Mathématiques, Logique, Types, Réecriture et Automation

Fairouz Kamareddine (Université de Heriot-Watt, Edimbourgh)

6 Mai 2003

Chambéry, le 6 Mai 2003

Le language de Mathématique

D'habitude, le mathématicien ignore la logique formelle. Les mathématiciens écrivent la mathématique avec un language (style) commun qu'on appelle le CML. Les avantages de CML:

- *Expressivité:* On peut exprimer toutes genres de notions.
- *Acceptabilité:* CML est accepté par la plupart des mathématiciens.
- *traditionalité:* CML existe depuis très longtemps et a été raffiné avec le temps.
- *Universalité:* CML est utilisé partout dans le monde.
- *Flexibilité:* Avec CML on peut décrire plusieures branches de mathématiques.

Les désavantages de CML:

- *Informel et ambigu:* CML est basé sur le language naturelle.
- *Incomplet:* Des choses implicites, l'écrivain compte sur l'intuition du lecteur.
- *Pas facile à automatiser* CML
- Au 19ème ciècle, les problèmes en Analyse créaient le besoin d'un style *précis*.
- Plusieures de ces problèmes ont été resolu par le travail de Cauchy (par example par sa définition précise de convergence dans son Cours d'Analyse).
- Les systèmes des nombres sont devenus plus précis avec la définition exacte des nombres réel de Dedekind.
- Cantor commençait la formalisation de la théorie des ensembles et contribuait à la théorie des nombres.

Logique, Theories des Types/Ensembles

- Frege était frustré par les informalités de CML.
- *La definition Genérale de la fonction* était la clef de sa *formalisation de la logique* (1879).
- *L'application d'une fonction à elle-même* était la clef du *paradox de Russell* (1902). Voir [Kamareddine et al., 2002].
- Pour éliminer les paradox, Russell controllait l'application d'une function à un argument par la *theorie des types*.
- Russell (1908) donnait la 1ère theorie des types (RTT).
- Russell and Whitehead utilisaient RTT dans *Principia Mathematica* (1910–1912).
- *theorie simple des types* (STT): Ramsey (1926), Hilbert et Ackermann (1928).

- Fonctions de Frege $\neq$ fonctions du Principia $\neq$ fonctions du *λ-calcul* (1932).
- Frege, Russell et Church écrivait $x \mapsto x + 3$ comme $x + 3$, $\hat{x} + 3$ et $\lambda x.x + 3$.
- Church traitait chaque fonction comme un citoyen de classe 1ère.
- Mais pas nécessaire d'abstraire chaque fonction comme dans le λ-calcul.
- Historiquement, les functions ont toujours été traité comme meta-objets.
- Les $valeurs$ des fonctions etaient le plus important, pas les fonctions abstraites.
- La fonction sinus, est toujours exprimée avec une valeur: $\sin(\pi)$, $\sin(x)$ et des propriétés comme: $\sin(2x) = 2\sin(x)\cos(x)$.
- Dans les cours de mathématiques, on appelle $f(x)$—pas f– la fonction.
- *Le λ-calcul simplement typé* $\lambda\!\!\rightarrow$ de Church $=$ λ-calcul $+$ $\textsc{stt}$ (1940).
- *Problèmes* dans $\textsc{rtt}$ et $\textsc{stt}$. Alors la naissance des *systèmes de typages differents*, chacun avec son *pouvoir d'abstraire des fonctions*. Tous ses

systèmes sont basés sur le λ-calcul.

- *Fonctions qui ne sont pas 1ère classe* nous permettent de rester sur un niveau de typage plus bas (guardant la décidabilité) sans perdre la flexibilité.

- *8 λ-calculs typés importants* 1940–1988 ont été unifié dans le *cube de Barendregt*.

