Calculabilité
Cours 1 : Fonctions récursives et λ-calcul

Objectifs de cette partie (3 cours)

Présenter les éléments essentiels de la calculabilité en évitant les détails trop techniques

- 3 modèles de calcul
  - fonctions récursives, λ-calcul, machines de Turing
- La non-calculabilité
  - démontrer l'indécidabilité, présentation de quelques problèmes indécidables, théorème de Rice

Une référence principale :

Pierre Wolper[1], *Introduction à la calculabilité*, Dunod

[1] cf `http://www.montefiore.ulg.ac.be/~pw/cours/calc.html`

Motivations

- Comprendre les limites de l'informatique.
- Distinguer problèmes solubles et insolubles par des algorithmes.
- Obtenir des résultats indépendants de la technologie employée pour construire les ordinateurs.

Problèmes et Langages

Quels problèmes sont solubles par un programme exécuté sur un ordinateur ?

Il faut préciser :

- la notion de problème,
- la notion de programme exécuté sur un ordinateur.

Nous aborderons aujourd'hui le 2ième point.

La notion de procédure effective

Trois modèles de calcul

- Fonctions récursives (Gödel, 1930)
- λ-calcul (Church, 1930)
- Machine de Turing (Turing et Post, 1936)

Conjecture (non-démontrable) : ces formalisations sont des modèles appropriés.

Théorème[2] : ces modèles sont équivalents.

[2] cf `http://www.lsv.ens-cachan.fr/~comon/ftp.articles/poly-calculabilite.ps.gz`

Les fonctions récursives

Introduction

Définie les fonctions calculables sur les naturels.

- fonctions dans l'ensemble

$$\bigcup_{k \in \mathbb{N}} \mathcal{F}_k \qquad \text{avec } \mathcal{F}_k = \mathbb{N}^k \to \mathbb{N}$$

Fonctions calculables ?

- fonctions de base
- composition de fonctions
- mécanisme de récursion

Fonctions primitives récursives

Définition (Fonctions primitives récursives de base)

- fonction zéro : $\mathbf{0}() \in \mathcal{F}_0$

$$\mathbf{0}() = 0$$

- fonction de projection : $\pi_i^k \in \mathcal{F}_k, k \geqslant 1, 1 \leqslant i \leqslant k$

$$\pi_i^k(n_1, \ldots, n_k) = n_i$$

- fonction successeur : $\sigma \in \mathcal{F}_1$

$$\sigma(n) = n + 1$$

Fonctions primitives récursives

Définition (Composition de fonctions)

Soit $g \in \mathcal{F}_l, h_1, \ldots, h_l \in \mathcal{F}_k$, la composition de g et des fonctions $h_1, \ldots, h_l$ est définie par $f \in \mathcal{F}_k$:

$$f(\overline{n}) = g(h_1(\overline{n}), \ldots, h_l(\overline{n}))$$

avec $\overline{n} = (n_1, \ldots, n_k)$.

Fonctions primitives récursives

Définition (Récursion primitive)

Soit $g \in \mathcal{F}_k$ et $h \in \mathcal{F}_{k+2}$. La fonction $f \in \mathcal{F}_{k+1}$ définit par

$$\begin{aligned} f(\overline{n}, 0) &= g(\overline{n}) \\ f(\overline{n}, m+1) &= h(\overline{n}, m, f(\overline{n}, m)) \end{aligned}$$

est la fonction définie à partir de g et h par récursion primitive.

Remarque : f est calculable si g et h sont calculables.

Fonctions primitives récursives

Définition (Fonctions primitives récursives)

Les fonctions primitives récursives sont toutes les fonctions que l'on peut construire à partir des fonctions de base par composition et récursion primitive.

Exemples

- Fonction constante $\mathbf{j}() \in \mathcal{F}_0$
- Fonction addition
- Fonction multiplication
- Fonction puissance
- Généralisation $\rightarrow$ fonction d'Ackermann (pas primitive récursive)
- Fonction factorielle
- Fonction prédecesseur
- Fonction différence
- Fonction signe

Prédicats primitifs récursifs

Un prédicat est une fonction dont les valeurs sont prises dans un ensemble binaire de la forme $\{\text{vrai}, \text{faux}\}$.

