ULTRA: Logique, Types, Reécriture et Applications

Fairouz Kamareddine (Université de Heriot-Watt, Edimbourgh)

28 Avril 2003

Marseille, 28 Avril 2003

Le lambda Calcul à la de Bruijn

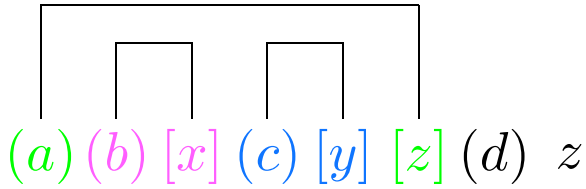
$'$: Notation classique	$\mapsto$	Notation de de Bruijn
x	$\mapsto$	x
$\lambda x.B$	$\mapsto$	$[x]B'$
AB	$\mapsto$	$(B')A'$

Example: $(\lambda x.\lambda y.xy)z \mapsto (z)[x][y](y)x$

- Dans le *train* $(z)[x][y](y)$, les *wagons* sont (z), $[x]$, $[y]$ and (y).
- Le dernier x dans $(z)[x][y](y)x$ est le *coeur* du terme.
- Le *wagon d'application* (z) et le *wagon d'abstraction* $[x]$ se trouvent l'un *à coté* de l'autre.
- La règle β $\quad (\lambda x.A)B \to_\beta A[x := B]$ devient: $(B)[x]A \to_\beta [x := B]A$

Réduction dans la notation de de Bruijn

Notation classique	Notation de de Bruijn
$\underline{((\lambda_x.(\lambda_y.\lambda_z.zd)c)b)a}$	$(a)\underline{(b)[x]}(c)[y][z](d)z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{((\lambda_y.\lambda_z.zd)c)a}$	$(a)\underline{(c)[y]}[z](d)z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{(\lambda_z.zd)a}$	$\underline{(a)[z]}(d)z$
$\downarrow_\beta$	$\downarrow_\beta$
ad	$(d)a$



Chaque wagon a un *partenaire* sauf (d) qui est *célibataire*.

Réductions generalisées

- $(a)(b)[x](c)[y][z](d)z \rightarrow_\beta (a)(b)[x]\{[z](d)z\}[y := c]$
- $(a)(b)[x](c)[y][z](d)z \rightarrow_\beta (a)\{(c)[y][z](d)z\}[x := b]$
- $(a)(b)[x](c)[y][z](d)z \hookrightarrow_\beta \{(b)[x](c)[y](d)z\}[z := a]$

Les Formes Canoniques

- Dans [Kamareddine et al., 2001]:

des [ ] célibataires	des ()[ ]-couples	des ()s célibataires	coeur
$[x_1]\ldots[x_n]$	$(A_1)[y_1]\ldots(A_m)[y_m]$	$(B_1)\ldots(B_p)$	x

- Dans [Regnier, 1994] et [Kfoury and Wells, 1995]

$$\lambda x_1 \cdots \lambda x_n.(\lambda y_1.(\lambda y_2.\cdots(\lambda y_m.xB_p\cdots B_1)A_m\cdots)A_2)A_1$$

- Par example, la forme canonique de:

$$[x][y](a)[z][x'](b)(c)(d)[y'][z'](e)x$$

est

$$[x][y][x'](a)[z](d)[y'](c)[z'](b)(e)x$$

Comment arriver aux formes canoniques?

Règle	Notation classique	Notation de de Bruijn
(θ)	$((\lambda_x.N)P)Q$ $\downarrow$ $(\lambda_x.NQ)P$	$(Q)(P)[x]N$ $\downarrow$ $(P)[x](Q)N$
(γ)	$(\lambda_x.\lambda_y.N)P$ $\downarrow$ $\lambda_y.(\lambda_x.N)P$	$(P)[x][y]N$ $\downarrow$ $[y](P)[x]N$

[Kamareddine and Nederpelt, 1995; Kamareddine et al., 1999, 1998] montrent que ces réductions aident à economiser (temps et espace).

[Kamareddine, 2000] etablit plusieures propriétés des réductions generalisées.

