sigma_1,\ldots,\Sigma_{i-1}](i=1,\ldots,n)\end{array}}{\mathfrak{B};\Gamma \vdash b(\Sigma_1,\ldots,\Sigma_n) : \Omega_2[x_1,\ldots,x_n{:=}\Sigma_1,\ldots,\Sigma_n]}$$

(abstr.1)
$$\frac{\mathfrak{B};\Gamma \vdash \Sigma_1{:}\mathtt{type} \quad \mathfrak{B};\Gamma,x{:}\Sigma_1 \vdash \Omega_1{:}\mathtt{type}}{\mathfrak{B};\Gamma \vdash [x{:}\Sigma_1]\Omega_1 : \mathtt{type}}$$

(abstr.2)
$$\frac{\mathfrak{B};\Gamma \vdash \Sigma_1{:}\mathtt{type} \quad \mathfrak{B};\Gamma,x{:}\Sigma_1 \vdash \Omega_1{:}\mathtt{type} \quad \mathfrak{B};\Gamma,x{:}\Sigma_1 \vdash \Sigma_2{:}\Omega_1}{\mathfrak{B};\Gamma \vdash [x{:}\Sigma_1]\Sigma_2 : [x{:}\Sigma_1]\Omega_1}$$

(application)
$$\frac{\mathfrak{B};\Gamma \vdash \Sigma_1 : [x{:}\Omega_1]\Omega_2 \quad \mathfrak{B};\Gamma \vdash \Sigma_2 : \Omega_1}{\mathfrak{B};\Gamma \vdash \langle\Sigma_2\rangle\Sigma_1 : \Omega_2[x{:=}\Sigma_2]}$$

(conversion)
$$\frac{\mathfrak{B};\Gamma \vdash \Sigma : \Omega_1 \quad \mathfrak{B};\Gamma \vdash \Omega_2{:}\mathtt{type} \quad \mathfrak{B};\Gamma \vdash \Omega_1 =_{\beta\mathrm{d}} \Omega_2}{\mathfrak{B};\Gamma \vdash \Sigma : \Omega_2}$$

When using the parameter rule, we assume that $\mathfrak{B};\Gamma \vdash \text{OK}$, even if $n = 0$.

Definitional Equality

- (β) $\langle\Sigma\rangle[x{:}\Omega_2]\Omega_1 \to_\beta \Omega_1[x{:=}\Sigma]$.

- (δ) If $\Sigma = b(\Sigma_1,\ldots,\Sigma_n)$, and $\mathfrak{B}$ contains a line $(x_1{:}\alpha_1,\ldots,x_n{:}\alpha_n;b;\Xi_1;\Xi_2)$ where $\Xi_1 \in \mathcal{E}$, then $\Sigma \to_\delta \Xi_1[x_1,\ldots,x_n{:=}\Sigma_1,\ldots,\Sigma_n]$.

From AUT-68 to PTSs

$$
\begin{array}{llll}
\overline{[\ldots]} : & \text{Correct Expressions in } \mathcal{E} & \mapsto & \mathbb{T} \\
& x & \mapsto & x \\
& \texttt{type} & \mapsto & * \\
& b(\Sigma_1, \ldots, \Sigma_n) & \mapsto & b\overline{\Sigma_1} \cdots \overline{\Sigma_n} \\
& \overline{\langle\Omega\rangle\Sigma} & \mapsto & \overline{\Sigma}\ \overline{\Omega} \\
& [x{:}\Sigma]\Omega & \mapsto & \begin{cases} \Pi x{:}\overline{\Sigma}.\overline{\Omega} & \text{if } [x{:}\Sigma]\Omega \text{ has type } \texttt{type}, \\ \lambda x{:}\overline{\Sigma}.\overline{\Omega} & \text{otherwise} \end{cases}
\end{array}
$$

Common features of modern types and functions

- We can *construct* a type by abstraction. (Write $A : *$ for A *is a type*)
  - $\lambda_{y:A}.y$, the identity over A *has type* $A \to A$
  - $\lambda_{A:*}.\lambda_{y:A}.y$, the polymorphic identity *has type* $\Pi_{A:*}.A \to A$

- We can *instantiate* types. E.g., if $A = \mathbb{N}$, then the identity over $\mathbb{N}$
  - $(\lambda_{y:A}.y)[A := \mathbb{N}]$ *has type* $(A \to A)[A := \mathbb{N}]$ or $\mathbb{N} \to \mathbb{N}$.
  - $(\lambda_{A:*}.\lambda_{y:A}.y)\mathbb{N}$ *has type* $(\Pi_{A:*}.A \to A)\mathbb{N} = (A \to A)[A := \mathbb{N}]$ or $\mathbb{N} \to \mathbb{N}$.

