Programmer = démontrer?

La correspondance de Curry-Howard aujourd'hui

Quatrième cours

Il faut qu'une porte soit ouverte ou fermée! Logique classique, continuations, et opérateurs de contrôle

Xavier Leroy

Collège de France

2018-12-05

COLLÈGE
DE FRANCE
1530

Objectif du cours

Essayer de répondre à la question

> *Si la logique intuitionniste correspond au lambda-calcul typé,*
> *à quoi correspond la logique classique ?*

Pour ce faire :

- Mieux comprendre la logique classique en relation avec la logique intuitionniste.
- Comprendre les ≪opérateurs de contrôle≫ (`call/cc` et al)
  ($\approx$ le `goto` en lambda-calcul).

I

Logique classique, logique intuitionniste

Logique classique, logique intuitionniste

Dans le premier cours, nous avons introduit la logique intuitionniste comme la logique classique ≪moins≫ certains principes :

- le tiers exclu $P \vee \neg P$, c.à.d. tout P est soit vrai soit faux ;
- $\neg\neg P \Rightarrow P$, c.à.d. le raisonnement par l'absurde.

Cette idée choque de nombreux mathématiciens, à commencer par Hilbert qui, en 1927, règle son compte à Brouwer en ces termes :

> *Taking the principle of excluded middle from the mathematician would be the same, say, as proscribing the telescope to the astronomer or to the boxer the use of his fists. To prohibit existence statements and the principle of excluded middle is tantamount to relinquishing the science of mathematics altogether.*
>
> (Hilbert, *The foundations of mathematics*, 1927 ; dans van Heijenoort, *From Frege to Gödel : A Source Book in Mathematical Logic*, 1976)

Logique classique, logique intuitionniste

Continuons à choquer : prenons la logique intuitionniste comme point de départ bien compris et utilisons-la pour mieux comprendre la logique classique.

- La logique classique comme la logique intuitionniste ≪plus≫ certaines lois et certaines symétries ;
- La logique classique comme fragment de la logique intuitionniste (modulo des codages).

Des lois classiques

Les propositions ci-dessous ($\forall P, Q, R$) sont :

- vraies en logique classique (dixit leurs tables de vérité) ;
- non prouvables mais équivalentes en logique intuitionniste.

$P \vee \neg P$	tiers exclu, *tertium non datur*, *excluded middle*
$\neg\neg P \Rightarrow P$	élimination de la double négation
$(\neg P \Rightarrow P) \Rightarrow P$	*mirabile consequentia*, loi de Clavius
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	loi de Peirce
$(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$	principe du contre-exemple
$P \vee (P \Rightarrow Q)$	formule de Tarski
$(P \Rightarrow Q) \vee (Q \Rightarrow R)$	principe de linéarité

Vision : logique classique = logique intuitionniste + (une de) ces lois.

Des lois classiques

Les propositions ci-dessous ($\forall P, Q, R$) sont :

- vraies en logique classique (dixit leurs tables de vérité) ;
- non prouvables mais équivalentes en logique intuitionniste.

$P \vee \neg P$	tiers exclu, *tertium non datur*, *excluded middle*
$\neg\neg P \Rightarrow P$	élimination de la double négation
$(\neg P \Rightarrow P) \Rightarrow P$	*mirabile consequentia*, loi de Clavius
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	loi de Peirce
$(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$	principe du contre-exemple
$P \vee (P \Rightarrow Q)$	formule de Tarski
$(P \Rightarrow Q) \vee (Q \Rightarrow R)$	principe de linéarité

Vision : logique classique = logique intuitionniste + (une de) ces lois.

Des lois classiques

Les propositions ci-dessous ($\forall P, Q, R$) sont :

- vraies en logique classique (dixit leurs tables de vérité) ;
- non prouvables mais équivalentes en logique intuitionniste.

$P \vee \neg P$	tiers exclu, *tertium non datur*, *excluded middle*
$\neg\neg P \Rightarrow P$	élimination de la double négation
$(\neg P \Rightarrow P) \Rightarrow P$	*mirabile consequentia*, loi de Clavius
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	loi de Peirce
$(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$	principe du contre-exemple
$P \vee (P \Rightarrow Q)$	formule de Tarski
$(P \Rightarrow Q) \vee (Q \Rightarrow R)$	principe de linéarité

Vision : logique classique = logique intuitionniste + (une de) ces lois.

Des lois classiques

Les propositions ci-dessous ($\forall P, Q, R$) sont :

- vraies en logique classique (dixit leurs tables de vérité) ;
- non prouvables mais équivalentes en logique intuitionniste.

$P \vee \neg P$	tiers exclu, *tertium non datur*, *excluded middle*
$\neg\neg P \Rightarrow P$	élimination de la double négation
$(\neg P \Rightarrow P) \Rightarrow P$	*mirabile consequentia*, loi de Clavius
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	loi de Peirce
$(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$	principe du contre-exemple
$P \vee (P \Rightarrow Q)$	formule de Tarski
$(P \Rightarrow Q) \vee (Q \Rightarrow R)$	principe de linéarité

Vision : logique classique = logique intuitionniste + (une de) ces lois.

Des lois classiques

Les propositions ci-dessous ($\forall P, Q, R$) sont :

- vraies en logique classique (dixit leurs tables de vérité) ;
- non prouvables mais équivalentes en logique intuitionniste.

$P \vee \neg P$	tiers exclu, *tertium non datur*, *excluded middle*
$\neg\neg P \Rightarrow P$	élimination de la double négation
$(\neg P \Rightarrow P) \Rightarrow P$	*mirabile consequentia*, loi de Clavius
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	loi de Peirce
$(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$	principe du contre-exemple
$P \vee (P \Rightarrow Q)$	formule de Tarski
$(P \Rightarrow Q) \vee (Q \Rightarrow R)$	principe de linéarité

Vision : logique classique = logique intuitionniste + (une de) ces lois.

Des lois classiques

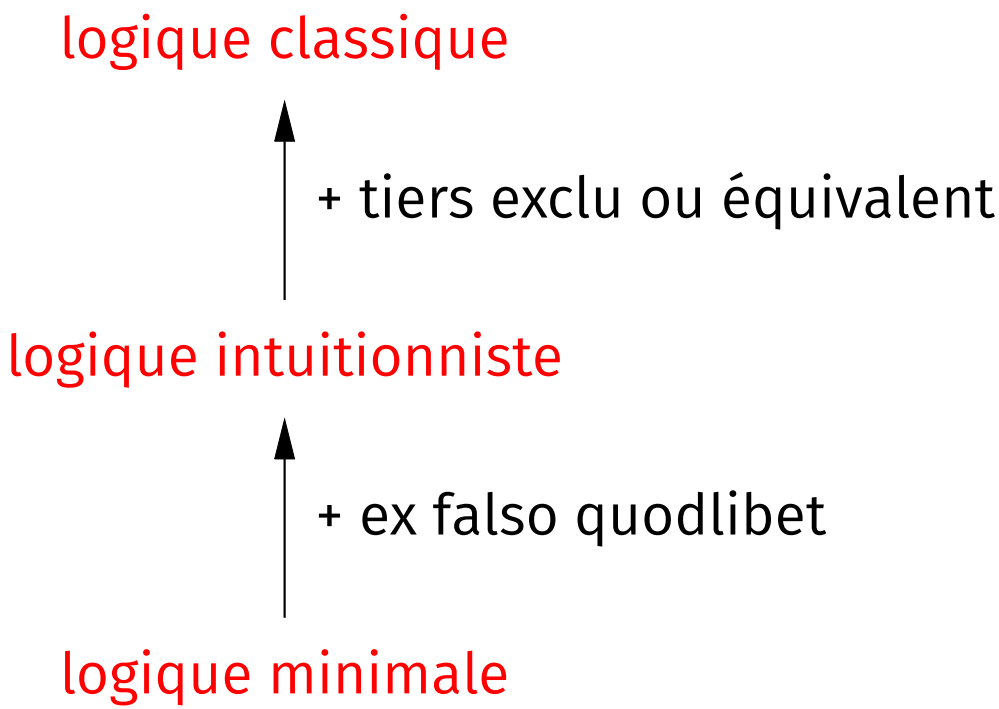
Les propositions ci-dessous ($\forall P, Q, R$) sont :

- vraies en logique classique (dixit leurs tables de vérité) ;
- non prouvables mais équivalentes en logique intuitionniste.

