

Extraction de programme classique dans le calcul des constructions

Alexandre Miquel – U. Paris Diderot, PPS
Alexandre.Miquel@pps.jussieu.fr

1er février 2008
Séminaire LogiCal/ProVal
LIX, Palaiseau

'We build too many walls and not enough bridges.'
— Isaac Newton

Motivation (1/2)

Jeter un pont entre deux conceptions différentes
de la **correspondance de Curry-Howard**:

Les calculs des constructions (1985–)

- Théorie des types, logique intuitionniste
- Forte puissance théorique
- Évaluation forte, repose sur la confluence (conversion)
- Normalisation forte
- Effort collaboratif sur le long terme (Coq)
- Implémentation populaire (“la preuve pour les masses”)
- Extraction automatique (intuitionniste seulement)
- Machinerie complexe, difficile à étendre

Motivation (2/2)

Réalissabilité classique de Krivine (1999–)

- Réalisabilité, logique classique
- Définie pour l'arithmétique du second ordre (PA2)
- S'étend à ZF
- Évaluation faible, pas besoin de confluence
- Création d'un seul mathématicien (conception élitiste)
- Seule la machine virtuelle est implémentée [Legrand-Gérard]
- Termes de preuves construits/extraits à la main
- Supporte l'axiome du choix
- Machinerie légère, facile à étendre (pour ceux qui la comprennent)

... comment faire communiquer les deux mondes?

Un calcul des constructions *proof-irrelevant* ($\text{CC}_{\omega}^{\text{irr}}$)

Sortes	$s ::= \text{Prop} \mid \text{Type}_i \quad (i \geq 1)$
Termes	$M, N, T, U ::= x \mid s \mid \Pi x: T. U$ $\mid \lambda x: T. M \mid MN$

- Pas de distinction syntaxique entre les termes et les types
- $\text{Type}_i =$ terme dont le type est une sorte (i.e. Prop or Type_i)
 - $\text{Type}_i =$ type des **types de données**
 - $\text{Prop} =$ type des **propositions**
- **Proof-irrelevance** obtenue par typage
 - Aucun prédicat ne peut distinguer deux preuves d'une même proposition
 - Correspond à la pratique mathématique (modularité)

Typage: principes

$\vdash \Gamma \text{ ctx}$	Γ est un contexte bien formé
$\Gamma \vdash T \leq T'$	T est un sous-type de T' (dans le contexte Γ)
$\Gamma \vdash M : T$	M a pour type T (dans le contexte Γ)
$\Gamma \vdash M_1 = M_2 : T$	M_1 et M_2 sont des objets égaux de type T

- Typage: règles standard + **subsomption** (\supseteq conversion)

$$\frac{\Gamma \vdash M : T \quad \Gamma \vdash T \leq T'}{\Gamma \vdash M : T'}$$

- Sous-typage: propage **la cumulativité des univers** à travers les Π

$$\text{Prop} \leq \text{Type}_1, \quad \text{Type}_i \leq \text{Type}_{i+1}$$

- Égalité: β + **proof-irrelevance**

$$\frac{\Gamma \vdash M_1 : T \quad \Gamma \vdash M_2 : T \quad \Gamma \vdash T : \text{Prop}}{\Gamma \vdash M_1 = M_2 : T}$$

Typage: les règles (1/3)

- Formation de contexte

$$\frac{}{\vdash \square \text{ ctx}} \qquad \frac{\Gamma \vdash T : s \quad x \notin \text{dom}(\Gamma)}{\vdash \Gamma, x : T \text{ ctx}}$$

- Typage

$$\frac{\vdash \Gamma \text{ ctx}}{\Gamma \vdash x : T} \quad (x:T) \in \Gamma \qquad \frac{\vdash \Gamma \text{ ctx}}{\Gamma \vdash s_1 : s_2} \quad (s_1, s_2) \in \mathcal{A}$$

$$\frac{\Gamma \vdash T : s_1 \quad \Gamma, x : T \vdash U : s_2}{\Gamma \vdash \Pi x : T. M : s_3} \quad (s_1, s_2, s_3) \in \mathcal{R}$$

$$\frac{\Gamma \vdash \Pi x : T. U : s \quad \Gamma, x : T \vdash M : U}{\Gamma \vdash \lambda x : T. M : \Pi x : T. U}$$

$$\frac{\Gamma \vdash M : \Pi x : T. U \quad \Gamma \vdash N : T}{\Gamma \vdash MN : U\{x := N\}}$$

$$\frac{\Gamma \vdash M : T \quad \Gamma \vdash T \leq T'}{\Gamma \vdash M : T'}$$

