Théorie des types

Joëlle Despeyroux

INRIA Sophia-Antipolis

Module "Mécanisation des Preuves"

Dea d'informatique, Nice, janvier-mars 2000

Le langage mathématique sur ordinateur

double: $x \mapsto 2x$
puissance: $x, n \mapsto x * \cdots * x$, n fois.
sinus: $x \mapsto x - x^3/3! + x^5/5! - x^7/7! + \cdots$

Déf. par récurrence: $\begin{cases} puissance(x,0) = 1 \\ puissance(x,n+1) = x * puissance(x,n) \end{cases}$

Déf. par spécification: $\begin{cases} \forall x \ f(x) \geq 0 \\ \forall x \ f(x) * f(x) = x \\ \text{racine est l'unique f vérifiant ces équations} \end{cases}$

puissance=$\{f \ / \ \forall x \ f(x,0) = 1 \text{ et } f(x,n+1) = x * f(x,n)\}$

Si non unicité de l'objet f: Opérateur de choix (ϵ de Hilbert).
Ex: $\{x \ / \ x * x = 9\} = \{3, -3\}$

Un programme est une fonction f tel que $P(f)$

Programmation certifiée

Le langage mathématique sur ordinateur
(suite)

Notion de calcul.

Exécution d'un programme: règles d'évaluation, ou de réécriture. La plus simple est la β-reduction: substitution d'une variable par une valeur. Ex: $(5+x)[2/x] = 5+2 = 7$

Langages:

- fonctionnels: fonctions, avec un opérateur de récursion,
- impératifs: C++, Java, etc...,
- déclaratifs (ens. d'équations): Prolog, lang. par contraintes.

Fonctions non calculables. Ex: pb de l'arret. Suppose le tiers exclu.

Intuitionisme (constructivisme) ou classique?

Thm. Sans le tiers exclu, toutes les fonctions sont calculables.

Une démonstration constructive de l'existence de $\{f/P(f)\}$ fournit f, que l'on peut exécuter.

"Exécution" en logique: élimination des coupures.
Gestion des ressources: Logique linéaire.

Les démonstrations sont des objets $\rightarrow$ la théorie des types.

La logique du premier ordre

Langage logique: variables, symboles d'individu, de fonction et de prédicat, avec leur arité.
Ex. l'arithmétique: $\{\{x_0, x_1, \cdots\}, 0, \{S, +, *\}, =\}$.

Les termes (Ex. $S(0) + S(S(0))$)**:**

- les variables sont des termes,
- les symboles d'individu sont des termes,
- si f est un symbole de fonction d'arité n et $t_1 \cdots t_n$ des termes alors $f(t_1, \cdots, t_n)$ est un terme.

Les propositions (Ex. $\forall x.\ \neg(S(x) = x)$)**:**

- si P est un symbole de prédicat d'arité n et $t_1 \cdots t_n$ des termes alors $P(t_1, \cdots, t_n)$ est une proposition,
- $\bot$ est une proposition, si A est une proposition alors $\neg A$ est une proposition, si A et B sont des propositions alors $A \wedge B$, $A \vee B$, $A \Rightarrow B$ et $A \Leftrightarrow B$ sont des propositions,
- si A est une proposition et x une variable alors $\forall x.\ A$ et $\exists x.\ A$ sont des propositions.

Notation non ambigüe.
Ex. "Il y a un nombre entier supérieur à tout nombre entier":

- $\forall x.\ \exists y.\ y \geq x$ "Pour chaque nombre il existe un nombre qui lui est supérieur"
- $\exists y.\ \forall x.\ y \geq x$ "Il existe un nombre qui est supérieur à tous les nombres"

La logique du premier ordre (suite)

Les démonstrations:
Axiomes et règles.
Exemple: l'égalité: 2 axiomes:

$\forall x.\ x = x$ Principe d'identité
$\forall x.\ \forall y.\ ((x = y) \Rightarrow (A[x/z] \Rightarrow A[y/z]))$ Axiome de Leibnitz

Définition:
Substitution: $A[t/x]$ déf. par récurrence sur la structure de A.

Exemple: l'arithmétique: les axiomes de Peano:

$\forall x.\ \forall y.\ (S(x) = S(y) \Rightarrow x = y$
$\forall x.\ \neg(0 = S(x))$
$(A[0/z] \wedge (\forall x.\ (A[x/z] \Rightarrow A[S(x)/z]))) \Rightarrow \forall y.\ A[y/z]$
$\forall x.\ (0 + x = x)$
$\forall x.\ \forall y.\ (S(x) + y = S(x + y))$
$\forall x.\ (0 * x = 0)$
$\forall x.\ \forall y.\ (S(x) * y = (x * y) + y)$