$P \vee \neg P$	tiers exclu, *tertium non datur*, *excluded middle*
$\neg\neg P \Rightarrow P$	élimination de la double négation
$(\neg P \Rightarrow P) \Rightarrow P$	*mirabile consequentia*, loi de Clavius
$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$	loi de Peirce
$(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$	principe du contre-exemple
$P \vee (P \Rightarrow Q)$	formule de Tarski
$(P \Rightarrow Q) \vee (Q \Rightarrow R)$	principe de linéarité

Vision : logique classique = logique intuitionniste + (une de) ces lois.

Logique classique, intuitionniste, minimale



Des lois classiques

En logique minimale (sans le *ex falso quodlibet*, $\bot \Rightarrow P$), ces lois classiques ne sont pas toutes équivalentes :

élimination double négation $\Leftrightarrow$ principe du contre-exemple

$\neg\neg P \Rightarrow P$ $\qquad$ $(\neg(P \Rightarrow Q)) \Rightarrow P \wedge \neg Q$

$\Downarrow$

loi de Peirce $\Leftrightarrow$ formule de Tarski $\Leftrightarrow$ linéarité

$((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$ $\qquad$ $P \vee (P \Rightarrow Q)$ $\qquad$ $(P \Rightarrow Q) \vee (Q \Rightarrow R)$

$\Downarrow$

tiers exclu $\Leftrightarrow$ loi de Clavius

$P \vee \neg P$ $\qquad$ $(\neg P \Rightarrow P) \Rightarrow P$

$\Downarrow$

tiers exclu faible $\Leftrightarrow$ loi de de Morgan

$\neg P \vee \neg\neg P$ $\qquad$ $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

(Diener et McKubre-Jordens, *Classifying Material Implications over Minimal Logic*, 2018)

Lois de de Morgan et dualité

Les lois qui relient «et», «ou», «non» (lois de de Morgan) ne sont pas toutes vraies en logique intuitionniste :

Classique	Intuitionniste
$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$	$\Leftarrow$
$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$	$\Leftrightarrow$
$\neg(\neg P \wedge \neg Q) \Leftrightarrow P \vee Q$	$\Leftarrow$
$\neg(\neg P \vee \neg Q) \Leftrightarrow P \wedge Q$	$\Leftarrow$

Vision : logique classique = dualité entre $\wedge$ et $\vee$ via $\neg$

Traductions l. classique $\rightarrow$ l. intuitionniste

Vision : la logique intuitionniste comme moyen d'étude de la logique classique.

Une approche initiée par Gödel vers 1933 pour montrer la cohérence de l'arithmétique classique (arithmétique de Peano) :

- Définir une traduction $[\![\cdot]\!]$ des formules logiques φ telle que :
  si $L.C. \vdash \varphi$ alors $L.I. \vdash [\![\varphi]\!]$;
  si φ est une contradiction alors $[\![\varphi]\!]$ est une contradiction.
- Montrer la cohérence de l'arithmétique intuitionniste (arithmétique de Heyting), à l'aide d'interprétations de type BHK.

Les traductions $[\![\cdot]\!]$ sont généralement à base de double négation.

La double négation

La double négation $\neg\neg P$ se lit comme ≪c'est pas faux que P≫.

En logique classique, $\neg\neg P$ équivaut à P,
et ≪c'est pas faux≫ équivaut à ≪c'est vrai≫.

En logique intuitionniste, P implique $\neg\neg P$ mais la réciproque est fausse (pas d'élimination de la double négation). Donc ≪c'est pas faux≫ est plus faible que ≪c'est vrai≫.

En revanche, le ≪c'est pas faux≫ intuitionniste se comporte presque comme le ≪c'est vrai≫ classique. En particulier, le tiers exclu n'est pas faux !

La double négation

Théorème (Le tiers exclu n'est pas faux)

$\neg\neg(P \vee \neg P)$ *en logique intuitionniste.*

Démonstration.

Supposons $\neg(P \vee \neg P)$ et montrons une absurdité $\bot$.
Par la contraposée (si $A \Rightarrow B$ alors $\neg B \Rightarrow \neg A$) et par $P \Rightarrow P \vee \neg P$, on obtient $\neg P$.
Par la contraposée et par $\neg P \Rightarrow P \vee \neg P$ on obtient $\neg\neg P$.
D'où l'absurdité. □

Traduction négative : cas propositionnel

Théorème (Glivenko, 1929)

Soit Φ une formule propositionnelle.
$L.C. \vdash \Phi$ si et seulement si $L.I. \vdash \neg\neg\Phi$.

Démonstration.

Si $L.I. \vdash \neg\neg\Phi$, alors $L.C. \vdash \neg\neg\Phi$ et donc, classiquement, $L.C. \vdash \varphi$.

Réciproquement, supposons $L.C. \vdash \Phi$. Soient $P_1, \ldots, P_n$ les variables de Φ. Par la technique des tables de vérité,

$$\Psi \stackrel{def}{=} P_1 \vee \neg P_1 \Rightarrow \cdots \Rightarrow P_n \vee \neg P_n \Rightarrow \Phi$$

est vraie en L.I., et donc sa double négation $\neg\neg\Psi$ aussi. Or,

$$\neg\neg\Psi \iff (\neg\neg(P_1 \vee \neg P_1) \Rightarrow \cdots \Rightarrow \neg\neg(P_n \vee \neg P_n) \Rightarrow \neg\neg\Phi)$$

Les prémisses $\neg\neg(P_i \vee \neg P_i)$ étant vraies en L.I., on a $L.I. \vdash \neg\neg\Phi$. □

Traduction négative : ajout des quantificateurs

Avec les quantificateurs $\forall$, $\exists$, ajouter un $\neg\neg$ en tête ne suffit plus.
Kolmogorov (1925) propose d'ajouter $\neg\neg$ devant chaque sous-formule.

$$[\![A]\!] = \neg\neg A \text{ si } A \text{ est atomique} \qquad [\![P \Rightarrow Q]\!] = \neg\neg([\![P]\!] \Rightarrow [\![Q]\!])$$
$$[\![P \wedge Q]\!] = \neg\neg([\![P]\!] \wedge [\![Q]\!]) \qquad [\![P \vee Q]\!] = \neg\neg([\![P]\!] \vee [\![Q]\!])$$
$$[\![\forall x.\, P]\!] = \neg\neg\forall x.\, [\![P]\!] \qquad [\![\exists x.\, P]\!] = \neg\neg\exists x.\, [\![P]\!]$$

Théorème

Si $L.C. \vdash P$ alors $L.I. \vdash [\![P]\!]$.

Comme $\neg\neg\bot \Rightarrow \bot$, il s'ensuit que si $L.C. \vdash \bot$ alors $L.I. \vdash \bot$.
Donc si L.I. est cohérente, L.C. est cohérente.