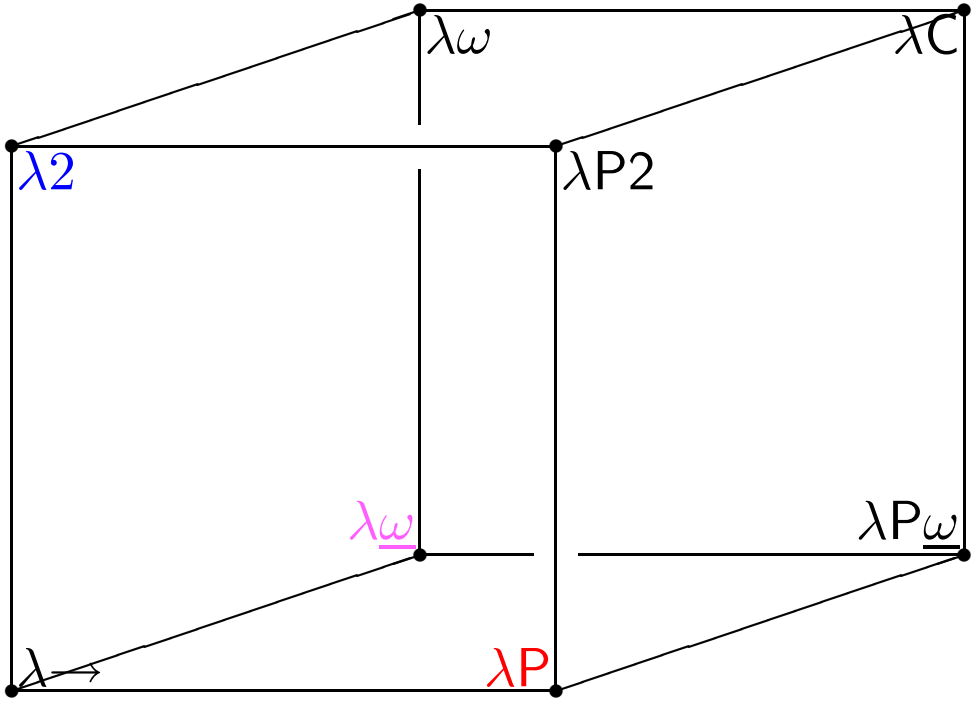
- [Kamareddine et al., 2003, 2001] étend ces calculs avec la notion de fonctions qui ne sont pas 1ère classe, et explique que cette extension pertmette bien de placer des systèmes importants comme le ML de Milner, l'Automath et le LF d'Edimbourgh précisement dans la hierarchie des types.

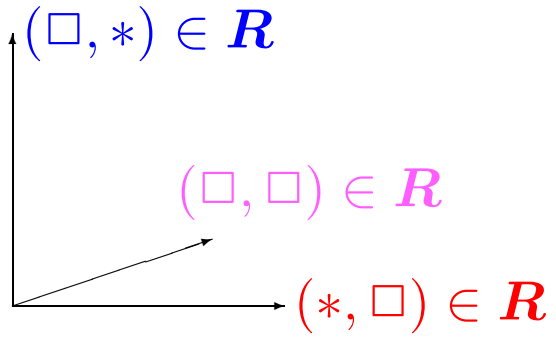
Le cube de Barendregt

- Syntax: $A ::= x \mid * \mid \Box \mid AB \mid \lambda x{:}A.B \mid \Pi x{:}A.B$
- Formation rule: $$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2} \quad \text{if } (s_1, s_2) \in \boldsymbol{R}$$

	Simple	Poly-morphic	Depend-ent	Constr-uctors	Related system	Refs.
$\lambda\!\!\rightarrow$	$(*,*)$				λ^{τ}	[Church, 1940; B
$\lambda 2$	$(*,*)$	$(\Box,*)$			F	[Girard, 1972; Re
λP	$(*,*)$		$(*,\Box)$		AUT-QE, LF	[Bruijn, 1968; Ha
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$	POLYREC	[Renardel de Lava
λP2	$(*,*)$	$(\Box,*)$	$(*,\Box)$			[Longo and Mogg
$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$	Fω	[Girard, 1972]
λP$\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$		
λC	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$	CC	[Coquand and Hu

Le cube de Barendregt





ML

- ML traite let val id $=$ (fn $x \Rightarrow x$) in (id id) end comme le terme
  $(\lambda \mathsf{id}{:}(\Pi\alpha{:}*.\ \alpha \to \alpha).\ \mathsf{id}(\beta \to \beta)(\mathsf{id}\,\beta))(\lambda\alpha{:}*.\ \lambda x{:}\alpha.\ x)$

- On a besoin de la règle $(\Box, *)$ (i.e., $\lambda 2$) pour pouvoir typer ce terme.