Les systèmes de Hilbert

Axiomes logiques:

$$
\begin{array}{l}
A \Rightarrow (B \Rightarrow A) \\
(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)) \\
A \Rightarrow B \Rightarrow (A \wedge B) \\
(A \wedge B) \Rightarrow A \\
(A \wedge B) \Rightarrow B \\
A \Rightarrow (A \vee B) \\
B \Rightarrow (A \vee B) \\
(A \vee B) \Rightarrow ((A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow C)) \\
(A \Rightarrow \bot) \Rightarrow \neg A \\
A \Rightarrow (\neg A \Rightarrow \bot) \\
\bot \Rightarrow A \\
(\forall x\ (A \Rightarrow B)) \Rightarrow (A \Rightarrow \forall x\ B) \text{ si } x \notin A \\
\forall x\ A \Rightarrow A[t/x] \\
A[t/x] \Rightarrow \exists x\ A \\
\exists x\ A \Rightarrow ((\forall x\ (A \Rightarrow B)) \Rightarrow B) \text{ si } x \notin B
\end{array}
$$

En logique classique, on ajoute l'axiome: $A \vee \neg A$

Les systèmes de Hilbert *(suite)*

Une **démonstration** dans Γ peut être obtenue par:

- axiome: si A est un axiome logique ou une proposition de Γ,
- modus ponens: $\dfrac{A \Rightarrow B \quad A}{B}$,
- généralisation: $\dfrac{A}{\forall x.\ A}$ x non libre dans Γ.

Représentation par un *arbre de dérivation*:

$$\dfrac{\dfrac{A \Rightarrow (B \Rightarrow (A \wedge B)) \qquad A}{B \Rightarrow (A \wedge B)} \qquad B}{A \wedge B} \qquad\qquad \dfrac{\dfrac{\mathcal{C} \Rightarrow \mathcal{B} \Rightarrow (A \Rightarrow A)) \qquad \mathcal{C}}{\mathcal{B} \Rightarrow (A \Rightarrow A)} \qquad \mathcal{B}}{A \Rightarrow A}$$

où $\mathcal{B}$ est $A \Rightarrow (A \Rightarrow A)$ et $\mathcal{C}$ est $A \Rightarrow ((A \Rightarrow A) \Rightarrow A)$.

Les systèmes de Hilbert *(suite)*

Lemme de déduction:
$A \Rightarrow B$ est démontrable dans Γ ssi B est démontrable dans Γ, A.

Preuve.

– ($\Rightarrow$) Si il y a une démonstration π de $A \Rightarrow B$ dans Γ, alors

$$\frac{\dfrac{\pi}{A \Rightarrow B)} \qquad A}{B}$$ est une démonstration de B dans Γ, A.

– ($\Leftarrow$) On montre par récurrence sur la hauteur de la démonstration de B dans Γ, A l'existence d'une démonstration de $A \Rightarrow B$ dans Γ.

  * Cas où B est un axiome. $\dfrac{B \Rightarrow (A \Rightarrow B) \qquad B}{A \Rightarrow B}$ est une démonstration de $A \Rightarrow B$.
  * Cas où $B = A$. cf la démonstration de $A \Rightarrow A$.
  * Cas modus ponens: $\dfrac{C \Rightarrow B \quad C}{B}$.
    Par hypothèse de récurrence, il existe une démonstration π_1 de $A \Rightarrow (C \Rightarrow B)$ et une démonstration π_2 de $A \Rightarrow C$.
    [...] est une démonstration de $A \Rightarrow B$.
  * Cas généralisation ($B = \forall x.\ C$): $\dfrac{C}{\forall x.\ C}$ x non libre dans Γ.
    Par hypothèse de récurrence, il existe une démonstration π de $A \Rightarrow C$ dans Γ.
    [...] est une démonstration de $A \Rightarrow B$ dans Γ.

La logique du premier ordre (suite)

Langages avec plusieurs sortes: types simples associés aux variables, fonctions et prédicats.

La déduction modulo un système de réécriture (qui doit être confluent et terminant).
Ex.: règles de réécriture pour l'addition et la multiplication:

$$\begin{aligned} 0 + y &\hookrightarrow y \\ S(x) + y &\hookrightarrow S(x+y) \\ 0 * y &\hookrightarrow 0 \\ S(x) * y &\hookrightarrow (x * y) + y \end{aligned}$$

La déduction naturelle

Déf. Un *séquent* est une paire $\Gamma \vdash A$.

On prend l'axiome $\Gamma \vdash A$ *si* $A \in \Gamma$, et la règle: $\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$.

Les 3 1ers axiomes du système de Hilbert deviennent prouvables:

$$\dfrac{\dfrac{\Gamma, A, B \vdash A}{\Gamma, A \vdash B \Rightarrow A}}{\Gamma \vdash A \Rightarrow (B \Rightarrow A)} \qquad \dfrac{\dfrac{\dfrac{\Gamma, A, B \vdash A \qquad \Gamma, A, B \vdash B}{\Gamma, A, B \vdash A \wedge B}}{\Gamma, A \vdash B \Rightarrow (A \wedge B)}}{\Gamma \vdash A \Rightarrow (B \Rightarrow (A \wedge B))}$$

On remplace l'axiome $A \Rightarrow B \Rightarrow (A \wedge B)$ par la règle $\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$
Cet axiome peut maintenant se prouver: [...].