Traduction négative : variantes

Gödel (1933) et Gentzen (1936) utilisent une traduction plus ≪économe≫, le $\neg\neg$ n'étant pas nécessaire devant $\wedge$, $\Rightarrow$ et $\forall$.

$$\begin{aligned} [\![A]\!] &= \neg\neg A & [\![P \Rightarrow Q]\!] &= [\![P]\!] \Rightarrow [\![Q]\!] \\ [\![P \wedge Q]\!] &= [\![P]\!] \wedge [\![Q]\!] & [\![P \vee Q]\!] &= \neg\neg([\![P]\!] \vee [\![Q]\!]) \\ [\![\forall x.\, P]\!] &= \forall x.\, [\![P]\!] & [\![\exists x.\, P]\!] &= \neg\neg\exists x.\, [\![P]\!] \end{aligned}$$

Kuroda (1951) est plus ≪économe≫ encore : $[\![P]\!] = \neg\neg P^*$ avec

$$\begin{aligned} A^* &= A & (P \Rightarrow Q)^* &= P^* \Rightarrow Q^* \\ (P \wedge Q)^* &= P^* \wedge Q^* & (P \vee Q)^* &= P^* \vee Q^* \\ (\forall x.\, P)^* &= \forall x.\, \neg\neg P^* & (\exists x.\, P)^* &= \exists x.\, P^* \end{aligned}$$

(G. Ferreira, P. Oliva. *On Various Negative Translations*. EPTCS 47, 2011.)

Négation relative

À la manière de H. Friedman :

On remplace l'absurdité $\bot$ par une proposition R de notre choix, et donc on remplace la négation $\neg P \overset{def}{=} P \Rightarrow \bot$ par la négation relative

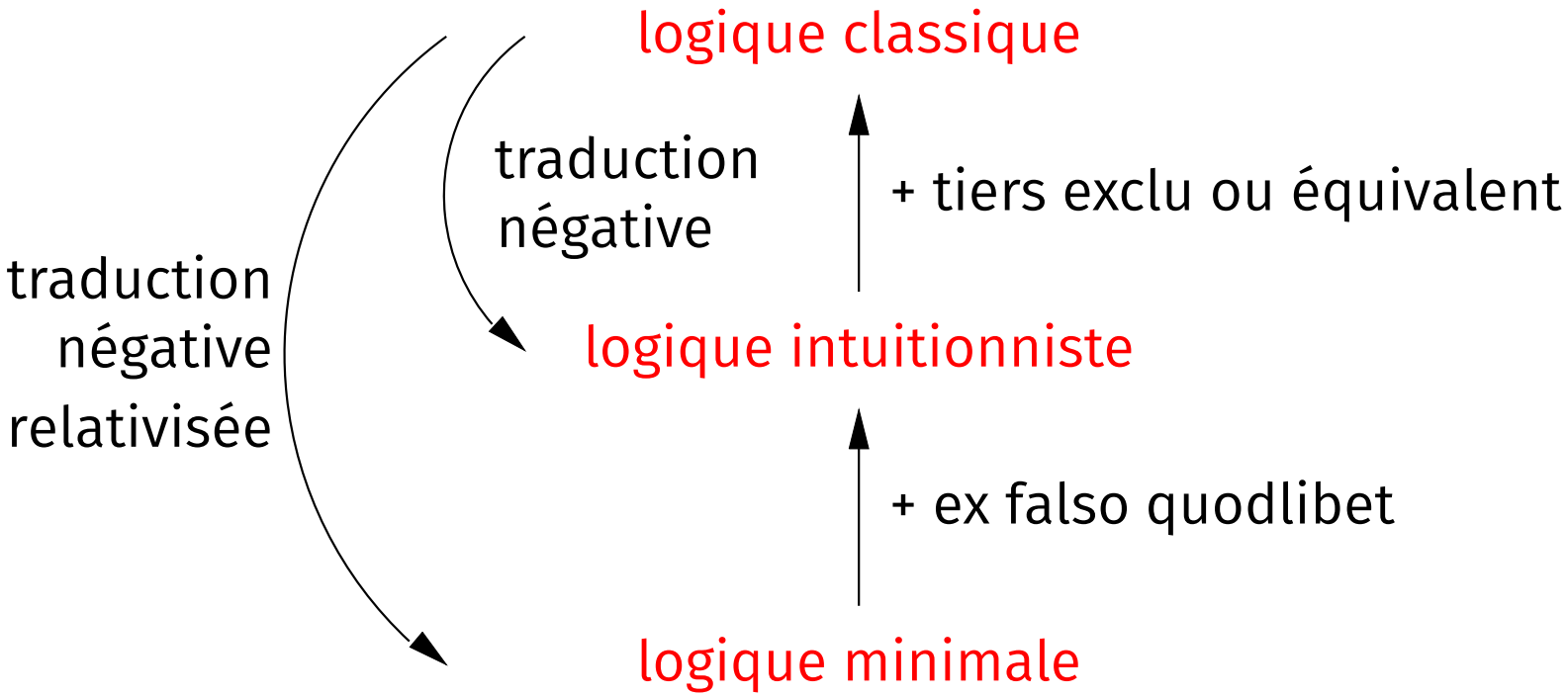
$$\neg_R P \quad \overset{def}{=} \quad P \Rightarrow R$$

obtenant ainsi une traduction négative relative :

$$\begin{aligned}
[\![A]\!]_R &= \neg_R \neg_R A & [\![P \Rightarrow Q]\!]_R &= \neg_R \neg_R ([\![P]\!]_R \Rightarrow [\![Q]\!]_R) \\
[\![P \wedge Q]\!]_R &= \neg_R \neg_R ([\![P]\!]_R \wedge [\![Q]\!]_R) & [\![P \vee Q]\!]_R &= \neg_R \neg_R ([\![P]\!]_R \vee [\![Q]\!]_R) \\
[\![\forall x.\, P]\!]_R &= \neg_R \neg_R \forall x.\, [\![P]\!]_R & [\![\exists x.\, P]\!]_R &= \neg_R \neg_R \exists x.\, [\![P]\!]_R
\end{aligned}$$

Il est maintenant clair que l'on cible la logique minimale (sans symbole $\bot$ ni sa règle ex falso quodlibet).

Panorama des traductions négatives



La double négation relative

Préserve les bonnes propriétés de la double négation :
(ces propriétés sont prouvables en logique minimale)

$$
\begin{array}{rcl}
P & \Rightarrow & \neg_R \neg_R P \\
\neg_R \neg_R \neg_R \neg_R P & \Rightarrow & \neg_R \neg_R P \\
\neg_R \neg_R P \Rightarrow \neg_R \neg_R Q & \Leftrightarrow & \neg_R \neg_R (P \Rightarrow Q) \\
\neg_R \neg_R P \wedge \neg_R \neg_R Q & \Leftrightarrow & \neg_R \neg_R (P \wedge Q) \\
\neg_R \neg_R P \vee \neg_R \neg_R Q & \Rightarrow & \neg_R \neg_R (P \vee Q)
\end{array}
$$

Valide le tiers exclu ≪relativisé≫ : $\neg_R \neg_R (P \vee \neg_R P)$.

Valide l'élimination de la double négation de R : $\neg_R \neg_R R \Rightarrow R$.

Conservativité

Théorème

Si L.C. $\vdash P$ *alors L.I.* $\vdash [\![P]\!]_R$.

Corollaire

Si $[\![R]\!]_R$ *implique R, et si L.C.* $\vdash R$*, alors L.I.* $\vdash R$.