- Les règles de typage de ML ne permettent pas le terme:
  let val id $=$ (fn $x \Rightarrow x$) in (fn $y \Rightarrow y\,y$)(id id) end
  Mais, ce terme là est equivalent à un terme de $\lambda 2$ qui est bien-typé:
  $(\lambda \mathsf{id} : (\Pi\alpha{:}*.\ \alpha \to \alpha).$
  $\quad (\lambda y{:}(\Pi\alpha{:}*.\ \alpha \to \alpha).\ y(\beta \to \beta)(y\,\beta))$
  $\quad (\lambda\alpha{:}*.\ \mathsf{id}(\alpha \to \alpha)(\mathsf{id}\,\alpha)))$
  $(\lambda\alpha{:}*.\ \lambda x{:}\alpha.\ x)$

- ML ne doit pas avoir tout le pouvoir de la règle de Π-formation $(\Box, *)$.

LF

- LF [Harper et al., 1987] a un peu le pouvoir de la règle $(*, \Box)$, mais n'est pas equivalent au système λP du cube.

- l'usage de la règle $(*, \Box)$ est très rare.

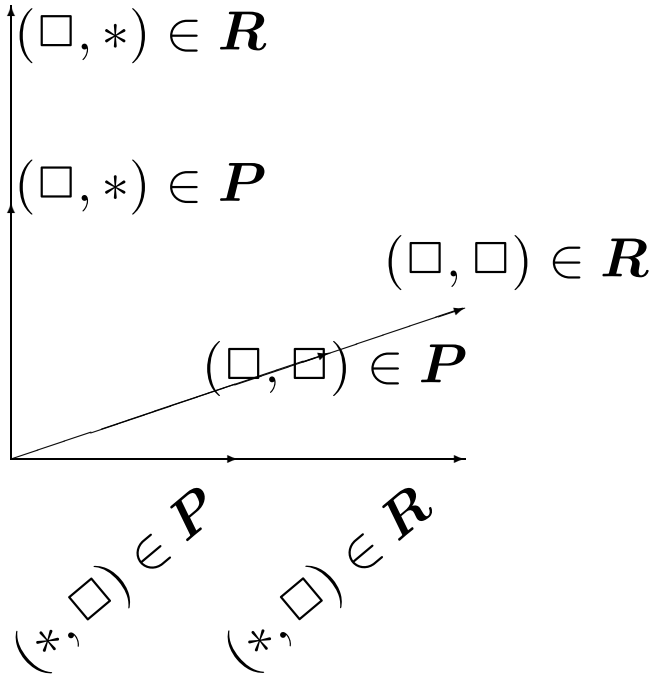
- On a besoin d'un type $\Pi x{:}A.B : \Box$ seulement lorceque le principe: "Propositions-As-Types" PAT s'applique pendant la construction du type $\Pi\alpha{:}\mathtt{prop}.*$ de l'operateur Prf qui pour une proposition Σ, donne Prf(Σ) le type des pereuves de Σ.