On remplace l'axiome $(A \vee B) \Rightarrow ((A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow C))$ par la règle: $\dfrac{\Gamma \vdash A \vee B \quad \Gamma \vdash A \Rightarrow C \quad \Gamma \vdash B \Rightarrow C}{\Gamma \vdash C}$ ou mieux par $\dfrac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$.

On obtient ainsi des **règles d'introduction et d'élimination** des opérateurs logiques, selon que l'opérateur concerné apparaît en conclusion ou en prémisse de la règle.

Modus ponens $\dfrac{A \Rightarrow B \quad A}{B}$ devient la règle d'élimination de $\Rightarrow$.
La règle de généralisation $\dfrac{A}{\forall x.\ A}$ x non libre dans Γ devient la règle d'introduction de $\forall$.

La déduction naturelle: le système

$ax : \Gamma \vdash A \quad si\ A \in \Gamma$

$$\Rightarrow_I: \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \Rightarrow_E: \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\wedge_I : \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \wedge_{E1} : \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \wedge_{E2} : \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$\vee_{I1} : \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \vee_{I2} : \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\vee_E : \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

$$\neg_I : \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A} \qquad \neg_E : \frac{\Gamma \vdash A \qquad \Gamma \vdash \neg A}{\Gamma \vdash \bot} (\bot_I) \qquad \bot_E : \frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

$$\forall_I : \frac{\Gamma \vdash A}{\Gamma \vdash \forall x.\ A} (x \notin \Gamma) \qquad \forall_E : \frac{\Gamma \vdash \forall x.\ A}{\Gamma \vdash A[t/x]}$$

$$\exists_I : \frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x.\ A} \qquad \exists_E : \frac{\Gamma \vdash \exists x.\ A \qquad \Gamma, A \vdash B}{\Gamma \vdash B} (x \notin \Gamma, B)$$

Thm. Si A est démontrable à partir de Γ dans le système de Hilbert, $\Gamma \vdash A$ est démontrable en déduction naturelle.

La déduction naturelle classique

En logique classique on ajoute le tiers exclu: $\Gamma \vdash A \vee \neg A$

Alternatives:

$$\neg_I : \frac{\Gamma, A \vdash B \qquad \Gamma, A \vdash \neg B}{\Gamma \vdash \neg A} \qquad \neg_E : \frac{\Gamma \vdash A \qquad \Gamma \vdash \neg A}{\Gamma \vdash B}$$

$$\text{tiers exclu} : \frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$$

Exemple en logique classique: Proposition de Peirce:

$$\dfrac{\dfrac{\ldots}{\Gamma \vdash \neg\neg(((P \Rightarrow Q) \Rightarrow P) \Rightarrow P)}}{\Gamma \vdash ((P \Rightarrow Q) \Rightarrow P) \Rightarrow P}$$

La déduction naturelle: les coupures

Les coupures: Introduction suivie d'une élimination.

Ex: $$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash A} \quad \dfrac{\pi_2}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}}{\Gamma \vdash A} \hookrightarrow \dfrac{\pi_1}{A} \qquad \dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash \forall x.\ A}}{\Gamma \vdash A[t/x]} \hookrightarrow \dfrac{\pi[t/x]}{\Gamma \vdash A[t/x]}$$

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\pi_2}{\Gamma \vdash A}}{\Gamma \vdash B} \hookrightarrow \dfrac{\pi}{\Gamma \vdash B}$$

π: π_1 où on enlève A dans tous les contextes et on remplace

les axiomes $\Gamma, A, \Delta \vdash A$ par $\dfrac{\pi_2[\text{on rajoute l'hypothèse } \Delta \text{ partout}]}{\Gamma, \Delta \vdash A}$.

Thm. Le processus d'élimination des coupures termine.

Le calcul des séquents

Motivation: la recherche automatique de démonstrations.
Règles structurelles (manipulation explicite des contextes).
Règles intro/élim symétriques.

$$axiome: \ A \vdash A \qquad coupure: \ \frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}$$

$$aff: \ \frac{\Gamma \vdash A}{\Gamma, B \vdash A} \quad contr: \ \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \quad perm: \ \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

$$\Rightarrow_I: \ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \Rightarrow_E: \ \frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \Rightarrow B \vdash C}$$

$$\wedge_I: \ \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \wedge_E: \ \frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}$$

$$\vee_{I1}: \ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \vee_{I2}: \ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \quad \vee_E: \ \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}$$

$$\forall_I: \ \frac{\Gamma \vdash A}{\Gamma \vdash \forall x.\ A} \ (x \notin \Gamma) \qquad \forall_E: \ \frac{\Gamma, A[t/x] \vdash B}{\Gamma, \forall x.\ A \vdash B}$$

$$\exists_I: \ \frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x.\ A} \qquad \exists_E: \ \frac{\Gamma, A \vdash B}{\Gamma, \exists x.\ A \vdash B} \ (x \notin \Gamma, B)$$

$$\neg_I: \ \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A} \qquad \neg_E: \ \frac{\Gamma \vdash A}{\Gamma, \neg A \vdash B} \qquad \bot_E: \ \Gamma, \bot \vdash A$$

Le calcul des séquents (suite)

Thm. Tout séquent démontrable en déduction naturelle est démontrable en calcul des séquents, et vis et versa.