Définition

Un prédicat P à k arguments est un sous-ensemble de $\mathbb{N}^k$ (les éléments de N^k pour lesquels P est vrai).

La fonction caractéristique d'un prédicat $P \subset \mathbb{N}^k$ est la fonction $f \in \mathbb{N}^k \to \{0, 1\}$ telle que

$$f(\overline{n}) = \begin{cases} 0 & \text{si } \overline{n} \notin P \\ 1 & \text{si } \overline{n} \in P \end{cases}$$

Définition

Un prédicat est primitif récursif si sa fonction caractéristique est primitive récursive.

Exemples

- zéro ? $\in \mathbb{N}^1 \to \{0,1\}$
- $<\in \mathbb{N}^2 \to \{0,1\}$
- opération booléennes
- quantification bornée $\in \mathbb{N}^{k+1} \to \{0,1\}$

$$\forall i \leqslant m\ p(\overline{n}, i) \qquad \exists i \leqslant m\ p(\overline{n}, i)$$

- définition par cas

$$f(\overline{n}) = \begin{cases} g_1(\overline{n}) & \text{si } p_1(\overline{n}) \\ \dots & \\ g_l(\overline{n}) & \text{si } p_l(\overline{n}) \end{cases}$$

Minimisation bornée

La fonction $\mu i \leqslant \cdot\, q(\cdot, i) \in \mathcal{F}_{k+1}$

$$\mu i \leqslant m\ q(\overline{n}, i) = \begin{cases} \text{le plus petit } i \leqslant m \text{ tel que } q(\overline{n}, i) = 1 \\ 0 \text{ s'il n'existe pas} \end{cases}$$

est primitive récursive.

Fonctions récursives générales

Théorème

Il existe des fonctions qui ne sont pas primitives récursives.

Théorème

Il existe des fonctions calculables qui ne sont pas primitives récursives.

Minimisation non bornée

$$\mu i\, q(\overline{n}, i) = \begin{cases} \text{le plus petit } i \text{ tel que } q(\overline{n}, i) = 1 \\ 0 \text{ si un tel } i \text{ n'existe pas} \end{cases}$$

Un prédicat $q(\overline{n}, i)$ est dit *sûr* si

$$\forall \overline{n}\ \exists i\ q(\overline{n}, i) = 1$$

Fonctions récursives générales

Définition (Fonctions μ-récursives)

Les fonctions μ-récursives sont toutes les fonctions que l'on peut construire à partir des fonctions primitives récursives de base par :

- composition, récursion primitive et
- minimisation non bornée de prédicats sûrs.

Remarque : Les fonctions μ-récursives sont calculables par une procédure effective.

$\lambda-$calcul[3]

[3] cf les notes de cours de Larry Paulson, *Foundations of Functional Programming*,
`http://www.cl.cam.ac.uk/~lp15/papers/Notes/Founds-FP.pdf`

Syntaxe

On fixe un ensemble de *variables* $x, y, z, \ldots \in \mathbb{V}$

Les *termes* du λ−calcul sont de la forme

$$\begin{array}{rl} x & \\ (\lambda x.\, M) & \text{avec } M \text{ un terme} \\ (M\, N) & \text{avec } M \text{ et } N \text{ des termes} \end{array}$$

Exemple : $\lambda x.\, x \quad \lambda f.\, \lambda x.\, (f\, x)$ (on omet les parenthèses superflus)

$\lambda x.\, M$ représente la fonction f telle que $f(x) = M$.