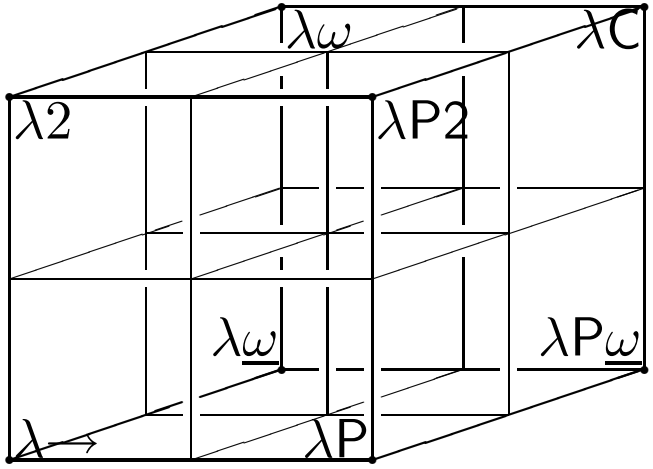
$$\frac{\mathtt{prop}{:}* \vdash \mathtt{prop}: * \qquad \mathtt{prop}{:}*, \alpha{:}\mathtt{prop} \vdash *{:}\Box}{\mathtt{prop}{:}* \vdash \Pi\alpha{:}\mathtt{prop}.* : \Box}.$$

- Dans LF, c'est l'unique place où on utilise la règle $(*, \Box)$.

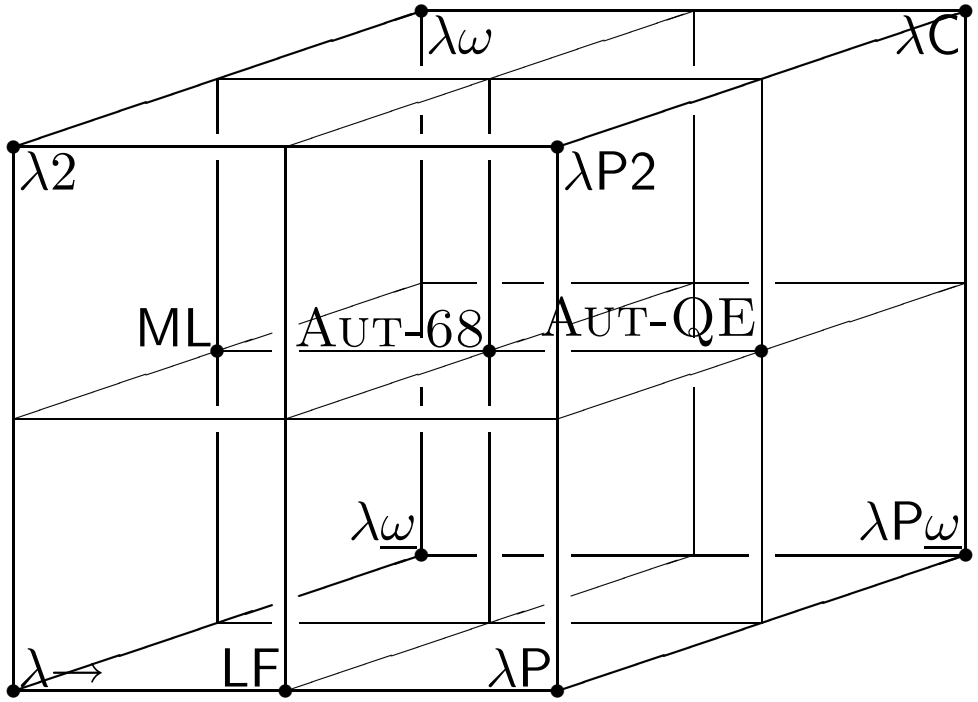
- Mais on utilise Prf seulement avec un $\Sigma{:}\mathtt{prop}$. On n'utilise jamais le Prf seul.

Une version raffinée du cube [Laan, 1997; Kamareddine et al., 2003]





LF, ML, Aut-68, et Aut-QE dans la version raffinée du cube



Le logicien, le mathématicien et l'induction sur les nombres

- Un Logicien utilise `ind: Ind`. Le type `Ind` peut seulement être décrit dans un $\lambda\boldsymbol{R}$ avec $\boldsymbol{R} = \{(*,*),(*,\Box),(\Box,*)\}$:

$$\texttt{Ind} = \Pi p{:}(\mathbb{N}\to *).p0\to(\Pi n{:}\mathbb{N}.\Pi m{:}\mathbb{N}.pn\to Snm\to pm)\to\Pi n{:}\mathbb{N}.pn \quad (1)$$

- Un Mathématicien utilise `ind` seulement avec
  $P : \mathbb{N}\to *$, $Q : P0$ et $R : (\Pi n{:}\mathbb{N}.\Pi m{:}\mathbb{N}.Pn\to Snm\to Pm)$
  pour former un terme $(\texttt{ind}PQR){:}(\Pi n{:}\mathbb{N}.Pn)$.