Classes des termes modulo une conduite réductionnel

- $\rightarrow_\theta$ et $\rightarrow_\gamma$ sont SN et CR. Alors θ-nf et γ-nf sont uniques.
- $\theta(\gamma(A))$ et $\gamma(\theta(A))$ sont tous les deux en *forme canonique*.
- $\theta(\gamma(A)) =_p \gamma(\theta(A))$ où $\rightarrow_p$ est la règle

$$(A_1)[y_1](A_2)[y_2]B \rightarrow_p (A_2)[y_2](A_1)[y_1]B \qquad \text{si } y_1 \notin \mathrm{FV}(A_2)$$

- On définit $[A]$ par $\{B \mid \theta(\gamma(A)) =_p \theta(\gamma(B))\}$.
- Quand $B \in [A]$, on écrit que $B \approx_{\mathtt{equi}} A$.
- $\rightarrow_\theta, \rightarrow_\gamma, =_\gamma, =_\theta, =_p \subset \approx_{\mathtt{equi}} \subset =_\beta$ (inclusions strictes).
- Définissons $\mathrm{CCF}(A)$ par $\{A' \text{ en forme canonique} \mid A' =_p \theta(\gamma(A))\}$.

Réduction basée sur les classes

- Un pas de classe-réduction $\leadsto_\beta$ est la plus petite relation qui est compatible tel que:

$$A \leadsto_\beta B \quad \text{ssi} \quad \exists A' \in [A].\exists B' \in [B].\ A' \to_\beta B'$$

- *$\leadsto_\beta$ se comporte comme une réduction de classes:*

- Si $A \leadsto_\beta B$ alors pour tout $A' \approx_{\texttt{equi}} A$, pour tout $B' \approx_{\texttt{equi}} B$, on a $A' \leadsto_\beta B'$.

Propriétés des réductions modulo les classes

- *$\rightsquigarrow_\beta$ generalise $\rightarrow_g$ et $\rightarrow_\beta$:* $\rightarrow_\beta \subset \rightarrow_g \subset \rightsquigarrow_\beta \subset =_\beta$.
- *$\approx_\beta$ et $=_\beta$ sont équivalents:* $A \approx_\beta B$ ssi $A =_\beta B$.
- *$\twoheadrightsquigarrow_\beta$ est Church Rosser:*
  Si $A \twoheadrightsquigarrow_\beta B$ et $A \twoheadrightsquigarrow_\beta C$, alors pour un certain D: $B \twoheadrightsquigarrow_\beta D$ et $C \twoheadrightsquigarrow_\beta D$.
- *Les classes préservent $SN_{\rightarrow_\beta}$:* Si $A \in SN_{\rightarrow_\beta}$ et $A' \in [A]$ alors $A' \in SN_{\rightarrow_\beta}$.
- *Les classes préservent $SN_{\rightsquigarrow_\beta}$:* Si $A \in SN_{\rightsquigarrow_\beta}$ et $A' \in [A]$ alors $A' \in SN_{\rightsquigarrow_\beta}$.
- *$SN_{\rightarrow_\beta}$ et $SN_{\rightsquigarrow_\beta}$ sont équivalents:* $A \in SN_{\rightsquigarrow_\beta}$ ssi $A \in SN_{\rightarrow_\beta}$.

La nouvelle notation dans les systèmes des types

- à la place de $()$ et $[]$, on utilise $()$.
- $(\lambda_x.x)y$ s'écrit comme: $(y\delta)(\lambda_x)x$ à la place de $(y)[x]x$.
- $\Pi_{z:*}.(\lambda_{x:z}.x)y$ s'écrit comme: $(*\Pi_z)(y\delta)(z\lambda_x)x$.

Le cube de Barendregt dans la nouvelle notation et avec la réduction de classe

- La formulation est la même sauf que les termes deviennet:
- $\mathcal{T} = * \mid \Box \mid V \mid (\mathcal{T}\delta)\mathcal{T} \mid (\mathcal{T}\lambda_V)\mathcal{T} \mid (\mathcal{T}\Pi_V)\mathcal{T}$.
- Les règles de typages ne changent pas même que la réduction est la réduction de classe $\rightsquigarrow_\beta$ à la place de la β-réduction $\rightarrow_\beta$.
- Les règles de typages ne changent pas parceque $=_\beta$ est la même relation que $\approx_\beta$.