- $(\lambda x{:}\alpha.A)B \to_\beta A[x := B]$ $\qquad$ $(\Pi x{:}\alpha.A)B \to_\Pi A[x := B]$

- Write $A \to A$ as $\Pi_{y:A}.A$ when y not free in A.

Extending PTSs with Π-reduction and Π-application

- **Π-reduction** $(\Pi x{:}A.B)N \to_\Pi B[x{:=}N]$

- **Π-application** $\dfrac{\Delta;\Gamma \vdash M : \Pi x{:}A.B \qquad \Delta;\Gamma \vdash N : A}{\Delta;\Gamma \vdash MN : (\Pi x{:}A.B)N}$

- Also need to change in conversion, $=_\beta$ to $=_{\beta\Pi}$

- Kamareddine in 1996 [9] showed that PTSs with Π-reduction and Π-application lose Subject Reduction. For instance, one can derive $\alpha{:}*, x{:}\alpha \vdash (\lambda y{:}\alpha.y)x : (\Pi y{:}\alpha.\alpha)x$, but it is not possible to derive $\alpha{:}*, x{:}\alpha \vdash x : (\Pi y{:}\alpha.\alpha)x$.

- Kamareddine in 1999 [8] showed that PTSs with Π-reduction and Π-application have the desirable properties if a definition system is used.

Identifying λ and Π

Kamareddine 2002 [7] showed that:

- as long as the usual application rule of PTSs is used, a PTS system remains unchanged whether Π-reduction is included or not.
- If the usual application rule of PTSs is used, a PTS system remains unchanged whether λs and Πs are unified or not.
- [7] concluded that a PTS system where λs and Πs are unified and where the application is changed to Π-application faces the same problem (and inherits the same solution) as that of the PTSs where λs and Πs are not unified but where Π-application and Π-reduction are used.
- $\mathcal{T}_\flat ::= \mathcal{V} \mid \mathcal{C} \mid \mathcal{T}_\flat\mathcal{T}_\flat \mid \flat\mathcal{V}{:}\mathcal{T}_\flat.\mathcal{T}_\flat$

- ($\flat$) $$(\flat_{x:A}.B)C \to_\flat B[x := C].$$

(axiom) $\langle\rangle \vdash s_1 : s_2$ if $(s_1, s_2) \in \boldsymbol{A}$

(start) $$\frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A} \; x \notin \text{DOM}(\Gamma)$$

(weak) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x{:}C \vdash A : B} \; x \notin \text{DOM}(\Gamma)$$

($\flat_2$) $$\frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash (\flat x{:}A.B) : s}{\Gamma \vdash (\flat x{:}A.b) : (\flat x{:}A.B)}$$

(app$\flat$) $$\frac{\Gamma \vdash F : (\flat x{:}A.B) \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$$

($\flat_1$) $$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash (\flat x{:}A.B) : s_3} \; (s_1, s_2, s_3) \in \boldsymbol{R}$$

(conv) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_\flat B'}{\Gamma \vdash A : B'}$$

- For $A \in \mathcal{T}$, we define $\overline{A} \in \mathcal{T}_\flat$ as follows:
  - $\overline{s} \equiv s \qquad \overline{x} \equiv x \qquad \overline{AB} \equiv \overline{A}\,\overline{B}$
  - $\overline{\lambda_{x:A}.B} \equiv \overline{\Pi_{x:A}.B} \equiv \flat_{x:\overline{A}}.\overline{B}$.

- For contexts we define: $\overline{\langle\rangle} \equiv \langle\rangle \quad \overline{\Gamma, x : A} \equiv \overline{\Gamma}, x : \overline{A}$.

- For $A \in \mathcal{T}_\flat$, we define $[A]$ to be $\{A' \in \mathcal{T}$ such that $\overline{A'} \equiv A\}$.

- For context, obviously: $[\Gamma] \equiv \{\Gamma'$ such that $\overline{\Gamma'} \equiv \Gamma\}$.

Isomorphism of the cube and the $\flat$-cube

Kamareddine 2002 [7] showed that:

- If $\Gamma \vdash A : B$ then $\overline{\Gamma} \vdash_\flat \overline{A} : \overline{B}$.
- If $\Gamma \vdash_\flat A : B$ then there are unique $\Gamma' \in [\Gamma]$, $A' \in [A]$ and $B' \in [B]$ such that $\Gamma' \vdash_\pi A' : B'$.
- The $\flat$-cube enjoys the desirable properties of the cube such as Church Rosser, Strong Normalisation and Subject reduction.