Plus exactement:
À une preuve de $\Gamma \vdash_s A$ correspond une déduction $\Gamma \vdash_d A$.
Une déduction $\Gamma \vdash_d A$ peut être représentée dans le calcul des séquents, mais pas de manière unique.

Démonstration: *Proofs and Types* de J.Y. Girard page 35, etc...

Le calcul des séquents classique

On peut rajouter la règle:

tiers exclu : $\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$

Un système plus élégant définit le séquent $\Gamma \vdash \Delta$ où Δ est une liste de propositions, possiblement vide ($\perp$).
$A_1, \cdots, A_n \vdash B_1, \cdots, B_m$ signifie "A_1 et ... $A_n \Rightarrow B_1$ ou ... B_m".

Le calcul intuitioniste est la restriction de ce système au cas où Δ est réduit à une seule proposition B ou $\perp$, excepté pour la règle $\vee_E$, qui est: $\vee_E$: $\dfrac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta}{\Gamma, \Gamma', A \vee B \vdash \Delta}$.

Le système:

$$axiome : \; A \vdash A \qquad coupure : \; \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

$$aff_l : \; \frac{\Gamma \vdash \Delta}{\Gamma, B \vdash \Delta} \qquad aff_r : \; \frac{\Gamma \vdash \Delta}{\Gamma \vdash B, \Delta}$$

$$contr_l : \; \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad contr_r : \; \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

$$perm_l : \; \frac{\Gamma, A, B, \Gamma' \vdash \Delta}{\Gamma, B, A, \Gamma' \vdash \Delta} \qquad perm_r : \; \frac{\Gamma \vdash \Delta, A, B\Delta'}{\Gamma \vdash \Delta, B, A, \Delta'}$$

Le calcul des séquents classique (suite)

$$\Rightarrow_I: \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta} \qquad \Rightarrow_E: \frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \Rightarrow B \vdash \Delta, \Delta'}$$

$$\wedge_I : \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'}$$

$$\wedge_{E1} : \frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \qquad \wedge_{E2} : \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}$$

$$\vee_{I1} : \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \vee B, \Delta} \qquad \vee_{I2} : \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta}$$

$$\vee_E : \frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \vee B \vdash \Delta, \Delta'}$$

$$\neg_I : \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta} \qquad \neg_E : \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}$$

$$\forall_I : \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x.\ A, \Delta}\ (x \notin \Gamma, \Delta) \qquad \forall_E : \frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, \forall x.\ A \vdash \Delta}$$

$$\exists_I : \frac{\Gamma \vdash A[t/x], \Delta}{\Gamma \vdash \exists x.\ A, \Delta} \qquad \exists_E : \frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x.\ A \vdash \Delta}\ (x \notin \Gamma, \Delta)$$

La théorie des types simples

Motivation initiale: définir une théorie des ensembles cohérente.
Le paradoxe de Russel: L'ensemble $\{x/\neg x \in x\}$ appartient à lui-même ssi il ne s'appartient pas à lui-même.
Pour l'éviter: 2 possibilités:
Toute proposition n'est pas obligatoirement un objet
$\rightarrow$ La *théorie des ensembles* de Zermelo-Fraenkel.
Toute proposition ne peut pas s'appliquer à tout objet
$\rightarrow$ la *théorie des types* simples, où l'on distingue les objets selon leur fonctionnalité: objets de bases, fonctions, etc...

Types: i et o sont des types; Si A et B sont des types alors $A \rightarrow B$ est un type.
Ex. $0 : i, \quad S : i \rightarrow i, \quad =: i \rightarrow i \rightarrow o, \quad \vee : o \rightarrow o \rightarrow o, \ldots.$
Il faut ajouter pour chaque type:
$=_A$: $A \rightarrow A \rightarrow o, \quad \forall_A, \exists_A : \ (A \rightarrow o) \rightarrow o,$
avec éventuellement des axiomes d'extensionalité:
$\forall f, g.\ (\forall x.\ f(x) = g(x)) \Rightarrow (f = g).$

Le λ-calcul simplement typé

types : $A ::= \alpha \mid A \rightarrow A'$

termes : $M ::= x \mid \lambda x : A.M' \mid (M\ \ M')$

contextes : $\Gamma ::= \ . \mid \Gamma, x : A$

reduction : $(\lambda x : A.M \quad N) \hookrightarrow_\beta M[N/x]$

Un contexte est légal si tous les x_i sont distincts.

$$var\ \frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \Gamma \text{ légal} \qquad lam\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M \ :\ A \rightarrow B}$$

$$app\ \frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ \ N) : B}$$

Exemples:

$$\begin{array}{ll}
 & \vdash \lambda x : A.x : A \rightarrow A \\
y : B & \vdash \lambda x : A.y : A \rightarrow B \\
y : B, z : A & \vdash (\lambda x : A.y \quad z) : B \\
 & \vdash \lambda x : A.\lambda y : B.x : A \rightarrow B \rightarrow A
\end{array}$$

Le terme $\lambda x.\ (x\ \ x)$ n'est pas bien typé.