L'hypothèse $\ll [\![R]\!]_R$ implique $R \gg$ est vraie dans de nombreux cas :

- R est une formule atomique A de la forme $f(x_1, \ldots, x_n) = 0$
- R est $A_1 \vee A_2$
- R est $\exists x_1, \ldots, x_n . A$ (formule Σ_1^0)
- R est $\forall x_1, \ldots x_n . \exists y_1 \ldots y_m . A$ (formule Π_2^0).

Théorème (Friedman)

Toute formule Π_2^0 *prouvable en arithmétique classique est prouvable en arithmétique intuitionniste.*

Conservativité

Ce résultat est impressionnant mais quand même limité :

Les formules atomiques de l'arithmétique sont toutes décidables, car équivalentes à $f(x_1, \ldots, x_n) = 0$ où f est une fonction primitive récursive. C'est pour cela qu'on n'a ≪pas vraiment besoin≫ du tiers exclu pour raisonner sur des formules Π^0_2.

Dans l'exemple classique d'énoncé où le tiers exclu est essentiel :

$$\forall m : TM,\ \texttt{termine}(m) \vee \neg\texttt{termine}(m)$$

le prédicat $\texttt{termine}(m)$ n'est pas une formule atomique et l'énoncé n'est pas Π^0_2. En effet :

$$\texttt{termine}(m) \overset{def}{=} \exists n, \texttt{exec}(m, n) = \texttt{terminé}$$

$$\neg\texttt{termine}(m) \Leftrightarrow \forall n, \texttt{exec}(m, n) \neq \texttt{terminé}$$

avec $\texttt{exec}(m, n)$ = exécuter n étapes de la machine m.

II

Continuations et opérateurs de contrôle

Notion de continuation

Étant donnés un programme fonctionnel p et une sous-expression a de p, la continuation de a est la suite des calculs restant à exécuter, une fois que a est évaluée, pour obtenir la valeur de p.

On peut la voir comme une fonction : (valeur de a) $\mapsto$ (valeur de p).

Exemple

Soit le programme $p = (1 + 2) \times (3 + 4)$, évalué de gauche à droite.

La continuation de $a = (1 + 2)$ est $\lambda v.\, v \times (3 + 4)$.

La continuation de $a' = (3 + 4)$ est $\lambda v.\, 3 \times v$.

(Et non pas $\lambda v.\, (1 + 2) \times v$, car $1 + 2$ a déjà été évalué en 3.)

Continuations et stratégies d'évaluation

Dans une sémantique pour un langage de programmation, expliciter les continuations permet de définir précisément la stratégie d'évaluation, c.à.d. l'ordre dans lequel les calculs sont effectués.

Par exemple, dans les langages fonctionnels on a deux stratégies d'évaluation classiques :

- Appel par valeur (CBV, *Call By Value*) :
  L'argument N d'un appel de fonction $(\lambda x.M)\ N$ est réduit en valeur (lambda-abstraction ou constante) avant d'être lié au paramètre x.
- Appel par nom (CBN, *Call By Name*) :
  L'argument N est lié à x sans évaluation. Il sera évalué quand (et à chaque fois que) la valeur de x est nécessaire.

La transformation CPS

(*Continuation-Passing Style*, style à passage de continuation)

On peut fixer la stratégie d'évaluation d'un lambda-terme M en le transformant en un terme $[\![M]\!]$ en style CPS. Ce terme

- attend un argument k représentant la continuation de M ;
- réduit M en une valeur v ;
- et finit en appliquant k à v.

Appel par nom

$$
\begin{aligned}
[\![cst]\!]_n &= \lambda k.\, k\ cst \\
[\![\lambda x.\, M]\!]_n &= \lambda k.\, k\ (\lambda x.\, [\![M]\!]_n) \\
[\![x]\!]_n &= \lambda k.\, x\ k \\
[\![M\ N]\!]_n &= \lambda k.\, [\![M]\!]_n\ (\lambda f.\, f\ [\![N]\!]_n\ k)
\end{aligned}
$$

La transformation CPS

(*Continuation-Passing Style*, style à passage de continuation)

On peut fixer la stratégie d'évaluation d'un lambda-terme M en le transformant en un terme $[\![M]\!]$ en style CPS. Ce terme

- attend un argument k représentant la continuation de M ;
- réduit M en une valeur v ;
- et finit en appliquant k à v.

Appel par nom

$$
\begin{aligned}
[\![cst]\!]_n &= \lambda k.\, k\ cst \\
[\![\lambda x.\, M]\!]_n &= \lambda k.\, k\ (\lambda x.\, [\![M]\!]_n) \\
[\![x]\!]_n &= \lambda k.\, x\ k \\
[\![M\ N]\!]_n &= \lambda k.\, [\![M]\!]_n\ (\lambda f.\, f\ [\![N]\!]_n\ k)
\end{aligned}
$$

Appel par valeur

$$
\begin{aligned}
[\![cst]\!]_v &= \lambda k.\, k\ cst \\
[\![\lambda x.\, M]\!]_v &= \lambda k.\, k\ (\lambda x.\, [\![M]\!]_v) \\
[\![x]\!]_v &= \lambda k.\, k\ x \\
[\![M\ N]\!]_v &= \lambda k.\, [\![M]\!]_v\ (\lambda f.\, \\
&\qquad [\![N]\!]_v\ (\lambda a.\, \\
&\qquad\quad f\ a\ k))
\end{aligned}
$$

La transformation CPS

La transformation CPS préserve la sémantique attendue tout en étant indifférente à l'ordre d'évaluation :

- $M \xrightarrow{*} cst$ en appel par valeur ssi
  $[\![M]\!]_v\ (\lambda x.\, x) \xrightarrow{*} cst$ dans n'importe quelle stratégie.
- $M \xrightarrow{*} cst$ en appel par nom ssi
  $[\![M]\!]_n\ (\lambda x.\, x) \xrightarrow{*} cst$ dans n'importe quelle stratégie.

Les opérateurs de contrôle

Les opérateurs de contrôle ajoutent aux langages fonctionnels des mécanismes pour réifier les continuations comme des valeurs de première classe, permettant ainsi aux programmes de manipuler leurs propres continuations.

Le plus ancien opérateur de contrôle est le `J` de Landin (*A generalization of Jumps and Labels*, 1965) :

$$f = \lambda x.\, \texttt{let}\ g = \texttt{J}(\lambda y.\, N)\ \texttt{in}\ M$$

L'opérateur `J` modifie la fonction locale $g = \lambda y.\, N$ de sorte que lorsque g est appelée dans M, elle retourne directement à l'appelant de f.

L'opérateur de contrôle le plus connu est le `call/cc` (*call with current continuation*) du langage Scheme.

L'opérateur `callcc`

L'expression `callcc`$(\lambda k.\, M)$ s'évalue comme suit :

- La continuation de cette expression est liée à la variable k.
- M est évaluée ; sa valeur est la valeur de `callcc`$(\lambda k.\, M)$.
- Si, pendant l'évaluation de M ou plus tard, on applique k à une valeur v, l'évaluation continue comme si `callcc`$(\lambda k.\, M)$ avait renvoyé la valeur v.

  En d'autres termes, la continuation de l'expression `callcc` est rétablie et relancée avec v comme résultat pour cette expression.

Exemple d'utilisation

Les bibliothèques de listes, ensembles, et autres collections fournissent souvent un itérateur impératif `iter`, p.ex.

```
(* list_iter: ('a -> unit) -> 'a list -> unit *)

let rec list_iter f l =
    match l with
    | [] -> ()
    | head :: tail -> f head; list_iter f tail
```

Exemple d'utilisation

À l'aide d'un opérateur de contrôle, un itérateur impératif peut être transformé en une fonction renvoyant le premier élément d'une collection satisfaisant un prédicat `pred`.

```
let find pred lst =
    callcc (λk.
        list_iter
            (λx. if pred x then k (Some x) else ())
            lst;
        None)
```

Si l'itération rencontre un élément `x` tel que `pred x = true`, l'application de `k` fait que `Some x` est immédiatement renvoyé comme résultat de `find pred lst`.

