Logique et Calculabilité
INF551

Dr. Stéphane Lengrand,

Stephane.Lengrand@Polytechnique.edu

Cours 9

Le lambda-calcul et les preuves constructives

3 notions équivalentes

– Fonctions calculables (“μ-récursives”) - Gödel, 1933

– λ-calcul - Church, 1936

– Machines de Turing, 1937

Notion de calcul plus bas niveau, réalisable physiquement (Bletchley Park)

Modèle de coût dans le temps et dans l’espace

I. Le lambda-calcul

Se rapprocher de l'expression des fonctions en maths

Si le programme e implémente la fonction qui à p associe 2^p,

comment implémenter la fonction qui à p associe 2^{2^p} ?

$\circ_1^1(e, e)$ ou alors $\circ_1^1(e, \circ_1^1(e, \pi_1^1))$

fun $x \to App(e, App(e, x))$

Un symbole binaire App, un symbole unaire *fun* (ou $\mapsto$ ou λ) qui lie une variable dans son argument

On écrit $(t\ u)$ pour $App(t, u)$

fun $x \to (e\ (e\ x))$

La currification

Pas besoin de fonctions de plusieurs arguments

$$\mathit{fun}\, x \to \mathit{fun}\, y \to (x + (x \times y))$$

La réduction

On applique *fun* $x \to t$ à u : transformer en $(u/x)t$

$$((\textit{fun}\, x \to t)\ u) \longrightarrow (u/x)t$$

Radical : terme réductible par $\longrightarrow$, $((\textit{fun}\, x \to t)\ u)$

$\rhd$ inductivement définie par

– si $t \longrightarrow t'$, alors $t \rhd t'$

– si $t \rhd t'$, alors $(t\ u) \rhd (t'\ u)$

– si $u \rhd u'$, alors $(t\ u) \rhd (t\ u')$

– si $t \rhd t'$, alors $(\textit{fun}\, x \to t) \rhd (\textit{fun}\, x \to t')$

$\rhd^*$ fermeture réflexive-transitive de $\rhd$

Malheureusement pas un ensemble de règles de réécriture au sens de la dernière fois, car lieur

La terminaison

t irréductible si ne peut pas être réduit par $\triangleright$ (= aucun sous-terme radical)

t termine s'il existe t' irréductible tel que $t \triangleright^* t'$

$\omega = ((\mathit{fun}\, x \to (x\ x))\ (\mathit{fun}\, x \to (x\ x)))$?

$((\mathit{fun}\, x \to y)\ \omega)$?

La confluence

Theorem: $\triangleright$ confluente

Preuve : $\triangleright^{\|}$ fortement confluente (cf dernière PC)

La représentation des fonctions : comme pour la réécriture

À chaque entier p un terme irréductible $\underline{p}$

Definition:

F (λ-terme) représente f :

– si $f(p_1, ..., p_n) = q$, $(F\ \underline{p_1}\ ...\ \underline{p_n}) \rhd^* \underline{q}$

– si f pas définie en $p_1, ..., p_n$, $(F\ \underline{p_1}\ ...\ \underline{p_n})$ ne termine pas

La réduction en appel par nom : comme pour la réécriture

Pas une suite de petits pas car mille manières de réduire $(F\ \underline{p_1}\ \ldots\ \underline{p_n})$

Une stratégie particulière : la réduction en appel par nom

$\succ$ inductivement définie par

– si $t \longrightarrow t'$, alors $t \succ t'$

– si $(t\ u)$ n'est pas un radical (c'est-à-dire si t n'est pas de la forme *fun*) et si $t \succ t'$, alors $(t\ u) \succ (t'\ u)$

– si $(t\ u)$ n'est pas un radical et aucun sous-terme de t n'est un radical et $u \succ u'$, alors $(t\ u) \succ (t\ u')$

– si $t \succ t'$, alors $(\mathit{fun}\, x \to t) \succ (\mathit{fun}\, x \to t')$

La représentation des fonctions en appel par nom : comme pour la réécriture

Definition:

F représente f en appel par nom :

– si $f(p_1, ..., p_n) = q$, $(F\ \underline{p_1}\ ...\ \underline{p_n}) \succ^* \underline{q}$

– si f n'est pas définie en $p_1, ..., p_n$, $(F\ \underline{p_1}\ ...\ \underline{p_n})$ ne termine pas en appel par nom

Le théorème de standardisation Nouveau

Theorem:

si $t \rhd^* t'$ et t' est irréductible, alors $t \succ^* t'$

terminaison = terminaison en appel par nom

représentation = représentation en appel par nom

II. La représentation des fonctions calculables dans le lambda-calcul

La représentation des entiers : les entiers de Church

Représentation guidée par volonté de déf. des fonct. par récurrence

3, c'est itérer trois fois un processus

$$\underline{3} = \mathit{fun}\, x \to \mathit{fun}\, f \to (f\ (f\ (f\ x)))$$

$$\underline{p} = \mathit{fun}\, x \to \mathit{fun}\, f \to (\underbrace{f\ (f\ \ldots\ (f}_{p \text{ fois}}\ x)\ldots))$$

h est la fonction qui double son argument, qu'est $(\underline{n}\ \underline{1}\ h)$?