Typage: les règles (2/3)

- Sous-typage

$$\frac{\Gamma \vdash T = T' : s}{\Gamma \vdash T \leq T'}$$

$$\frac{\Gamma \vdash T \leq T' \quad \Gamma \vdash T' \leq T''}{\Gamma \vdash T \leq T''}$$

$$\frac{\vdash \Gamma \text{ ctx} \quad s_1 \leq s_2}{\Gamma \vdash s_1 \leq s_2}$$

$$\frac{\Gamma \vdash T = T' : s \quad \Gamma \vdash U \leq U'}{\Gamma \vdash \Pi x : T . U \leq \Pi x : T' . U'}$$

- Jugement d'égalité

$$\frac{\Gamma \vdash M : T}{\Gamma \vdash M = M : T}$$

$$\frac{\Gamma \vdash M = M' : T \quad \Gamma \vdash M' = M'' : T}{\Gamma \vdash M = M'' : T}$$

$$\frac{\Gamma \vdash M = M' : T}{\Gamma \vdash M' = M : T}$$

$$\frac{\Gamma \vdash M = M' : T \quad \Gamma \vdash T \leq T'}{\Gamma \vdash M = M' : T'}$$

...

Typage: les règles (3/3)

- Jugement d'égalité (suite et fin)

$$\frac{\Gamma \vdash T = T' : s_1 \quad \Gamma, x : T \vdash U = U' : s_2 \quad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash \Pi x : T. U = \Pi x : T'. U' : s_3}$$

$$\frac{\Gamma \vdash T = T' : s \quad \Gamma, x : T \vdash M = M' : U}{\Gamma \vdash \lambda x : T. M = \lambda x : T'. M' : \Pi x : T. U}$$

$$\frac{\Gamma \vdash M = M' : \Pi x : T. U \quad \Gamma \vdash N = N' : T}{\Gamma \vdash MN = M'N' : U\{x := N\}}$$

$$\frac{\Gamma \vdash \Pi x : T. U : s \quad \Gamma, x : T \vdash M : U \quad \Gamma \vdash N : T}{\Gamma \vdash (\lambda x : T. M)N = M\{x := N\} : U\{x := N\}}$$

$$\frac{\Gamma \vdash T : \text{Prop} \quad \Gamma \vdash M : T \quad \Gamma \vdash M' : T}{\Gamma \vdash M = M' : T}$$

Réalisation classique: le langage des réalisateurs

Le langage λ_c (Krivine)

Termes	$t, u ::= x \mid \lambda x. t \mid tu$ $\mid \mathfrak{c} \mid k_\pi \mid \kappa$	
Piles	$\pi ::= \beta \mid t \cdot \pi$	$(t, \pi \text{ clos})$
Processus	$p, q ::= t \star \pi$	

- \mathfrak{c} = “call with current continuation” (**call/cc**)
- k_π = (constante de) **continuation** associée à la pile π
- κ = constantes diverses (quote, clock, exit, ...)
- β = constantes de piles (**fonds de piles**)
- **Quasi-preuve** = terme (clos) sans k_π

Réalisation classique: évaluation

Évaluation: relation binaire $p \succ p'$ telle que:

Push	$tu \star \pi$	\succ	$t \star u \cdot \pi$
Grab	$\lambda x . t \star u \cdot \pi$	\succ	$t\{x := u\} \star \pi$
Call/cc	$\alpha \star t \cdot \pi$	\succ	$t \star k_\pi \cdot \pi$
Resume	$k_\pi \star t \cdot \pi'$	\succ	$t \star \pi$

- Normalisation de tête faible
- Règles non exhaustives (définition ouverte)
- La confluence est inutile...
- ... mais l'évaluation reste déterministe

(inutile d'un point de vue logique, uniquement pour la spécification)

Réalisation classique: les principes

- **Intuition:** terme = “**preuve**” / pile = “**contre-preuve**”
- Construction paramétrée par un ensemble $\perp \subseteq \Lambda \star \Pi$
clos par anti-évaluation ($p \succ p'$, $p' \in \perp$ impliquent $p \in \perp$)
- Chaque formule A est interprétée par deux ensembles:

Valeur de vérité	$ A \subseteq \Lambda$	=	“preuves” de A
Valeur de fausseté	$\ A\ \subseteq \Pi$	=	“contre-preuves” de A

- Valeur de vérité $|A|$ définie à partir de $\|A\|$ par orthogonalité

$$|A| = \|A\|^\perp = \{t \in \Lambda \mid \forall \pi \in \|A\| \quad t \star \pi \in \perp\}$$

(Valeur de fausseté $\|A\|$ définie de manière primitive)

Arithmétique du 2nd ordre (PA2): le langage

Variables du 1er ordre	$x, y, z \dots$
Variables du 2nd ordre	$X, Y, Z \dots$ (toutes arités)
Symboles de fonc. prim. réc.	$f, g, h \dots$ (0, s, +, \times , $\hat{}$, ...)

Syntaxe (langage de base)

Expr. numériques	$e ::= x \mid f(e_1, \dots, e_n)$
Formules	$A, B ::= X(e_1, \dots, e_n) \mid A \Rightarrow B$ $\mid \forall x A \mid \forall X A$

Langage enrichi = langage de base

+ symbole de prédicat n -aire \dot{F} pour chaque $F : \mathbb{N}^n \rightarrow \mathfrak{P}(\Pi)$

Syntaxe (langage enrichi)

Formules	$A, B ::= \dots \mid \dot{F}(e_1, \dots, e_n)$
-----------------	--

Interprétation des formules closes du langage enrichi

- Valeur de fausseté $\|A\|$ définie par induction sur A :

$$\|\dot{F}(e_1, \dots, e_n)\| = F(\downarrow e_1, \dots, \downarrow e_n)$$

$$\|A \Rightarrow B\| = |A| \cdot \|B\| = \{t \cdot \pi \mid t \in |A|, \pi \in \|B\|\}$$

$$\|\forall x A(x)\| = \bigcup_{n \in \mathbb{N}} \|A(n)\|$$

$$\|\forall X A(X)\| = \bigcup_{F: \mathbb{N}^n \rightarrow \mathfrak{P}(\Pi)} \|A(\dot{F})\|$$

$\downarrow e$ = forme normale de e

- Valeur de vérité $|A|$ définie par orthogonalité:

$$|A| = \|A\|^\perp = \{t \in \Lambda \mid \forall \pi \in \|A\| \quad t \star \pi \in \perp\}$$

Réalisabilité en arithmétique du 2nd ordre: résultats

Lemme: $\varepsilon \in |((A \Rightarrow B) \Rightarrow A) \Rightarrow A|$ (loi de Peirce)

Proposition (Adéquation)

D'une dérivation $d : (A_1, \dots, A_n \vdash B)$ (dans PA2), on peut extraire une quasi-preuve d^* de var. libres x_1, \dots, x_n t.q. pour tous t_1, \dots, t_n

$$t_1 \in |A_1|, \dots, t_n \in |A_n| \Rightarrow d^*\{x_1 := t_1; \dots; x_n := t_n\} \in |B|$$

(indépendemment du choix de $\perp\!\!\!\perp$)

- L'**axiome du choix dépendant** (DC) peut être réalisé en enrichissant le langage des réalisateurs avec une instruction "quote" ou "clock"

Fait: La réalisabilité de Krivine peut s'étendre à CC_ω^{irr}

Π -ensembles

Librement inspiré des ω -sets [Hyland-Longo-Moggi],
des D -sets [Streicher], des λ -sets [Altenkirch], etc.

Définition (Π -ensemble)

Un **Π -ensemble** est un couple $X = \langle |X|, \perp_X \rangle$ formé par

- Un ensemble $|X|$ (**support**)
- Une relation $(\perp_X) \subset |X| \times \Pi$ (**relation d'orthogonalité locale**)

- Intuition: $v \perp_X \pi$ signifie “ π refute v dans X ”
- Réalisabilité: $t \Vdash_X v \equiv t \in (v^{\perp_X})^{\perp}$
- Un Π -ensemble X est **grossier** quand $\perp_X = \emptyset$
($\approx X$ est un ensemble ordinaire)
- Π -ensemble pointé = Π -ensemble avec point distingué