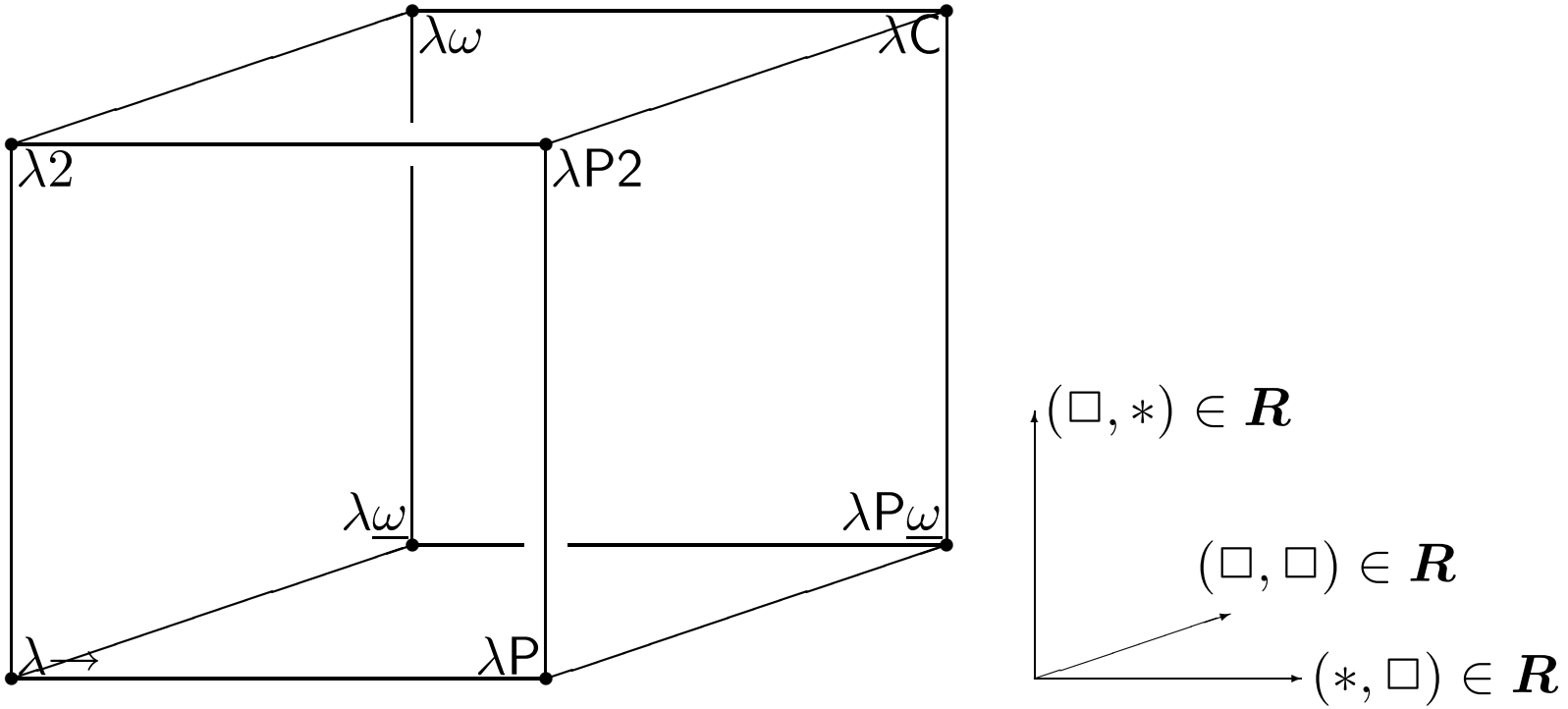


Figure 1: Le cube de Barendregt

On perds le "Subject Reduction"

- Dans le cube étendu avec la réduction modulo les classes, on guarde la plupart des propriétés (inclusif la normalisation forte SN). Hélas, la propriété de "Subject Reduction" n'est plus satisfaite que dans $\lambda_{\rightarrow}$ $(*,*)$ et $\lambda\underline{\omega}$ $(\Box,\Box)$.
- On perds SR dans λP $(*,\Box)$ (et alors dans $\lambda P2, \lambda P\underline{\omega}$ et λC).
- On perds SR dans $\lambda 2$ $(\Box,*)$ (et alors dans $\lambda P2, \lambda\omega$ et λC):

Quelle est la cause de la perte de SR?

- $(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x \rightsquigarrow_\beta (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x$.
- $(\lambda_{\alpha:*}.\lambda_{y:\alpha}.(\lambda_{x:\alpha}.x)y)\beta y' \rightsquigarrow_\beta (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta$
- $\beta : *, y' : \beta \vdash_{\lambda 2} (\lambda_{\alpha:*}.\lambda_{y:\alpha}.(\lambda_{x:\alpha}.x)y)\beta y' : \beta$
- Yet, $\beta : *, y' : \beta \not\vdash_{\lambda 2} (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta : \tau$ pour n'importe quel τ.
- L'information que $y' : \beta$ a remplacé $y : \alpha$ est perdu dans $(\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta$.
- Mais, on a besoin de $y' : \alpha$ pour pouvoir typer le sous-terme $(\lambda_{x:\alpha}.x)y'$ de $(\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta$ et alors pour pouvoir typer $\beta : *, y' : \beta \vdash (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta : \beta$.