Si cet élément `x` n'existe pas, `list_iter` termine normalement et `None` est renvoyé.

Exemple d'utilisation

L'exemple précédent peut être réalisé avec des exceptions. Cependant, `callcc` ajoute la possibilité de redémarrer la recherche.

```
let find pred lst =
    callcc (λk.
        list_iter
            (λx. if pred x
                 then callcc (λk'. k (Some(x, k')))
                 else ())
            lst;
        None)
```

Lorsqu'on trouve `x` tel que `pred x = true`, `find` renvoie non seulement `x` mais aussi une continuation `k'` qui effectue un *backtracking* : la recherche dans `lst` redémarre à partir de l'élément qui suit `x`.

Exemple d'utilisation

En itérant `find` comme suit, on imprime tous les éléments de la liste qui satisfont le prédicat :

```
let printall pred lst =
    match find pred list with
    | None -> ()
    | Some(x, k) -> print_string x; k ()
```

L'expression `k ()` redémarre `find pred list` là où il s'était arrêté précédemment.

Sémantique de `callcc`

La transformation CPS s'étend facilement aux opérateurs comme `callcc`. Cela donne une sémantique à ces opérateurs, mais permet aussi de les implémenter dans un compilateur ou par traduction manuelle.

$$[\![\texttt{callcc}\ M]\!]_v = \lambda k.\ [\![M]\!]_v\ (\lambda f.\ f\ (\lambda v.\lambda k'.\ k\ v)\ k)$$

Dans `callcc` M, la valeur fonctionnelle f de M reçoit la continuation courante k à la fois comme argument («enveloppée» dans $\lambda v.\lambda k'.\ k\ v$) et comme continuation.

Lorsque la continuation capturée k est relancée sur une valeur v, la continuation k' courante à ce moment est ignorée.

Autres opérateurs de contrôle

Introduit par Felleisen vers 1986, l'opérateur $\mathcal{C}$ est une version simplifiée de `callcc` où la continuation courante, une fois capturée, est remplacée par la continuation initiale. On a cependant

$$\texttt{callcc}(\lambda k.\, M) = \mathcal{C}(\lambda k.\, k\, M)$$

Les opérateurs $\mathcal{F}$ / # (Felleisen et al, 1987), `shift` / `reset` (Danvy et Filinski, 1990), `cupto` (Gunter et al, 1995), etc, permettent de capturer des continuations délimitées

valeur d'une expression $\mapsto$ valeur d'une expression englobante

au lieu de capturer des continuations complètes

valeur d'une expression $\mapsto$ valeur du programme

III

Correspondances entre logique classique et opérateurs de contrôle

Transformation CPS et traduction négative

(Chetan Murthy, *Extracting Constructive Content from Classical Proofs*, PhD, Cornell, 1990.)

Dans ses travaux de thèse, Chetan Murthy met en évidence une correspondance à la manière de Curry-Howard entre

- la transformation CPS en appel par nom ;
- la traduction négative de Kolmogorov, relativisée par Friedman.

Plus précisément : la traduction négative décrit l'effet de la transformation CPS sur les types.

Typer la transformation CPS en appel par nom

Soit r le type du programme complet. On définit la transformation des types simples :

$$[\![t]\!] = (t^* \to r) \to r$$

$$\iota^* = \iota$$
$$(t \to s)^* = [\![t]\!] \to [\![s]\!]$$

Intuition : un terme de type t devient une fonction $\lambda k \ldots$
La continuation k attend une valeur de type t^* pour produire le résultat du programme, de type r. Donc $k : t^* \to r$ et le terme transformé est de type $(t^* \to r) \to r$.

Théorème (La transformation CPS préserve le typage)

Si $\ldots\ x_i : t_i\ \ldots \vdash M : t$, *alors* $\ldots\ x_i : [\![t_i]\!]\ \ldots \vdash [\![M]\!]_n : [\![t]\!]$.

Correspondance avec la traduction négative

Cette transformation des types simples correspond exactement à la traduction négative relativisée de Kolmogorov pour le fragment $\Rightarrow$:

$$\begin{aligned} [\![A]\!] &= \neg_R\neg_R A \text{ si } A \text{ est atomique} \\ [\![P \Rightarrow Q]\!] &= \neg_R\neg_R([\![P]\!] \Rightarrow [\![Q]\!]) \end{aligned}$$

ou, de manière équivalente,

$$[\![P]\!] = \neg_R\neg_R P^* = (P^* \Rightarrow R) \Rightarrow R$$

$$\begin{aligned} A^* &= A \text{ si } A \text{ atomique} \\ (P \Rightarrow Q)^* &= [\![P]\!] \Rightarrow [\![Q]\!] \end{aligned}$$

Correspondance avec la traduction négative

La correspondance s'étend aux autres connecteurs logiques ($\wedge, \vee, \bot, \forall, \exists$) car la transformation CPS s'étend à système F et aux types inductifs non dépendants, et donc en particulier aux types $\times$, $+$, Σ, et au type vide.

$$[\![\Lambda X.\, M]\!]_n = \lambda k.\, k\ (\Lambda X.\, [\![M]\!]_n)$$

$$[\![M[t]\ ]\!]_n = [\![M]\!]_n\ (\lambda x.\, x[t]\ k)$$

$$[\![C\ M_1\ \cdots\ M_p]\!]_n = \lambda k.\, k\ (C\ [\![M_1]\!]_n\ \cdots\ [\![M_p]\!]_n)$$

$$[\![\texttt{match}\ M\ \texttt{with}\ \cdots \mid C_i\ \vec{x}_i \Rightarrow N_i \mid \cdots]\!]_n = \lambda k.\, M\ (\lambda x.\, \texttt{match}\ x\ \texttt{with}\ \cdots \mid C_i\ \vec{x}_i \Rightarrow [\![N_i]\!]_n\ k \mid \cdots)$$

Typer la transformation CPS en appel par valeur

En appel par valeur, la transformation des types change légèrement :

$$[\![t]\!] = (t^* \to r) \to r$$

$$\iota^* = \iota$$
$$(t \to s)^* = t^* \to [\![s]\!]$$

Une fonction transformée prend un argument qui est déjà évalué, et donc de type t^*, et non plus un argument en attente d'évaluation ($\lambda k \ldots$) qui serait de type $[\![t]\!]$.

Les résultats de préservation du typage s'étendent à la transformation CPS en appel par valeur.