Quatre notions d'entiers

3 c'est $S(S(S(0)))$ (Peano)

3 c'est le point commun des ensembles de trois éléments (Cantor)

3 c'est $\{0, 1, 2\}$ (Von Neumann)

3 c'est $\mathit{fun}\ x \to \mathit{fun}\ f \to (f\ (f\ (f\ x)))$ (Church)

π_i^n, Z^n, S et $\circ_m^n$

$$\textit{fun}\ x_1 \rightarrow\ \ldots.\ \textit{fun}\ x_n \rightarrow x_i$$

$$\textit{fun}\ x_1 \rightarrow\ \ldots.\ \textit{fun}\ x_n \rightarrow \underline{0}$$

$$\textit{fun}\ n \rightarrow \textit{fun}\ x \rightarrow \textit{fun}\ f \rightarrow f\ (n\ x\ f)$$

$$F = \textit{fun}\ x_1 \rightarrow\ \ldots\ \textit{fun}\ x_n \rightarrow (H\ (G_1\ x_1\ \ldots\ x_n))\ \ldots\ (G_m\ x_1\ \ldots\ x_n))$$

Un peu trop naïf

Si g n'est pas définie en 4 et $h = Z^1$

$f = h \circ g$ pas définie en 4

mais

$$(F\ \underline{4}) = (\mathit{fun}\ y \to ((\mathit{fun}\ x \to \underline{0})\ (G\ y))\ \underline{4})$$

$$\succ ((\mathit{fun}\ x \to \underline{0})\ (G\ \underline{4})) \succ \underline{0}$$

$\&$

Comme dans le cas de la réécriture

$t\&u$: t pourvu que u termine (sur un entier)

$\&$ est cette fois-ci implémenté par

$$t\&u = (u\ t\ (\mathit{fun}\ x \to x))$$

On vérifie que $t\&\underline{p} \succ^* t$ et $t\&u$ ne termine pas si u ne termine pas.

La minimisation et le point fixe

Tester qu'une valeur est nulle + itérer une fonction à l'infini

$$\mathit{Ifz} = \mathit{fun}\, x \to \mathit{fun}\, y \to \mathit{fun}\, z \to (x\ y\ \mathit{fun}\, z' \to z)$$

Un entier de Church infini ? terme t

$$Y_t = ((\mathit{fun}\, x \to (t\ (x\ x)))\ (\mathit{fun}\, x \to (t\ (x\ x))))$$

$$Y_t \succ (t\ Y_t) \succ (t\ (t\ Y_t)) \succ \ldots$$

La représentation des fonctions calculables

Projections :

$\mathit{fun}\, x_1 \to \ \ldots.\ \mathit{fun}\, x_n \to ((((x_i \& x_1) \& \ldots \& x_{i-1}) \& x_{i+1}) \& \ldots \& x_n)$

Zéros : $\mathit{fun}\, x_1 \to \ \ldots.\ \mathit{fun}\, x_n \to ((\underline{0} \& x_1) \& \ldots \& x_n)$

Successeur : $\mathit{fun}\, x \to \mathit{fun}\, f \to \mathit{fun}\, n \to (f\ (n\ x\ f))$

Plus : $\mathit{fun}\, p \to \mathit{fun}\, q \to (((\mathit{fun}\, x \to \mathit{fun}\, f \to (p\ (q\ x\ f)\ f)) \& p) \& q)$

Fois :

$\mathit{fun}\, p \to \mathit{fun}\, q \to (((\mathit{fun}\, x \to \mathit{fun}\, f \to (p\ x\ (\mathit{fun}\, y \to (q\ y\ f)))) \& p) \& q)$

$\chi_<$: $\mathit{fun}\, p \to \mathit{fun}\, q \to (((p\ (K\ \underline{1})\ T\ (q\ (K\ \underline{0})\ T)) \& p) \& q)$

où $K = \mathit{fun}\, x \to \mathit{fun}\, y \to x$ et $T = \mathit{fun}\, g \to \mathit{fun}\, h \to (h\ g)$

La représentation des fonctions calculables

Composition :

$$\mathit{fun}\, x_1 \to \ \ldots\ \mathit{fun}\, x_n \to (H\ (G_1\ x_1\ \ldots\ x_n))\ \ldots\ (G_m\ x_1\ \ldots\ x_n))$$