- L'usage de l'axiom d'induction par le mathématician peut être décrit par le schema (p, q et r sont les *parameters* du schema):

$$\texttt{ind}(p{:}\mathbb{N}\to *, q{:}p0, r{:}(\Pi n{:}\mathbb{N}.\Pi m{:}\mathbb{N}.pn\to Snm\to pm)) : \Pi n{:}\mathbb{N}.pn \quad (2)$$

- Le mathématicien n'a pas besoin du type `Ind` du logicien. Les types dans 2 peuvent être construit dans $\lambda\boldsymbol{R}$ avec $\boldsymbol{R} = \{(*,*)(*,\Box)\}$.

- Un Mathématicien: *applique* l'axiom d'induction et n'a pas besoin de la théorie des preuves derrière.
- Un logicien developpe l'axiom d'induction ou étudie ses propriétés.
- Le mathématicien n'a pas besoin de $(\Box, *)$ tandis que le logicien en a besoin pour pouvoir former l'abstraction $\Pi p{:}(\mathbb{N} \to *).\cdots$.
- Alors le système de types utilisé pour décrire l'usage du mathématicien est plus faible que celui pour le logicien.
- Ce nouveau cube est basé sur deux notions de mathématiques: Les définitions et les paramètres.

La notion de definition [Kamareddine and Bloo, 2002]

(axiom) (app) (abs) et (form) ne changent pas.

(start)
$$\frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A} \qquad \frac{\Gamma \vdash A : s \quad \Gamma \vdash B : A}{\Gamma,\ \mathsf{let}\ x : A\ = B \vdash x : A}$$
x fraiche

(weak)
$$\frac{\Gamma \vdash D : E \quad \Gamma \vdash A : s}{\Gamma, x{:}A \vdash D : E} \qquad \frac{\Gamma \vdash A : s \quad \Gamma \vdash B : A \quad \Gamma \vdash D : E}{\Gamma,\ \mathsf{let}\ x : A\ = B \vdash D : E}$$
x fraiche

(conv)
$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : S \qquad \Gamma \vdash B =_{\mathtt{def}} B'}{\Gamma \vdash A : B'}$$

(def)
$$\frac{\Gamma, \mathsf{let}\ x : A\ = B \vdash C : D}{\Gamma \vdash (\lambda_{x:A}.C)B : D[x := A]}$$

Le Cube avec les paramètres

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

- $\lambda\mathbf{R}\boldsymbol{P} = \lambda\mathbf{R}$ et les deux règles **($\vec{\mathbf{C}}$-weak)** et **($\vec{\mathbf{C}}$-app)**:

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

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

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

Propriétés du Cube Raffiné

- (Correctnesse des types) Si $\Gamma \vdash A : B$ alors $(B \equiv \Box$ or $\Gamma \vdash B : S)$.
- (Réduction du Sujet SR) Si $\Gamma \vdash A : B$ et $A \twoheadrightarrow_\beta A'$ alors $\Gamma \vdash A' : B$
- (Normalisation forte) Pour chaque terme M qui est $\vdash$-legal, on a $\mathrm{SN}_{\twoheadrightarrow_\beta}(M)$.
- On a aussi Unicité des types et typabilité des sous-terms, etc.
- $\lambda\mathbf{R}\boldsymbol{P}$ est le system qui a les règles: de Π-formation $\boldsymbol{R}$ et de paramètre $\boldsymbol{P}$.
- Soit $\lambda\mathbf{R}\boldsymbol{P}$ tel que $(s_1, s_2) \in \boldsymbol{P}$ implique $(s_1, s_2) \in \boldsymbol{R}$.
  - Le système sans paramètre $\lambda\mathbf{R}$ est le plus petit système qui a au moin le pouvoir de $\lambda\mathbf{R}\boldsymbol{P}$.
  - Si $\Gamma \vdash_{\mathbf{R}\boldsymbol{P}} a : A$ alors $\{\Gamma\} \vdash_{\boldsymbol{R}} \{a\} : \{A\}$.