Correspondance avec la traduction négative

Cette transformation des types simples correspond à une variante de la traduction ≪économe≫ de Kuroda :

$$[\![P]\!] = \neg_R \neg_R P^* = (P^* \Rightarrow R) \Rightarrow R$$

$$\begin{aligned} A^* &= A \text{ si } A \text{ atomique} \\ (P \Rightarrow Q)^* &= P^* \Rightarrow [\![Q]\!] \qquad \text{(CPS en appel par valeur)} \\ (P \Rightarrow Q)^* &= P^* \Rightarrow Q^* \qquad \text{(Kuroda)} \end{aligned}$$

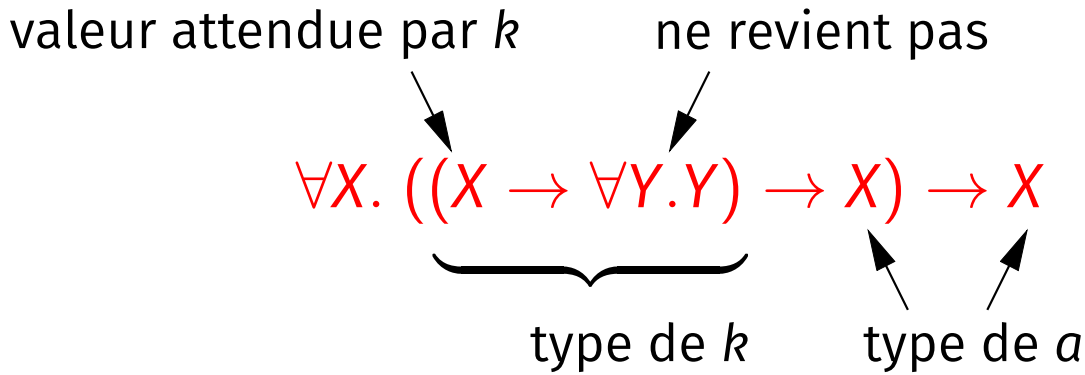
Vues comme des transformations de programmes, les traductions négatives de Kuroda et de Gödel-Gentzen ne sont pas intéressantes : elles préservent les types mais pas la sémantique dynamique.

Opérateurs de contrôle et lois classiques

(Timothy Griffin, *A Formulae-as-Types Notion of Control*, POPL 1990.)

En 1989, Griffin observe que les opérateurs de contrôle `callcc` et $\mathcal{C}$ peuvent recevoir des types qui correspondent à des lois de la logique classique.

Par exemple, le type de `callcc` est : (penser à `callcc`$(\lambda k.\, a)$)

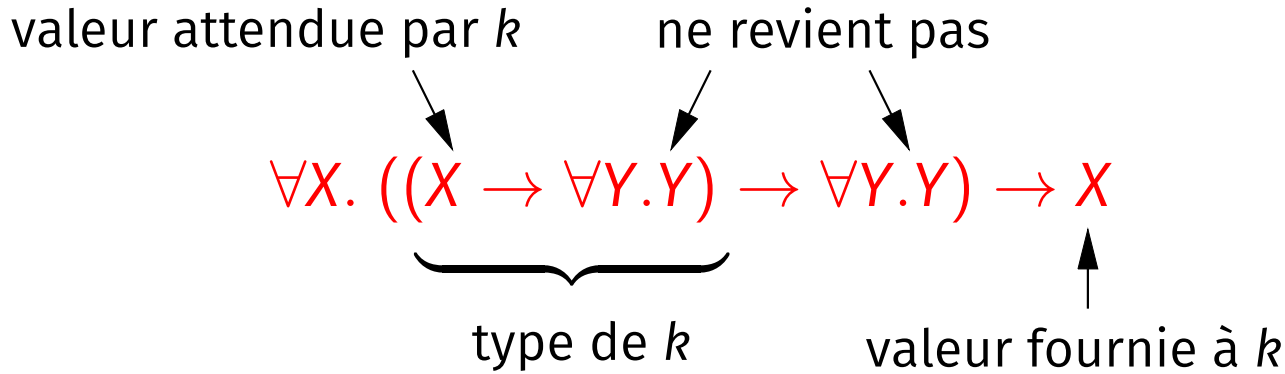


En lisant $\forall Y.Y$ comme $\bot$ et $X \to \forall Y.Y$ comme $\neg X$,
c'est la loi de Clavius, *mirabile consequentia* :

$$\forall P.\, (\neg P \Rightarrow P) \Rightarrow P$$

Opérateurs de contrôle et lois classiques

De même, le type de l'opérateur $\mathcal{C}$ est : (penser à $\mathcal{C}(\lambda k.\, a)$)



En lisant $T \to \forall Y.Y$ comme $\neg T$, c'est l'élimination de la double négation :

$$\forall P.\ \neg\neg P \Rightarrow P$$

Une «construction» du tiers exclu

Nous avons vu que, en logique minimale, la loi de Clavius implique le tiers exclu.

À partir de l'opérateur `callcc` qui a pour type la loi de Clavius, il est donc possible de construire un terme qui a pour type le tiers exclu :

$$\Lambda P.\ \texttt{callcc}\ (\lambda k : P \vee \neg P \Rightarrow \bot.\ \texttt{inj}_2\ (\lambda p : P.\ k\ (\texttt{inj}_1(p))))$$

$$: \forall P.\ P \vee \neg P$$

Mais quel est donc le comportement de ce terme ? ?

Un pacte Faustien

(D'après Phil Wadler)

LE DIABLE : J'ai un pacte à vous proposer. Ou bien (a) je vous offre un million d'euros, ou bien (b) vous me donnez un million d'euros et je vous accorde un souhait de votre choix.

FAUST : Pas d'autres conditions ? Je peux garder mon âme ?

LE DIABLE : Gardez-là. Mais c'est moi qui choisit entre (a) et (b).

FAUST : D'accord !

LE DIABLE : Je choisis (b).

Beaucoup plus tard, Faust revient avec la somme demandée et la donne au diable.

LE DIABLE : Merci ! Finalement, j'ai changé d'avis et je choisis (a).
Voici votre million d'euros !

Une ≪construction≫ du tiers exclu

$$\texttt{callcc}\ (\lambda k : P \vee \neg P \Rightarrow \bot.\ \ \texttt{inj}_2\ (\lambda p : P.\ k\ (\texttt{inj}_1(p))))$$

Le terme renvoie un $\texttt{inj}_2$, c.à.d. il choisit le cas $\neg P$.

Il renvoie aussi une ≪construction≫ de $\neg P$, c.à.d. une fonction $\lambda p : P \ldots$ de type $P \to \bot$.

Si le reste de la preuve n'utilise pas cette affirmation $\neg P$, tout va bien.

Si le reste de la preuve l'utilise, c'est en ≪l'éliminant≫, c.à.d. en passant une preuve $p : P$ à la fonction $\lambda p : P \ldots$ afin d'obtenir une preuve de $\bot$.

À ce moment, la continuation k est appelée avec $\texttt{inj}_1(p)$: on revient au moment du choix, on choisit le cas P, et on donne p comme construction.

Exercice : quel pacte Faustien joue l'élimination de la double négation ?

Une «construction» du tiers exclu

Dans le 1er cours, nous avons vu qu'il n'existe pas de construction (au sens de l'interprétation BHK) du tiers exclu, car une telle construction permettrait de décider le problème de l'arrêt :

$$\forall m : TM,\ \texttt{termine}(m) \vee \neg\texttt{termine}(m)$$

Où est le problème ?

Une incohérence logique ? Non !

Système F + `callcc` est normalisant.

Une autre notion de construction ? Oui !

Le terme $\texttt{callcc}(\lambda k \ldots)$ a le bon type mais peut se réduire en $\texttt{inj}_2(np)$ ou en $\texttt{inj}_1(p)$ selon le contexte. Cela sort du cadre de l'interprétation BHK, où une construction de $P \vee Q$ détermine une fois pour toute lequel des cas P ou Q est démontré.

IV

Séquents classiques et L-calculs

Dualité perdue

La vision ≪logique classique = logique intuitionniste + tiers exclu≫
nous fait perdre une symétrie formelle importante :
la dualité entre conjonction et disjonction via la négation.

$$\neg(P \wedge Q) = \neg P \vee \neg Q$$
$$\neg(P \vee Q) = \neg P \wedge \neg Q$$

Perdu également : la possibilité de définir l'implication à partir des autres connecteurs : $P \Rightarrow Q = \neg P \vee Q = \neg(P \wedge \neg Q)$.

Et enfin : l'élimination des coupures pour le $\vee$ est beaucoup plus difficile que pour $\Rightarrow$ et $\wedge$.