Minimisation :

$$\mathit{fun}\, x_1 \to \ \ldots\ \mathit{fun}\, x_n \to (Y_{G'}\ x_1\ \ldots\ x_n\ \underline{0})$$

où $G' = \mathit{fun}\, f \to \mathit{fun}\, x_1 \to \ \ldots\ \mathit{fun}\, x_n \to \mathit{fun}\, x_{n+1} \to$
$(\mathit{Ifz}\ (G\ x_1\ \ldots\ x_n\ x_{n+1})\ x_{n+1}\ (f\ x_1\ \ldots\ x_n\ (S\ x_{n+1})))$

Le théorème

Si $f(p_1, ..., p_n) = q$, alors $(F\ \underline{p_1}\ ...\ \underline{p_n}) \succ^* \underline{q}$

Si f pas définie en $p_1, ..., p_n$, alors $(F\ \underline{p_1}\ ...\ \underline{p_n})$ ne termine pas

La postérité du lambda-calcul

Milner : on ajoute des entiers primitifs (4 opérations + test), un point fixe primitif et un let : PCF

et des arbres : un vrai langage de programmation : ML

Utilisé comme base pour aborder de nouveaux domaines : *e.g.* lambda-calcul polymorphe, lambda-calcul parallèle, lambda-calcul quantique, ...

III. La constructivité

Un autre type de relations entre preuves et algorithmes

Jusqu'ici : algorithmes pour (semi-)décider de la prouvabilité

maintenant : aspect algorithmique des preuves que l'on a trouvées

L'absence de témoin

P prédicat : $P(0), \neg P(2)$

Peut-on montrer qu'il existe un entier x tel que $P(x) \wedge \neg P(S(x))$?

$$P(0), \neg P(2) \vdash \exists x\, (P(x) \wedge \neg P(S(x)))$$

Existe-t-il un entier n tel qu'on puisse montrer $P(n) \wedge \neg P(S(n))$?

$$P(0), \neg P(2) \vdash P(n) \wedge \neg P(S(n))$$

L'ensemble des prop. prouvables n'a pas la propriété du témoin

Un exemple en mathématiques

$u_0 = \sqrt{2}, u_{n+1} = u_n^{\sqrt{2}}$

$P(n)$: u_n irrationnel

$P(0), P(2)$?

Il existe x tel que $P(x)$ et $\neg P(x+1)$

donc il existe un irrationnel r tel que $r^{\sqrt{2}}$ rationnel

Ou bien $\sqrt{2}$ ou bien $\sqrt{2}^{\sqrt{2}}$ selon que $\sqrt{2}^{\sqrt{2}}$ est rationnel ou non

Un autre exemple

P prédicat

Peut-on montrer qu'il existe un entier x tel que $P(x) \vee \neg P(S(x))$?

Existe-t-il un entier n tel qu'on puisse montrer $P(n) \vee \neg P(S(n))$?

Le responsable: :

Le tiers exclu : ou bien $P(1)$ ou bien $\neg P(1)$,

dans le premier cas 1 dans le second 0

Les preuves constructives

En déduction naturelle, on supprime la règle du tiers exclu

En calcul des séquents, on contraint les séquents à avoir une conclusion au plus

On garde (mais redémontre) : l'équivalence entre la DN et le CS

et le théorème d'élimination des coupures

La propriété de la dernière règle

Une preuve dans le calcul des séquents

– constructive

– dans la théorie vide

– sans coupures

se termine par une règle droite

La propriété de la dernière règle

Une preuve de $\exists x\ A$ dans le calcul des séquents

– constructive

– dans la théorie vide

– sans coupures

se termine par une règle $\exists$-droite

Si $\exists x\ A$ a une preuve dans le calcul des séquents

– constructive

– dans la théorie vide

alors il existe un terme t tel que $(t/x)A$ soit prouvable

Le théorème

L'ensemble des propositions qui ont une preuve dans la théorie vide et constructive a la propriété du témoin

Pour calculer le témoin : on élimine les coupures

S'étend à d'autres théories : l'arithmétique mais pas $\exists x\ P(x)$

Programmer avec des preuves

Dans l'arithmétique, la proposition

$$\forall x\, \exists y\, (x = 2 \times y \vee x = 2 \times y + 1)$$

a-t-elle une preuve constructive ?

Et

$$\exists y\, (25 = 2 \times y \vee 25 = 2 \times y + 1)$$

?

Quel est le témoin ?

Comment le calcule-t-on ?

Les preuves sont des programmes

Une preuve (constructive) de

$$\forall x\, \exists y\, (x = 2 \times y \vee x = 2 \times y + 1)$$

est un programme qui calcule la moitié

Son mécanisme d'exécution est l'élimination des coupures

Il est correct vis à vis de la spécification

$$x = 2 \times y \vee x = 2 \times y + 1$$

La suite

En PC : La correspondance de Curry-Howard

La prochaine fois : Exam !

Questions?