Le système avec définitions aide à résoudre les problèmes

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

- Si on remplace (app) par (n-app) et on ajoute $(\Pi_{x:A}.B)C \rightarrow_{\Pi} B[x := C]$.

$$\text{(n-app)}\quad \frac{\Gamma \vdash F : (\Pi_{x:A}.B) \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : (\Pi_{x:A}.B)a}$$

  Voir [Kamareddine and Nederpelt, 1996; Kamareddine et al., 1999].

- Si on remplace (app) par ($\flat$-app) et on identifie le λ et le Π.

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

  Voir [Kamareddine, 2002].

Les wagons de substitutions [Kamareddine and Nederpelt, 1993]

- $(B)[x]A \rightarrow_\beta [x := B]A$ devient $(B\delta)(\lambda_x)A \rightarrow (B\sigma^x)A$
- Si on utilise les indices de de Bruijn, ca devient: $(B\delta)(\lambda)A \rightarrow (B\sigma^1)A$
- Les σ-wagons propagent les λ- et δ-wagons:

$$\begin{array}{lrcl} \sigma\delta\text{-}transition & (C\sigma^i)(B\delta)A & \rightarrow & ((C\sigma^i)B\delta)(C\sigma^i)A \\ \sigma\lambda\text{-}transition & (C\sigma^i)(\lambda)A & \rightarrow & (\lambda)(C\sigma^{i+1})A \end{array}$$

- On a aussi besoin des φ-wagons qui changent les variables. Ces wagons, eux aussi propagent les λ- et δ-wagons:

$$\begin{array}{lrcl} \varphi\delta\text{-}transition & (\varphi^i_k)(B\delta)A & \rightarrow & ((\varphi^i_k)B\delta)(\varphi^i_k)A \\ \varphi\lambda\text{-}transition & (\varphi^i_k)(\lambda)A & \rightarrow & (\lambda)(\varphi^i_{k+1})A \end{array}$$

Le λs_e- calcul [Kamareddine and Ríos, 1995, 1997]

- Il y avait aussi les règles de destruction des nouveaux wagons, et des règles permettant un σ-wagon (resp., un δ-wagon) à propager les σ- et δ-wagons.
- [Kamareddine and Ríos, 1995] a démontré qu'une restriction λs de ce calcul (qui reflète le calcul lui même de de Bruijn), satisfait les bonnes propriétés sur les termes clos.
- Pour la confluence de λs étendu avec les termes ouverts, [Kamareddine and Ríos, 1997] ajoutait des règles qui reflètent six lemmes sur les propriétés du calcul de de Bruijn donnat λs_e.
- Comme $\lambda\sigma$, λs_e est confluent et n'a pas PSN.
- Mais, $\lambda\sigma$ n'as pas une restriction sur les termes clos satisfaisant PSN, simulation de beta et confluence tandis que λs_e a λs.

Le language de [Kamareddine and Nederpelt, 2003]

- Une logique fait des choix (types/ensembles/categories, predicative/impredicative, etc.). Mais le choix doit dépendre sur la partie de la Mathématique étudiée. Il n'y a pas un formalism acceptable par tout le monde.
- La formalisation de CML par un logicien perd la structure originelle du texte et sert le mathématicien à rien.
- Les mathématiciens n'utilisent pas la logique formelle.
- Aussi dans l'automation, on doit faire des choix: comment implanter l'induction, quel système à utilier (Isabelle, Automath, Coq, etc.).
- Les preuves informelles en Mathématiques sont très difficile à automatiser.
- Alors l'automation n'est pas utile aux mathématiciens.