Problème de **capture de variable**:
$\lambda x.\lambda x.x \equiv \lambda x'.\lambda x.x$ ou $\lambda x.\lambda x'.x$?

Solution: α**-conversion:** $\lambda x.x \equiv \lambda y.y$ et $\lambda x.\lambda x.x \equiv \lambda x.\lambda y.y$.

Autres réductions:

$\lambda x : A.M \hookrightarrow_\alpha \lambda y : A.M[y/x]$ – renommage

$\lambda x : A.(M\ \ x) \hookrightarrow_\eta M$ si $x \notin M$ – Ex : $\lambda x : i.\ (S\ \ x) = S$

Le λ-calcul simplement typé: propriétés

Unicité du type: Pour tout contexte Γ et terme t, il existe au plus un type A tel que $\Gamma \vdash t : A$.

Il existe un algorithme qui étant donnés un contexte Γ et terme t, décide si t est typable dans Γ et donne son type, s'il existe.

Confluence: Si $t \hookrightarrow u_1$ et $t \hookrightarrow u_2$ alors il existe v tq $t \hookrightarrow v$.

Un terme *normal* est de la forme $\lambda x_1 : A_1. \ldots \lambda x_n : A_n. (x\ u_1 \ldots u_p)$.

t est *faiblement normalisable* si il existe v normal tq $t \hookrightarrow^* v$.

t est *fortement normalisable* si toute suite de réduction issue de t est finie.

Thm. Tout terme est fortement normalisable (pour β), et la forme normale est unique.

Thm. Tout terme est faiblement normalisable pour $\beta\eta$.

Déf. Un terme t de type $A_1 \to \ldots \to A_n \to B$ est en *forme β-normale η-longue* s'il est de la forme $\lambda x_1 : A_1. \ldots \lambda x_n : A_n. (x\ u_1 \ldots u_p)$, où les u_i sont en forme β-normale η-longue; Cette forme est unique.

Sémantique (interprétations de la logique)

Sémantique de Tarski: (tables de vérité)

A	B	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$\neg A$
t	t	t	t	t	f
f	t	f	t	t	t
t	f	f	t	f	
f	f	f	f	t	

Sémantique de Heyting-Kolmogorov:

- Une démonstration de $A \Rightarrow B$ est une fonction qui associe à toute démonstration de A une démonstration de B,
- Une démonstration de $A \wedge B$ est un couple formé d'une démonstration de A et d'une démonstration de B,
- Une démonstration de $A \vee B$ est un couple (i, p), où soit $i = 0$ et p est une démonstration de A, soit $i = 1$ et p est une démonstration de B,
- Une démonstration de $\neg A$ est une démonstration de $A \Rightarrow \perp$, où $\perp$ n'a aucune démonstration,
- Une démonstration de $\forall x.\ A$ est une fonction qui associe à tout objet t une démonstration de $A[t/x]$,
- Une démonstration de $\exists x.\ A$ est un couple (t, p), où t est un object, et p est une démonstration de $A[t/x]$.

L'isomorphisme de Curry-Howard

Les types:

Une formule de la logique *est représentée par un type* du λ-calcul:

– Les prop. atomiques $A, B, \cdots$ sont des types atomiques.

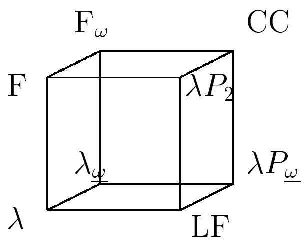
– À $A \Rightarrow B$ correspond $A \rightarrow B$, à $A \wedge B$ correspond $A \times B$.

Une preuve d'une formule A est un terme de type A:

Les termes:

logique	λ – calcul
$\Gamma \vdash A \quad (A \in \Gamma)$	$x_1 : A_1 \; \cdots \; x_n : A_n] \vdash x_i : A_i$
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$	$\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.\, t : A \rightarrow B}$
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash (t\ a) : B}$
$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	$\dfrac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : (A \times B)}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$	$\dfrac{\Gamma \vdash t : (A \times B)}{\Gamma \vdash \mathrm{fst}\ t : A}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash t : (A \times B)}{\Gamma \vdash \mathrm{snd}\ t : B}$

Le λ-cube



Présentation uniforme de 8 λ-calculs typés
[H. Barendregt, S. Berardi, J.Terlow]

– Typage à la Church ($\lambda x : A.M$)

– λ-calculs vérifiant l'isomorphisme de Curry-Howard

– *Logical Frameworks*: cadre logique pour étudier (spécifications et preuves de propriétés) des logiques ou des langages.

Le cube est orienté; le calcul le plus riche est CC: le Calcul des Constructions.

On ajoute aux λ-calculs du cube des *types inductifs*, et des opérateurs de *récursion*.
Ex: CCind: le Calcul des Constructions + des opérateurs permettant de définir des types inductifs + des opérateurs de récursion (raisonnement par cas et point fixe).