Extending the $\flat$-cube with Π-reduction

If we change (app$\flat$) by (new app$\flat$) in the $\flat$-cube we lose subject reduction.

$$\text{(app}\flat\text{)} \quad \frac{\Gamma \vdash_\flat F : (\Pi_{x:A}.B) \qquad \Gamma \vdash_\flat a : A}{\Gamma \vdash_\flat Fa : B[x := a]}$$

$$\text{(new app}\flat\text{)} \quad \frac{\Gamma \vdash_\flat F : (\flat_{x:A}.B) \qquad \Gamma \vdash_\flat a : A}{\Gamma \vdash_\flat Fa : (\flat_{x:A}.B)a}$$

ML

- The example below is due to Joe Wells:
- ML treats let val id $=$ (fn $x \Rightarrow x$) in (id id) end as this Cube term
  $(\lambda \mathsf{id}{:}(\Pi\alpha{:}*.\ \alpha \to \alpha).\ \mathsf{id}(\beta \to \beta)(\mathsf{id}\,\beta))(\lambda\alpha{:}*.\ \lambda x{:}\alpha.\ x)$
- To type this in the Cube, the $(\Box, *)$ rule is needed (i.e., $\lambda 2$).
- ML's typing rules forbid this expression:
  let val id $=$ (fn $x \Rightarrow x$) in (fn $y \Rightarrow y\, y$)(id id) end
  Its equivalent Cube term is this well-formed typable term of $\lambda 2$:
  $(\lambda \mathsf{id} : (\Pi\alpha{:}*.\ \alpha \to \alpha).$
  $\quad(\lambda y{:}(\Pi\alpha{:}*.\ \alpha \to \alpha).\ y(\beta \to \beta)(y\,\beta))$
  $\quad(\lambda\alpha{:}*.\ \mathsf{id}(\alpha \to \alpha)(\mathsf{id}\,\alpha)))$
  $(\lambda\alpha{:}*.\ \lambda x{:}\alpha.\ x)$

- Therefore, ML should not have the full Π-formation rule $(\square, *)$.

- ML has limited access to the rule $(\Box, *)$ enabling some things from $\lambda 2$ but not all.

- ML's type system is none of those of the eight systems of the Cube.

- Parameters helped Laan [10] place the type system of ML on a refined Cube (between $\lambda 2$ and $\lambda\underline{\omega}$).

LF

- LF [6] is often described as λP of the Barendregt Cube.

- Geuvers showed that Use of Π-formation rule $(*, \Box)$ is very restricted in the practical use of LF [5].

- The only need for a type $\Pi x{:}A.B : \Box$ is when the Propositions-As-Types principle PAT is applied during the construction of the type $\Pi\alpha{:}\texttt{prop}.*$ of the operator Prf where for a proposition Σ, Prf(Σ) is the type of proofs of Σ.

$$\frac{\texttt{prop}{:}* \vdash \texttt{prop} : * \qquad \texttt{prop}{:}*, \alpha{:}\texttt{prop} \vdash *{:}\Box}{\texttt{prop}{:}* \vdash \Pi\alpha{:}\texttt{prop}.* : \Box}.$$

- In LF, this is the only point where the Π-formation rule $(*, \Box)$ is used.

- But, Prf is only used when applied Σ:$\texttt{prop}$. We never use Prf on its own.
- This use is in fact based on a parametric constant rather than on Π-formation.
- Hence, the practical use of LF would not be restricted if we present Prf in a parametric form, and use $(*, \Box)$ as a parameter instead of a Π-formation rule.
- Again, Laan [10] finds a more precise position of LF on the Cube (between $\lambda\!\rightarrow$ and λP).

Extending the Cube with parametric constants

- We add parametric constants of the form $c(b_1, \ldots, b_n)$ with $b_1, \ldots, b_n$ terms of certain types and $c \in \mathcal{C}$.

- $b_1, \ldots, b_n$ are called the *parameters* of $c(b_1, \ldots, b_n)$.

- $\boldsymbol{R}$ allows several kinds of Π-constructs. We also use a set $\boldsymbol{P}$ of (s_1, s_2) where $s_1, s_2 \in \{*, \Box\}$ to allow several kinds of parametric constants.