Le nouveau language NML

- [Kamareddine and Nederpelt, 2003] propose un language de Mathématique NML qui guarde les avantages de CML mais élimine les désavantages.
- Le Mathématicien peut utiliser NML comme un 2ème langue qui a les critères:
- NML est formel et peut être automatisé.
- NML encourage à penser à la dépendence des notions l'une de l'autre.
- NML ne restreint pas le mathématicien à utiliser une théorie définitive (ensembles, types, etc.).
- Contrairement a ces théories, NML possède des notions de bases comme les définitions.
- Même que CML et NML sont incomplets, on peut compléter NML avec des niveaux qui sont de plus en plus complets.

- Déja le 1er chapitre du livre de Landau a été traduit par mon tésard Manuel Maarek. On veut compléter la traduction de ce livre et immédiatement commencer avec les éléments de la géometrie d'Euclid.

Bibliography

H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, revised edition, 1984.

N.G. de Bruijn. The mathematical language AUTOMATH, its usage and some of its extensions. In M. Laudet, D. Lacombe, and M. Schuetzenberger, editors, *Symposium on Automatic Demonstration*, pages 29–61, IRIA, Versailles, 1968. Springer Verlag, Berlin, 1970. Lecture Notes in Mathematics **125**; also in [**?**], pages 73–100.

A. Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

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.

J.R. Hindley and J.P. Seldin. *Introduction to Combinators and λ-calculus*, volume 1 of *London Mathematical Society Student Texts*. Cambridge University Press, 1986.

F. Kamareddine. On functions and types, a tutorial. *SOFSEM'02, LNCS*, 2540:74–93, 2002.

F. Kamareddine and R. Bloo. De Bruijn's syntax and reductional behaviour of lambda terms. *Submitted*, 2002.

F. Kamareddine, R. Bloo, and R. Nederpelt. On Π-conversion in the λ-cube and the combination with abbreviations. *Ann. Pure Appl. Logic*, 97(1–3):27–45, 1999.

F. Kamareddine, L. Laan, and R. P. Nederpelt. Types in logic and mathematics before 1940. *Bulletin of Symbolic Logic*, 8(2):185–245, June 2002.

F. Kamareddine, L. Laan, and R. P. Nederpelt. Revisiting the λ-calculus notion of function. *J. Algebraic & Logic Programming*, 54:65–107, 2003.

F. Kamareddine, L. Laan, and R.P. Nederpelt. Refining the Barendregt cube using parameters. In *Proceedings of the Fifth International Symposium on Functional and Logic Programming, FLOPS 2001*, pages 375–389, 2001.

F. Kamareddine and R.P. Nederpelt. A refinement of de bruijn's formal language of mathematics. *Logic, Language and Information*, 2003. To appear.

Fairouz Kamareddine and Rob Nederpelt. On stepwise explicit substitution. *Int'l J. Foundations Comput. Sci.*, 4(3): 197–240, 1993.

Fairouz Kamareddine and Rob Nederpelt. Canonical typing and Π-conversion in the Barendregt cube. *J. Funct. Programming*, 6(2):245–267, March 1996.

Fairouz Kamareddine and Alejandro Ríos. A λ-calculus à la de Bruijn with explicit substitution. In *7th Int'l Symp. Prog. Lang.: Implem., Logics & Programs, PLILP '95*, volume 982 of *LNCS*, pages 45–62. Springer-Verlag, 1995.

Fairouz Kamareddine and Alejandro Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *J. Funct. Programming*, 7(4):395–420, 1997.

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

G. Longo and E. Moggi. Constructive natural deduction and its modest interpretation. Technical Report CMU-CS-88-131, Carnegie Mellono University, Pittsburgh, USA, 1988.

G.R. Renardel de Lavalette. Strictness analysis via abstract interpretation for recursively defined types. *Information and Computation*, 99:154–177, 1991.

J.C. Reynolds. *Towards a theory of type structure*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer, 1974.