Dualité retrouvée

Le calcul des séquents classiques (Gentzen, 1934, 1935) est une formulation de la logique classique qui préserve la dualité entre conjonction et disjonction.

La notion centrale est celle de séquent classique

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_m \qquad \ll \text{si } A_1 \text{ et } \ldots \text{ et } A_n, \text{ alors } B_1 \text{ ou } \ldots \text{ ou } B_m \gg$$

Comparer avec les séquents intuitionnistes :

$$A_1, \ldots, A_n \vdash B \qquad \ll \text{si } A_1 \text{ et } \ldots \text{ et } A_n, \text{ alors } B \gg$$

Chaque connecteur logique a des règles «droites» (s'il apparaît dans les conclusions B_i) et des règles «gauches» (s'il apparaît dans les hypothèses A_i).

Calcul des séquents classiques

$$A \vdash A \ \text{(Id)} \qquad \frac{\Gamma \vdash \Delta, A \quad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \ \text{(coupure)}$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B} \ (\wedge D) \qquad \frac{A, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \ (\wedge G_1) \qquad \frac{B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \ (\wedge G_2)$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B} \ (\vee D_1) \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B} \ (\vee D_2) \qquad \frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta} \ (\vee G)$$

$$\frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \ (\neg D) \qquad \frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \ (\neg G)$$

$$\frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B} \ (\Rightarrow D) \qquad \frac{\Gamma \vdash \Delta, A \quad B, \Gamma' \vdash \Delta'}{A \Rightarrow B, \Gamma, \Gamma' \vdash \Delta, \Delta'} \ (\Rightarrow G)$$

(Plus : affaiblissement, contraction, échange.)

Le plaisir d'une logique classique

$$\dfrac{A \vdash A}{\vdash A,\ \neg A}\ (\neg D)$$

De là, on démontre le tiers exclu et l'élimination de la double négation :

$$\dfrac{\dfrac{\dfrac{\vdash A,\ \neg A}{\vdash A,\ A \vee \neg A}\ (\vee D_1)}{\vdash A \vee \neg A,\ A \vee \neg A}\ (\vee D_2)}{\vdash A \vee \neg A}\ (\text{contraction})$$

$$\dfrac{\dfrac{\vdash A,\ \neg A}{\neg\neg A \vdash A}\ (\neg G)}{\vdash \neg\neg A \Rightarrow A}\ (\Rightarrow D)$$

Curry-Howard pour les séquents classiques

Par Curry-Howard, le lambda-calcul typé correspond au calcul des séquents intuitionnistes.

Quel est le langage qui correspond au calcul des séquents classiques ?

Plusieurs propositions : $\lambda\mu$ de Parigot (1992, 1994) ; $\lambda\mu\tilde{\mu}$ de Curien et Herbelin (2000) ; le calcul dual de Wadler (2003) ; le L-calcul de Munch-Maccagnoni et al (2009) ; etc.

Deux idées centrales :

1. Un calcul $C = \langle M \mid K \rangle$ est l'interaction entre un terme M et un contexte K (aussi appelé co-terme, pile, ou continuation).
2. Pour refléter les multiples conclusions d'un séquent classique, les contextes peuvent avoir plusieurs ≪trous≫, identifiés par des co-variables α.

Un L-calcul pour les séquents classiques

(Inspiré de Wadler, *Call-by-value is dual to call-by-name*, ICFP 2003.)

Calculs : $C ::= \langle M \mid K \rangle$

Termes : $M, N ::= x \mid \ldots$

Contextes : $K, L ::= \alpha \mid \ldots$

Trois jugements de typage :

- $C : (x_1 : A_1, \ldots, x_n : A_n \vdash \alpha_1 : B_1, \ldots, \alpha_m : B_m)$
  ≪si les x_i reçoivent des valeurs de types A_i, l'exécution de C passe à une des continuations α_j une valeur de type B_j≫.
- $x_1 : A_1, \ldots, x_n : A_n \vdash \alpha_1 : B_1, \ldots, \alpha_m : B_m \mid M : B$
  ≪si les x_i reçoivent des valeurs de type A_i, l'évaluation de M produit une valeur de type B, ou passe à une des continuations α_j une valeur de type B_j≫.
- $K : A \mid x_1 : A_1, \ldots, x_n : A_n \vdash \alpha_1 : B_1, \ldots, \alpha_m : B_m$
  ≪si les x_i reçoivent des valeurs de type A_i et si K est appliquée à une valeur de type A, une des continuations α_j reçoit une valeur de type B_j≫.

Règles de typage

$$x : A \vdash \mid x : A \quad \text{(IdD)} \qquad \alpha : A \mid \vdash \alpha : A \quad \text{(IdG)}$$

$$\frac{\Gamma \vdash \Delta \mid M : A \qquad K : A \mid \Gamma' \vdash \Delta'}{\langle M \mid K \rangle : (\Gamma, \Gamma' \vdash \Delta, \Delta')} \quad \text{(coupure)}$$

$$\frac{\Gamma \vdash \Delta \mid M : A \qquad \Gamma \vdash \Delta \mid N : B}{\Gamma \vdash \Delta \mid (M, N) : A \wedge B} \quad (\wedge D) \qquad \frac{K : A \mid \Gamma \vdash \Delta}{\pi_1(K) : A \wedge B \mid \Gamma \vdash \Delta} \quad (\wedge G_1) \qquad \frac{K : B \mid \Gamma \vdash \Delta}{\pi_2(K) : A \wedge B \mid \Gamma \vdash \Delta} \quad (\wedge G_2)$$

$$\frac{\Gamma \vdash \Delta \mid M : A}{\Gamma \vdash \Delta \mid \mathtt{inj}_1(M) : A \vee B} \quad (\vee D_1) \qquad \frac{\Gamma \vdash \Delta \mid N : B}{\Gamma \vdash \Delta \mid \mathtt{inj}_2(N) : A \vee B} \quad (\vee D_2) \qquad \frac{K : A \mid \Gamma \vdash \Delta \qquad L : B \mid \Gamma \vdash \Delta}{(K \mid L) : A \vee B \mid \Gamma \vdash \Delta} \quad (\vee G)$$

$$\frac{K : A \mid \Gamma \vdash \Delta}{\Gamma \vdash \Delta \mid \mathtt{not}(K) : \neg A} \quad (\neg D) \qquad \frac{\Gamma \vdash \Delta \mid M : A}{\mathtt{not}(M) : \neg A \mid \Gamma \vdash \Delta} \quad (\neg G)$$

$$\frac{x : A, \Gamma \vdash \Delta \mid N : B}{\Gamma \vdash \Delta \mid \lambda x.N : A \Rightarrow B} \quad (\Rightarrow D) \qquad \frac{\Gamma \vdash \Delta \mid M : A \qquad K : B \mid \Gamma' \vdash \Delta'}{M \bullet K : A \Rightarrow B \mid \Gamma, \Gamma' \vdash \Delta, \Delta'} \quad (\Rightarrow G)$$

$$\frac{C : (\Gamma \vdash \Delta, \alpha : A)}{\Gamma \vdash \Delta \mid \mu\alpha.\, C : A} \quad \text{(actiD)} \qquad \frac{C : (x : A, \Gamma \vdash \Delta)}{\mu x.\, C : A \mid \Gamma \vdash \Delta} \quad \text{(actiG)}$$

(Plus : affaiblissement, contraction, échange.)