- $(s_1, s_2) \in \boldsymbol{P}$ means that we allow parametric constants $c(b_1, \ldots, b_n) : A$ where $b_1, \ldots, b_n$ have types $B_1, \ldots, B_n$ of sort s_1, and A is of type s_2.

- If both $(*, s_2) \in \boldsymbol{P}$ and $(\Box, s_2) \in \boldsymbol{P}$ then combinations of parameters allowed. For example, it is allowed that B_1 has type $*$, whilst B_2 has type $\Box$.

The Cube with parametric constants

- Let $(*,*) \subseteq \boldsymbol{R}, \boldsymbol{P} \subseteq \{(*,*),(*,\Box),(\Box,*),(\Box,\Box)\}$.

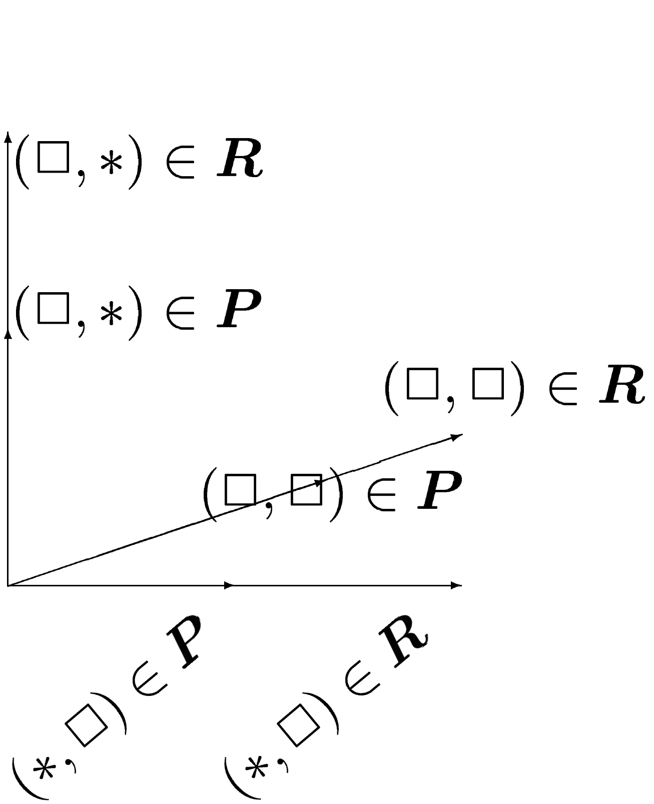
- $\lambda\boldsymbol{RP} = \lambda\boldsymbol{R}$ and the two rules **($\vec{\text{C}}$-weak)** and **($\vec{\text{C}}$-app)**:

$$\frac{\Gamma \vdash b : B \quad \Gamma, \Delta_i \vdash B_i : s_i \quad \Gamma, \Delta \vdash A : s}{\Gamma, c(\Delta) : A \vdash b : B} \; (s_i, s) \in \boldsymbol{P}, c \text{ is } \Gamma\text{-fresh}$$

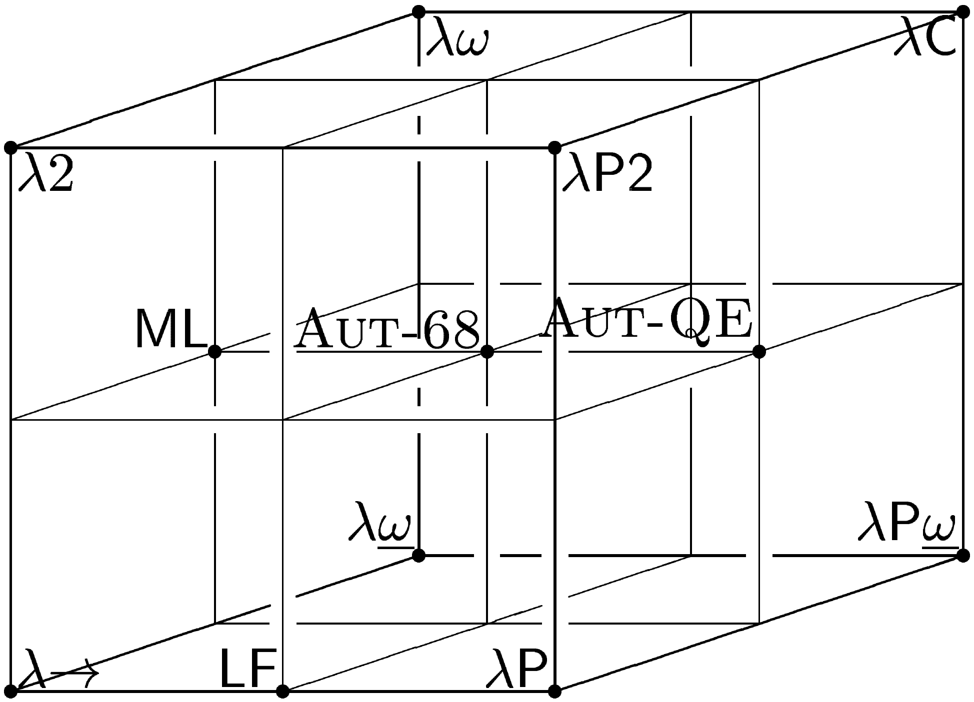
$$\frac{\begin{array}{lcll} \Gamma_1, c(\Delta){:}A, \Gamma_2 & \vdash & b_i{:}B_i[x_j{:=}b_j]_{j=1}^{i-1} & (i = 1, \ldots, n) \\ \Gamma_1, c(\Delta){:}A, \Gamma_2 & \vdash & A : s & (\text{if } n = 0) \end{array}}{\Gamma_1, c(\Delta){:}A, \Gamma_2 \vdash c(b_1, \ldots, b_n) : A[x_j{:=}b_j]_{j=1}^{n}}$$