Construction de Π -ensembles

- Produit dépendant d'une famille de Π -ensembles $(Y_v)_{v \in |X|}$
 - $\left| \prod(X, (Y)_{v \in |X|}) \right| = \prod_{x \in |X|} |Y_x|$ (sans restriction)
 - $f \perp_{\prod(X, Y)} \pi$ ssi $\pi \equiv t \cdot \pi'$
où $t \Vdash_X v$ et $f(v) \perp_{Y_v} \pi'$ pour un certain $v \in |X|$

- Si U ensemble d'ensembles ("univers") on note

$$U^{(\Pi)} = \text{ens. de tous les } \Pi\text{-ens. } X \text{ t.q. } |X| \in U$$

- **Π -ensemble dégénéré** $\equiv \Pi$ -ens. X où $|X| = \{\bullet\}$ (singleton)
 - \equiv élément de $\{\{\bullet\}\}^{(\Pi)}$
 - \equiv valeur de fausseté S avec structure de Π -ens. triviale: $\langle \{\bullet\}, \{\bullet\} \times S \rangle$

Construction du modèle

- Même construction que pour le modèle avec ω -sets [Longo-Moggi], en remplaçant les ω -sets par les Π -ensembles
- Propositions interprétées comme des Π -ensembles dégénérés
- Termes de preuve interprétés par \bullet
- $\llbracket \text{Prop} \rrbracket = \text{coarse}(\{\{\bullet\}\}^{(\Pi)})$
- $\llbracket \text{Type}_i \rrbracket = \text{coarse}\left(\left(V_{\lambda_i} \setminus \{\emptyset\}\right)^{(\Pi)}\right) \quad (\lambda_i \equiv \text{cardinal inaccessible } n^{\circ} i)$
- Produit dépendant interprété comme précédemment
- Abstraction/application interprétées de manière ensembliste
[+ Codage d'Aczel pour identifier $(v \in |X| \mapsto \bullet)$ avec \bullet]

Fonction d'interprétation

- $\mathcal{M} = \bigcup_{i \geq 1} V_{\lambda_i}$
- Valuation = fonction $\rho \in \mathcal{M}^{\text{Var}}$

Fonction d'interprétation: $\llbracket M \rrbracket : \mathcal{M}^{\text{Var}} \rightarrow \mathcal{M}$

$$\llbracket x \rrbracket_{\rho} = \rho(x)$$

$$\llbracket \text{Prop} \rrbracket_{\rho} = \text{coarse}(\{\{\bullet\}\}^{\langle \Pi \rangle})$$

$$\llbracket \text{Type}_i \rrbracket_{\rho} = \text{coarse}((V_{\lambda_i} \setminus \{\emptyset\})^{\langle \Pi \rangle})$$

$$\llbracket \Pi x : T . U \rrbracket_{\rho} = \Pi v : \llbracket T \rrbracket_{\rho} . \llbracket U \rrbracket_{\rho; x \leftarrow v} \quad (\text{produit de } \Pi\text{-ens.})$$

$$\llbracket \lambda x : T . M \rrbracket_{\rho} = (v \in \llbracket T \rrbracket_{\rho} \mid \mapsto \llbracket M \rrbracket_{\rho; x \leftarrow v})$$

$$\llbracket MN \rrbracket_{\rho} = \llbracket M \rrbracket_{\rho}(\llbracket N \rrbracket_{\rho})$$

Correction

Proposition (Correction)

Si $\Gamma \vdash M : T$, alors pour toute valuation $\rho \in \llbracket \Gamma \rrbracket$:

- 1 $\llbracket T \rrbracket_\rho$ est un Π -ensemble
- 2 $\llbracket M \rrbracket_\rho \in |\llbracket T \rrbracket_\rho|$

Corollaire

L'interprétation d'une proposition $A : \text{Prop}$ est un Π -ensemble dégénéré

$$\llbracket A \rrbracket = \langle \{\bullet\}, S_A \rangle \quad (S_A \subseteq \Pi)$$

Adéquation

La fonction d'extraction $M \mapsto M^*$ ($CC_\omega^{\text{irr}} \rightarrow \lambda_c$)

$$\begin{aligned}
 x^* &= x & (\lambda x : T. M)^* &= \lambda x. M^* & (MN)^* &= M^* N^* \\
 \text{Prop}^* &= \text{Type}_i^* &= (\Pi x : T. U)^* &= \lambda z. z
 \end{aligned}$$