Système F

$$
\begin{array}{lll}
\mathit{types} & A & ::= \alpha \mid A \rightarrow A' \mid \forall \alpha . A \\
\mathit{termes} & M & ::= x \mid \lambda x : M.M' \mid \Lambda \alpha . M \mid (M\ M') \mid (M\ A) \\
\mathit{contextes} & \Gamma & ::= . \mid \Gamma, x : A
\end{array}
$$

$$
\begin{array}{ll}
\mathit{reductions} & (\lambda x : A.M \quad N) \hookrightarrow_\beta M[N/x] \\
 & (\Lambda \alpha . M \quad A) \hookrightarrow_\beta M[A/\alpha]
\end{array}
$$

$$\mathit{var}\ \frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \Gamma\ \mathit{legal}$$

$$\mathit{lam}\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M \ : \ A \rightarrow B}$$

$$\mathit{app}\ \frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B}$$

$$\mathit{Lam}\ \frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda \alpha . M \ : \ \forall \alpha . A}\ \alpha \notin FV(\Gamma)$$

$$\mathit{App}\ \frac{\Gamma \vdash M : \forall \alpha . A}{\Gamma \vdash (M\ B) : A[B/\alpha]}\ B \in \mathit{types}$$

Exemples:

$\vdash \lambda x : A.x : A \rightarrow A$

$\vdash \Lambda \alpha . \lambda x : \alpha . x : \forall \alpha . \alpha \rightarrow \alpha$

$\vdash (\Lambda \alpha . \lambda x : \alpha . x \quad A) : A \rightarrow A$

Termes et types dépendants

– les termes peuvent dépendre de termes et/ou de types

– les types peuvent dépendre de termes et/ou de types

– λ

$f : A \to B, x : A \vdash (f\ x) : B$
le terme $(f\ x)$ depend du terme x.

– F - polymorphisme

$f : \forall \alpha.\alpha \to \alpha \vdash (f\ A) : A \to A$
le terme $(f\ A)$ depend du type A.

– $\lambda_{\underline{\omega}}$

$f = \lambda\alpha : types.\alpha \to \alpha$ avec $(f\ A) = A \to A$
le type $(f\ A)$ depend du type A.

2 constantes: $*$ et $\Box$. '$\alpha \in types$' devient $\vdash \alpha : *$.

$f = \lambda\alpha : *.\alpha \to \alpha \in kinds$

$Kinds = \{*, * \to *, * \to * \to *, \cdots\}$.

$kinds \quad K \quad ::= * \mid K \to K'$

'$k \in kinds$' devient $\vdash k : \Box$. en particulier $\vdash * : \Box$.

$\lambda_{\underline{\omega}}$. types dép. de types

$$\begin{array}{llll} \textit{termes} & M & ::= & x \mid c \mid \lambda x : M.M' \mid (M\ M') \mid M \rightarrow M' \\ \textit{sortes} & s & \in & \{*, \Box\} \subset \mathcal{C} \end{array}$$

$$ax \ \vdash * : \Box \qquad var \ \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \ x \notin \Gamma$$

$$aff \ \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B} \ x \notin \Gamma$$

$$lam \ \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash (A \rightarrow B) : s}{\Gamma \vdash \lambda x : A.M \ : \ A \rightarrow B}$$

$$app \ \frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B}$$

$$prod \ \frac{\Gamma \vdash A \ : \ s \qquad \Gamma \vdash B \ : \ s}{\Gamma \vdash A \rightarrow B \ : \ s}$$

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

Exemples:

$\alpha : *, \beta : * \vdash \alpha \rightarrow \beta : *$

$\vdash \lambda\alpha : *.(\alpha \rightarrow \alpha) : * \rightarrow * : \Box$

$$\begin{array}{ll} \textit{list} & : * \rightarrow * : \Box \\ \textit{nil} & : \Pi A : *.(\textit{list}\ A) \\ \textit{cons} & : \Pi A : *.A \rightarrow (\textit{list}\ A) \rightarrow (\textit{list}\ A) \end{array}$$

INRIA

LF. types dép. de termes

$termes \quad M \quad ::= x \mid c \mid \lambda x : M.M' \mid (M \ M') \mid \Pi x : M.M'$

$$ax \ \vdash * : \Box \qquad var \ \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \ x \notin \Gamma$$

$$aff \ \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B} \ x \notin \Gamma$$

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

$$app \ \frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M \ N) : B[N/x]}$$

$$prod \ \frac{\Gamma \vdash A \ : \ * \qquad \Gamma \vdash B \ : \ s}{\Gamma \vdash \Pi x : A.B \ : \ s}$$

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

$Kinds = \{*, * \to *, A \to *, \cdots\}.$

Exemples:

$$\begin{array}{ll} even & : nat \to * : \Box \\ even_0 & : (even\ 0) \\ even_S & : \Pi n : nat.(even\ n) \to (even\ (S\ (S\ n))) \end{array}$$

$$\begin{array}{ll} list_A & : nat \to * : \Box \\ nil_A & : (list_A\ 0) \\ cons_A & : \Pi n : nat.A \to (list_A\ n) \to (list_A\ (S\ n)) \end{array}$$