$\Delta \equiv x_1{:}B_1, \ldots, x_n{:}B_n$.
$\Delta_i \equiv x_1{:}B_1, \ldots, x_{i-1}{:}B_{i-1}$

The refined Barendregt Cube



LF, ML, AUT-68, and AUT-QE in the refined Cube



The $\flat_{i\delta\sigma p}$-cube

- Bloo showed in [2] that PTSs with explicit substitutions lose desirable properties. He gives a solution based on definitions in contexts.

- Kamareddine showed in [7] that the $\flat$-cube loses the desirable properties of correctness of types and subject reduction. She gives a solution based on definitions in contexts. Parameters cause no problems in the cube.

- Substitutions can help solve the problem of local reductions of $\Delta\Lambda$ (see tomorrow).

- Kamareddine [7] defines the cube which has definitions in contexts, substitutions, parameters, and identifies λ and Π

- $\mathcal{T}_a ::= * \mid \Box \mid \mathcal{V} \mid \mathcal{C}(\mathcal{L}_T) \ \mid \flat_{\mathcal{V}:\mathcal{T}_a}.\mathcal{T}_a \mid \mathcal{T}_a\mathcal{T}_a \mid \mathcal{T}_a[\mathcal{V} \leftarrow \mathcal{T}_a]$, and $\mathcal{L}_T \ ::= \ \varnothing \mid \mathcal{L}_T, \mathcal{T}_a$.

- [7] shows that this cube has the desirable properties of correctness of types and subject reduction.

References

[1] S. Berardi. Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems in Barendregt's cube. Technical report, Dept. of Computer Science, Carnegie-Mellon University and Dipartimento Matematica, Universita di Torino, 1988.

[2] R. Bloo. The Barendregt cube with explicit vsubstitutions. Mathematical Structures in Computer Science, 2000.

[3] D.T. van Daalen. A description of Automath and some aspects of its language theory. In P. Braffort, editor, *Proceedings of the Symposium APLASM*, volume I, pages 48–77, 1973.

[4] D.T. van Daalen. *The Language Theory of Automath*. PhD thesis, Eindhoven University of Technology, 1980.

[5] J.H. Geuvers. *Logics and Type Systems*. PhD thesis, Catholic University of Nijmegen, 1993.

[6] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Proceedings Second Symposium on Logic in Computer Science*, pages 194–204, Washington D.C., 1987. IEEE.

[7] F. Kamareddine. *On Functions and Types: A Tutorial*, volume 2540 of *Lecture Notes in Computer Science*, pages 74–93. Springer Verlag, 2002.

[8] F. Kamareddine, R. Bloo, and R.P. Nederpelt. On π-conversion in the λ-cube and the combination with abbreviations. *Annals of Pure and Applied Logics*, 97:27–45, 1999.

[9] F. Kamareddine and R.P. Nederpelt. Canonical typing and Π-conversion in the Barendregt Cube. *Journal of Functional Programming*, 6(2):245–267, 1996.

[10] T. Laan. *The Evolution of Type Theory in Logic and Mathematics*. PhD thesis, Eindhoven University of Technology, 1997.

[11] J. Terlouw. Een nadere bewijstheoretische analyse van GSTT's. Technical report, Department of Computer Science, University of Nijmegen, 1989.