Proposition (Adéquation)

Si $x_1 : T_1, \dots, x_n : T_n \vdash M : U$ (dans CC_ω^{irr}),

alors pour tout $\rho \in \llbracket \Gamma \rrbracket$, pour tous $v_1 \in \llbracket T_1 \rrbracket_\rho, \dots, v_n \in \llbracket T_n \rrbracket_\rho$
 et pour tous termes $t_1 \Vdash_{\llbracket T_1 \rrbracket_\rho} v_1, \dots, t_n \Vdash_{\llbracket T_n \rrbracket_\rho} v_n$

$$M^* \{x_1 := t_1; \dots; x_n := t_n\} \Vdash_{\llbracket U \rrbracket_\rho} \llbracket M \rrbracket_\rho$$

(indépendamment du choix de $\perp\!\!\!\perp$)

Extensions du mécanisme d'extraction

Le mécanisme de base (retour à la syntaxe)

- Soit T un type (ou une proposition) pour qui on a
 - une dénotation $v \in \llbracket T \rrbracket$ et
 - une quasi-preuve close $t \Vdash_{\llbracket T \rrbracket} \bullet$

- On ajoute à CC_{ω}^{irr} une constante fraîche $c : T$ et on pose

$$\llbracket c \rrbracket = v \quad \text{and} \quad c^* = t$$

- La correction et l'adéquation restent vraies dans $CC_{\omega}^{\text{irr}} + c : T$

... On peut aussi importer dans le jugement d'égalité toute équation satisfaite dans le modèle

Exemple: la loi de Peirce

$$T = \Pi A, B : \text{Prop}. ((A \rightarrow B) \rightarrow A) \rightarrow A$$

On pose $c : T$ avec $\llbracket c \rrbracket = \bullet$ et $c^* = \lambda _ . _ . \text{cc}$

Les entiers naturels

On étend $CC_{\omega}^{\text{irr}} + \text{Peirce}$ avec les constantes

$$\begin{aligned} \text{nat} & : \text{Type}_1 & 0 & : \text{nat} & s & : \text{nat} \rightarrow \text{nat} \\ \text{nat_ind} & : \prod X : \text{nat} \rightarrow \text{Prop} . (X0 \rightarrow \prod y : \text{nat} . (Xy \rightarrow X(sy))) \rightarrow \prod x : \text{nat} . Xx \\ \text{nat_rec}_i & : \prod X : \text{nat} \rightarrow \text{Type}_i . (X0 \rightarrow \prod y : \text{nat} . (Xy \rightarrow X(sy))) \rightarrow \prod x : \text{nat} . Xx \end{aligned}$$

interprétées et réalisées par

$$\begin{aligned} \llbracket \text{nat} \rrbracket & = \langle \mathbb{N}, \perp_{\mathbb{N}} \rangle & \text{où } n \perp_{\mathbb{N}} \pi & \text{ssi } \pi \in \llbracket \text{Nat}(n) \rrbracket & \text{(codage au 2nd ordre)} \\ \llbracket 0 \rrbracket & = 0 & \llbracket s \rrbracket & = (n \mapsto n + 1) \end{aligned}$$

$$\begin{aligned} \text{nat}^* & = \lambda z . z & \text{(ou n'importe quelle quasi-preuve)} \\ 0^* & = \lambda x f . x & s^* & = \lambda n x f . f(n x f) \\ \text{nat_ind}^* & = \lambda _ x f n . n (\lambda z . z 0^* x) (\lambda p . p (\lambda m y z . z (s^* m) (f m y))) (\lambda x y . y) \\ \text{nat_rec}_i^* & = \text{nat_ind}^* \end{aligned}$$

Les égalités attendues pour nat_rec_i sont vraies dans le modèle

Conclusion

- La réalisabilité classique de Krivine dans PA2 peut s'étendre à $CC_{\omega}^{\text{irr}} + \text{Peirce} + \text{nat}$
- Extraction classique \neq Extraction actuelle dans Coq
- Les réalisateurs coïncident sur le fragment commun
- On peut importer les résultats de réalisabilité classique dans CC_{ω}^{irr}

Travaux futurs

- Extension aux types inductifs et points fixes (Coq)
- Mieux comprendre les réalisateurs classiques
- Implémentation et expérimentation