Le λ-cube

$$\mathit{termes}\quad M \;::=\; x \mid c \mid \lambda x : M.M' \mid (M\;\; M') \mid \Pi x : M.M'$$

$$\mathit{ax}\quad \vdash * : \square$$

$$\mathit{var}\; \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\; x \notin \Gamma$$

$$\mathit{aff}\; \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\; x \notin \Gamma$$

$$\mathit{lam}\; \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash (\Pi x : A.B) : s}{\Gamma \vdash \lambda x : A.M \;:\; \Pi x : A.B}$$

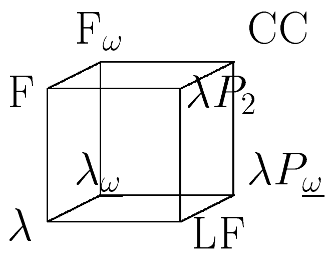
$$\mathit{app}\; \frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\;\; N) : B[N/x]}$$

$$\mathit{prod}\; \frac{\Gamma \vdash A \;:\; s_1 \qquad \Gamma \vdash B \;:\; s_2}{\Gamma \vdash \Pi x : A.B \;:\; s_2}$$

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

$$\{(s_1, s_2)\} \subseteq \{(*, *), (*, \square), (\square, *), (\square, \square)\}$$

Le λ-cube *(suite)*



λ	$(*,*)$	Church (40)
F	$(*,*)$ $(\Box,*)$	Girard (72), Reynolds (74)
LF	$(*,*)$ $(*,\Box)$	de Bruijn (70), HarperHP (87)
$\lambda_{\underline{\omega}}$	$(*,*)$ $(\Box,\Box)$	(polyrec) de Lavalette (91)
λP_2	$(*,*)$ $(\Box,*)$ $(*,\Box)$	Longo & Monggi (88)
F_ω	$(*,*)$ $(\Box,*)$ $(\Box,\Box)$	Girard (72)
$\lambda P_{\underline{\omega}}$	$(*,*)$ $(*,\Box)$ $(\Box,\Box)$	
CC	$(*,*)$ $(\Box,*)$ $(*,\Box)$ $(\Box,\Box)$	Coquant & Huet (88)

Isomorphisme de Curry-Howard:

face gauche	:	logique propositionelle
face droite	:	types dép. de termes $(*,\Box)$. Calcul des prédicats
haut	:	termes dép. types $(\Box,*)$. Ordre sup. Polymorphisme.
fond	:	types dép. de types $(\Box,\Box)$

$$CCind = CC + induction.$$

λ et F dans le λ-cube

λ
$A \to B = \Pi x : A.B$ où $x \notin FV(B)$.
Thm. $\Gamma \vdash A : * \Rightarrow A \in types_\lambda$ & $(A_i : *) \in \Gamma$.

système F
$\forall x.A = \Pi x : *.A$
$\Lambda x.M = \lambda x : *.M$
Les règles *lam* et *Lam* [resp. *app* et *App*] de F sont des cas particuliers de la règle *lam* [resp. *app*] du λ-cube.

Exemples dans le λ-cube

- λ - PROP

  $\vdash \lambda x : A.x : A \to A$

  $y : B \vdash \lambda x : A.y : A \to B$

  $y : B, z : A \vdash (\lambda x : A.y \;\; z) : B$

- F - PROP$_2$. $(\Box, *)$

  $\vdash \lambda \alpha : *.\lambda x : \alpha.x : \Pi \alpha : *.\alpha \to \alpha$

  $\vdash (\lambda \alpha : *.\lambda x : \alpha.x \;\; A) : A \to A$

  $\bot = \Pi P : *.P$

  $\vdash (\lambda A : *.\lambda B : \bot.(B \;\; A) : \Pi A : *.\bot \to A$

- $\lambda_{\underline{\omega}}$ - PROP$_{\underline{\omega}}$. $(\Box, \Box)$

  $\alpha : *, \beta : * \vdash \alpha \to \beta : *$

  $\vdash \lambda \alpha : *.(\alpha \to \alpha) : * \to * : \Box$

  $$\begin{array}{ll} list & : * \to * : \Box \\ nil & : \Pi A : *.(list \;\; A) \\ cons & : \Pi A : *.A \to (list \;\; A) \to (list \;\; A) \end{array}$$

- LF - PRED. $(*, \Box)$

  $A : *, P : (A \to *) \vdash \Pi x : A.(P \;\; x) : *$

  $$\begin{array}{ll} list_A & : nat \to * : \Box \\ nil_A & : (list_A \;\; 0) \\ cons_A & : \Pi n : nat.A \to (list_A \;\; n) \to (list_A \;\; (S \;\; n)) \end{array}$$