Règles de typage

$$A \vdash \mid \; A \;\; (\text{IdD}) \qquad\qquad A \mid \vdash \; A \;\; (\text{IdG})$$

$$\frac{\Gamma \vdash \Delta \mid \; A \qquad A \mid \Gamma' \vdash \Delta'}{(\Gamma, \Gamma' \vdash \Delta, \Delta')} \;\; (\text{coupure})$$

$$\frac{\Gamma \vdash \Delta \mid \; A \qquad \Gamma \vdash \Delta \mid \; B}{\Gamma \vdash \Delta \mid \; A \wedge B} \;\; (\wedge D) \qquad \frac{A \mid \Gamma \vdash \Delta}{A \wedge B \mid \Gamma \vdash \Delta} \;\; (\wedge G_1) \qquad \frac{B \mid \Gamma \vdash \Delta}{A \wedge B \mid \Gamma \vdash \Delta} \;\; (\wedge G_2)$$

$$\frac{\Gamma \vdash \Delta \mid \; A}{\Gamma \vdash \Delta \mid \; A \vee B} \;\; (\vee D_1) \qquad \frac{\Gamma \vdash \Delta \mid \; B}{\Gamma \vdash \Delta \mid \; A \vee B} \;\; (\vee D_2) \qquad \frac{A \mid \Gamma \vdash \Delta \qquad B \mid \Gamma \vdash \Delta}{A \vee B \mid \Gamma \vdash \Delta} \;\; (\vee G)$$

$$\frac{A \mid \Gamma \vdash \Delta}{\Gamma \vdash \Delta \mid \; \neg A} \;\; (\neg D) \qquad \frac{\Gamma \vdash \Delta \mid \; A}{\neg A \mid \Gamma \vdash \Delta} \;\; (\neg G)$$

$$\frac{A, \Gamma \vdash \Delta \mid \; B}{\Gamma \vdash \Delta \mid \; A \Rightarrow B} \;\; (\Rightarrow D) \qquad \frac{\Gamma \vdash \Delta \mid \; A \qquad B \mid \Gamma' \vdash \Delta'}{A \Rightarrow B \mid \Gamma, \Gamma' \vdash \Delta, \Delta'} \;\; (\Rightarrow G)$$

$$\frac{(\Gamma \vdash \Delta, \quad A)}{\Gamma \vdash \Delta \mid \; A} \;\; (\text{actiD}) \qquad \frac{(\quad A, \Gamma \vdash \Delta)}{A \mid \Gamma \vdash \Delta} \;\; (\text{actiG})$$

(Plus : affaiblissement, contraction, échange.)

Syntaxe des termes

Se déduit de la forme des règles de typage !

Calculs :	C	$::= \langle M \mid K \rangle$	
Termes :	M, N	$::= x \mid \lambda x.\, M$	variables, abstraction de fonction
		$\mid (M, N)$	constructeur de produits
		$\mid \mathtt{inj}_1(M) \mid \mathtt{inj}_2(M)$	constructeurs de sommes
		$\mid \mathtt{not}(K)$	complément d'un contexte
		$\mid \mu\alpha.C$	abstraction de co-variable
Contextes :	K, L	$::= \alpha$	co-variable
		$\mid M \bullet K$	application de fonction à M, puis K
		$\mid \pi_1(K) \mid \pi_2(K)$	projections de produits
		$\mid (K \mid L)$	analyse de cas sur somme
		$\mid \mathtt{not}(M)$	complément d'un terme
		$\mid \mu x.C$	abstraction de variable

Exercice : quel terme code l'élimination de la double négation ?

Règles de réductions

Quand un constructeur (dans le terme) rencontre son destructeur (dans le contexte) :

$$\langle (M, N) \mid \pi_1(K) \rangle \to \langle M \mid K \rangle \qquad \langle (M, N) \mid \pi_2(K) \rangle \to \langle N \mid K \rangle$$
$$\langle \mathtt{inj}_1(M) \mid (K \mid L) \rangle \to \langle M \mid K \rangle \qquad \langle \mathtt{inj}_2(M) \mid (K \mid L) \rangle \to \langle M \mid L \rangle$$
$$\langle \mathtt{not}(K) \mid \mathtt{not}(M) \rangle \to \langle M \mid K \rangle \qquad \langle \lambda x.\, M \mid N \bullet K \rangle \to \langle N \mid \mu x.\, \langle M \mid K \rangle \rangle$$

Plus : les β-réductions pour les lieurs μ :

$$\langle M \mid \mu x.\, C \rangle \to C\{x \leftarrow M\} \qquad \langle \mu \alpha.\, C \mid K \rangle \to C\{\alpha \leftarrow K\}$$

Problème : les réductions ne sont pas confluentes !
(Par exemple $\langle \mu \alpha.\, C \mid \mu x.\, C' \rangle$ se réduit de deux manières différentes.)

Stratégies de réduction

Il faut donc imposer une stratégie de réduction, p.ex. on réduit $\langle M \mid K \rangle$

- seulement si M est une valeur (en particulier : pas $\mu\alpha.\, C$)
  $\Rightarrow$ généralisation de l'appel par valeur
- ou seulement si K est une ≪co-valeur≫ (e.p. : pas $\mu x.\, C$)
  $\Rightarrow$ généralisation de l'appel par nom.

(Autre approche possible : par polarisation.)

Théorème (Wadler 2003)

Avec la stratégie ≪par valeur≫, $\neg(A \wedge \neg B)$ se comporte comme $A \Rightarrow B$.
Avec la stratégie ≪par nom≫, $\neg A \vee B$ se comporte comme $A \Rightarrow B$.

V

En guise de conclusion

Une démonstration classique a-t'elle un contenu calculatoire ?

Non, dixit Girard, Lafont, Taylor (1989). L'élimination des coupures pour le calcul des séquents classique n'est pas confluente, et donc :

> *[A BHK interpretation] is not possible with classical logic : there is no sensible way of considering proofs as algorithms. In fact, classical logic has no denotational semantics, except the trivial one which identifies all the proofs of the same type. This is related to the nondeterministic behaviour of cut elimination.*

Oui, d'après Murthy, Griffin, et les L-calculistes. Mais il faut au moins :

- fixer une stratégie de réduction (appel par nom ou par valeur) pour contourner le manque de confluence ;
- accepter que les ≪constructions≫ (termes de preuve) sortent du lambda-calcul pur et entrent dans les calculs avec effets.

Une démonstration classique a-t'elle un contenu calculatoire ?

Non, dixit Girard, Lafont, Taylor (1989). L'élimination des coupures pour le calcul des séquents classique n'est pas confluente, et donc :

> *[A BHK interpretation] is not possible with classical logic : there is no sensible way of considering proofs as algorithms. In fact, classical logic has no denotational semantics, except the trivial one which identifies all the proofs of the same type. This is related to the nondeterministic behaviour of cut elimination.*

Oui, d'après Murthy, Griffin, et les L-calculistes. Mais il faut au moins :

- fixer une stratégie de réduction (appel par nom ou par valeur) pour contourner le manque de confluence ;
- accepter que les ≪constructions≫ (termes de preuve) sortent du lambda-calcul pur et entrent dans les calculs avec effets.

VI

Bibliographie

Bibliographie

- Sur la programmation avec continuations et `callcc` :
  Shriram Krishnamurthi. *Programming Languages : Application and Interpretation*. Chapitre 14.
  `http://cs.brown.edu/courses/cs173/2012/book/index.html`

- Sur les liens entre logique classique et continuations :
  Morten Heine Sørensen, Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism*. 1998. Chapitre 6.
  `https://disi.unitn.it/~bernardi/RSISE11/Papers/curry-howard.pdf`.

- Sur le calcul des séquents classiques :
  Jean-Yves Girard. *Le point aveugle. Cours de logique. Tome 1 : vers la perfection*. Hermann, 2006. Chapitre 3.