La solution du problème de Subject Reduction: on utilise les "let expressions/définitions"

- Définitions/let expressions ont la forme: let $x : A = B$. On les ajoute aux contextes exactement comme les déclarations $y : C$.
- (def rule) $$\frac{\Gamma, \text{let } x : A = B \vdash^{\mathsf{c}} C : D}{\Gamma \vdash^{\mathsf{c}} (\lambda_{x:A}.C)B : D[x := A]}$$
- On definit $\Gamma \vdash^{\mathsf{c}} \cdot =_{\mathtt{def}} \cdot$ comme la relation d'équivalence generée par:
  - Si $A =_\beta B$ alors $\Gamma \vdash^{\mathsf{c}} A =_{\mathtt{def}} B$
  - Si let $x : M = N$ est dans Γ et si B vient de A par substituant une occurrence particulière de x dans A par N, alors $\Gamma \vdash^{\mathsf{c}} A =_{\mathtt{def}} B$.

Le cube (simplifié) avec définitions et réductions de classes

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

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

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

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

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

Les définitions aident à résoudre le subject reduction

1. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} y' : \beta$
2. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} \alpha =_{\mathtt{def}} \beta$
3. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} y' : \alpha$ (de 1 et 2)
4. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta,\ \mathsf{let}\ x : \alpha = y' \quad \vdash^{\mathsf{c}} x : \alpha$
5. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} (\lambda_{x:\alpha}.x)y' : \alpha[x := y'] = \alpha$

$$\beta : *, y' : \beta \quad \vdash^{\mathsf{c}} \quad (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta : \alpha[\alpha := \beta] = \beta$$

Les propriétés du Cube avec définitions et réductions de classes

- $\vdash^{\mathsf{c}}$ est une generalisation de $\vdash$: Si $\Gamma \vdash A : B$ alors $\Gamma \vdash^{\mathsf{c}} A : B$.

- Les termes équivalents ont les mêmes types:
  Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $A' \in [A]$, $B' \in [B]$ alors $\Gamma \vdash^{\mathsf{c}} A' : B'$.

- Subject Reduction de $\vdash^{\mathsf{c}}$ et $\twoheadrightarrow_\beta$:
  Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $A \twoheadrightarrow_\beta A'$ alors $\Gamma \vdash^{\mathsf{c}} A' : B$.

- L'unicité des Types pour $\vdash^{\mathsf{c}}$:
  - Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $\Gamma \vdash^{\mathsf{c}} A : B'$ alors $\Gamma \vdash^{\mathsf{c}} B =_{\mathtt{def}} B'$
  - Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $\Gamma \vdash^{\mathsf{c}} A' : B'$ et $\Gamma \vdash^{\mathsf{c}} A =_\beta A'$ alors $\Gamma \vdash^{\mathsf{c}} B =_{\mathtt{def}} B'$.

- La normalisation forte de $\twoheadrightarrow_\beta$:
  Dans le Cube, chaque terme légale est fortement normalisable par rapport à $\twoheadrightarrow_\beta$.

Bibliography

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.

Fairouz Kamareddine. Postponement, conservation and preservation of strong normalisation for generalised reduction. *J. Logic Comput.*, 10(5):721–738, 2000.

Fairouz Kamareddine, Roel Bloo, and Rob Nederpelt. De Bruijn's syntax and reductional equivalence of lambda terms. In *Proc. 3rd Int'l Conf. Principles & Practice Declarative Programming*, 5–7 September 2001. ISBN 1-58113-388-X.

Fairouz Kamareddine and Rob Nederpelt. Refining reduction in the λ-calculus. *J. Funct. Programming*, 5(4):637–651, October 1995.

Fairouz Kamareddine, Alejandro Ríos, and J. B. Wells. Calculi of generalised β-reduction and explicit substitutions: The type free and simply typed versions. *J. Funct. Logic Programming*, 1998(5), June 1998.

A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi. In *Proc. 10th Ann. IEEE Symp. Logic in Comput. Sci.*, pages 311–321, 1995. ISBN 0-8186-7050-9. URL `http://www.church-project.org/reports/electronic/Kfo+Wel:LICS-1995.pdf.gz`.

L. Regnier. Une équivalence sur les lambda termes. *Theoretical Computer Science*, 126:281–292, 1994.