Exemples dans le λ-cube *(suite)*

– λP_2 - PRED$_2$. F + LF $(*, \Box)$

$A : *, P : A \to * \vdash \lambda x : A.(P\ x) \to \bot : A \to *$

$A : *, P : A \to A \to * \vdash \Pi x, y : A.$
$(P\ x\ y) \to (P\ y\ x) \to \bot) \to (\Pi x : A.(P\ x\ x) \to \bot : *$

– F$_\omega$ - PROP$_\omega$. F + $\lambda_{\underline{\omega}}$ $(\Box, \Box)$

$\mathrm{sum} = \lambda A, B : *.\Pi C : *.(A \to C) \to (B \to C) \to C :$
$* \to * \to *$

– $\lambda P_{\underline{\omega}}$ - PRED$_{\underline{\omega}}$. $\lambda_{\underline{\omega}}$ + LF $(*, \Box)$

$A : * \vdash \lambda P : (A \to A \to *).\lambda x : A.(P\ x\ x) :$
$(A \to A \to *) \to (A \to *)$

– CC - PRED$_\omega$. F$_\omega$ + LF $(*, \Box)$

$list \quad : * \to nat \to * : \Box$
$nil \quad : \Pi A : *.(list\ A\ 0)$
$cons \quad : \Pi A : *.\Pi n : nat.A \to (list\ A\ n) \to (list\ A\ (S\ n))$

Les Systèmes de Types Purs (PTS)

termes $M \ ::= x \mid c \mid \lambda x : M.M' \mid (M\ M') \mid \Pi x : M.M'$
Un PTS est défini par la donnée d'un triplet $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ où $\mathcal{S} \subset \mathcal{C}$, $\mathcal{A}$ est un ens d'axiomes $(c : s)$ où $c \in \mathcal{C}$ et $s \in \mathcal{S}$, et $\mathcal{R}$ est un ens de triplets (s_1, s_2, s_3) avec $s_i \in \mathcal{S}$.

$$ax \ \vdash c : s \quad (c : s) \in \mathcal{A}$$

$$var \ \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \ x \notin \Gamma$$

$$aff \ \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B} \ x \notin \Gamma$$

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

$$app \ \frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B[N/x]}$$

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

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

Le λ-cube est un sous-ens des PTS, où $(s_1, s_2) = (s_1, s_2, s_2)$.

Propriétés des PTS

Thm *(Subst)* $(\Gamma, x : A, \Gamma_1 \vdash B : C)$ & $(\Gamma \vdash M : A) \Rightarrow$
$\Gamma, \Gamma_1[M/x] \vdash B[M/x] : C[M/x]$.
Thm *(Aff.)* Si $\Gamma \subset \Gamma_1$, alors $\Gamma \vdash M : A \Rightarrow \Gamma_1 \vdash M : A$.
Thm *(Inversion)*

1. $\Gamma \vdash c : A \Rightarrow \exists s \in \mathcal{S}.\ A = s$ & $(c : s) \in \mathcal{A}$
2. $\Gamma \vdash x : A \Rightarrow \exists s \in \mathcal{S}.\exists A' = A.\ \Gamma \vdash A' : s$ & $(x : A') \in \Gamma$
3. $\Gamma \vdash \lambda x : A.M \ : \ C \Rightarrow \exists s \in \mathcal{S}.\exists B.$
   $C = \Pi x : A.B$ & $\Gamma, x : A \vdash M : B$ & $\Gamma \vdash (\Pi x : A.B) : s$
4. $\Gamma \vdash (M\ \ N) : C \Rightarrow \exists A, B.$
   $C = B[N/x]$ & $\Gamma \vdash M : \Pi x : A.B$ & $\Gamma \vdash N : A$
5. $\Gamma \vdash \Pi x : A.B \ : \ C \Rightarrow \exists (s_1, s_2, s_3) \in \mathcal{R}.$
   $C = s_3$ & $\Gamma \vdash A \ : \ s_1$ & $\Gamma, x : A \vdash B \ : \ s_2$

Thm *(SRT)* $\Gamma \vdash M : A$ & $M \hookrightarrow_* M' \Rightarrow \Gamma \vdash M' : A$.
Thm *(Unicité du type)* $\Gamma \vdash M : A$ & $\Gamma \vdash M : B \Rightarrow A = B$. (*)
Thm *(SN)* $\Gamma \vdash M : A \Rightarrow SN(M)$ & $SN(A)$.

(*) L'*unicité du type* n'est vraie que pour les PTS 'singly sorted'.
Def. Un PTS est *'singly sorted'* ssi:
$(c : s), (c, s') \in \mathcal{A} \Rightarrow s = s'$
$(s_1, s_2, s_3), (s_1, s_2, s_3') \in \mathcal{R} \Rightarrow s_3 = s_3'$

Bibliographie

References

[Bar91] Henk Barendregt. Lambda calculi with types. In Samson Abramsky, D. M. Gabbai, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1991.

[Dow97] G. Dowek. Théories des types, 1997. Notes de cours du DEA : Sémantique, preuve et programmation.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[PW99] Christine Paulin and Benjamin Werner. Introduction to Coq. Lectures notes, DEA Paris University and Types summer school'99, August 1999.