- x est *la variable liée*
- M le *corps*

Variables libres, variables liés

Définition (Variable liées (*bound variables*))

$$\begin{aligned} \mathrm{BV}(x) &= \emptyset \\ \mathrm{BV}(\lambda x.M) &= \mathrm{BV}(M) \cup \{x\} \\ \mathrm{BV}(M\,N) &= \mathrm{BV}(M) \cup \mathrm{BV}(N) \end{aligned}$$

Définition (Variable libres (*free variables*))

$$\begin{aligned} \mathrm{FV}(x) &= \{x\} \\ \mathrm{FV}(\lambda x.M) &= \mathrm{FV}(M) - \{x\} \\ \mathrm{FV}(M\,N) &= \mathrm{FV}(M) \cup \mathrm{FV}(N) \end{aligned}$$

Exemples : $\lambda f.\,(f\ x) \quad (x\,\lambda x.(f\ x)) \quad ((\lambda x.\,x)\,\lambda f.(f\ x))$

Substitution

$M[L/y]$ est la substitution dans M de toutes les occurrences libres de y par L.

$$
\begin{aligned}
x[L/y] &= \begin{cases} L & \text{si } x = y \\ x & \text{sinon} \end{cases} \\
(\lambda x.\, M)[L/y] &= \begin{cases} (\lambda x.\, M) & \text{si } x = y \\ (\lambda x.\, M[L/y]) & \text{sinon} \end{cases} \\
(M\,N)[L/y] &= (M[L/y]\; N[L/y])
\end{aligned}
$$

Capture de variables

Exemples : $(\lambda z.\, x)[y/x]$ et $(\lambda y.\, x)[y/x]$

Définition

La substitution $M[N/x]$ est dite *sûre* si $\mathrm{BV}(M) \cap \mathrm{FV}(N) = \emptyset$.

On se limitera toujours aux substitutions sûres. Ce sera toujours possible après d'éventuels renommages (α−conversions).

Conversions

$\alpha-$conversion :

$$(\lambda x.\, M) \rightarrow_\alpha (\lambda y.\, M[y/x]) \qquad y \text{ non libre dans } M$$

Convention : Nous ne distinguerons pas les termes qui sont identiques après (éventuellement plusieurs) $\alpha-$conversion.

$\beta-$conversion :

$$((\lambda x.\, M)\, N) \rightarrow_\beta M[N/x]) \qquad \text{si la substitution est sûre}$$

Réduction

Ces deux conversions sont étendus aux sous-termes

$$\begin{aligned} (\lambda x.\, M) &\to (\lambda x.\, M') && \text{si } M \to M' \\ (M\,N) &\to (M'\,N) && \text{si } M \to M' \\ (M\,N) &\to (M\,N') && \text{si } N \to N' \end{aligned}$$

β−réduction :

$\to_\beta^* = \bigcup_{n\in\mathbb{N}} \to_\beta^n$ représente le calcul sur un λ−terme.

Exemples

- $((\lambda x.\, \lambda y.\, (x\,y))\, b)\, c$
- $(\lambda x.\, (a\, (\lambda y.\, (x\,y)))\, b)\, c$

Forme normale

Définition (Rédex)

Un *rédex* d'un terme est une sous-expression de la forme $((\lambda x.\, X)\, Y)$.

Définition (Forme normale)

Un terme est *en forme normale* si il ne contient aucun rédex.

Un terme en forme normale ne peut plus être β−réduit (le calcul est terminé).

Forme normale

Définition (Terme normalisable)

Un terme E est *normalisable* si il existe un terme E' en forme normale telle que $E \rightarrow_\beta^* E'$.

Exercices :

- $(\lambda x.\,(x\,x))\,(\lambda x.\,(x\,x))$ est-il normalisable ?
- Donner un exemple de terme normalisable pour lequel il existe un séquence infinie de β−réduction.

Stratégie de réduction

Il existe plusieurs façons de réduire un terme.

Théorème (Church-Rosser)

Si $E \to_\beta^* E_1$ *et* $E \to_\beta^* E_2$, *alors il existe* E' *tel que* $E_1 \to_\beta^* E'$ *et* $E_2 \to_\beta^* E'$.

Ce théorème assure l'unicité de la forme normale (exercice).

Stratégie de réduction

Il existe un stratégie gagnante.

Définition (Réduction normale)

La *réduction normale* d'un terme est la stratégie qui applique la β−conversion aux rédex le plus à gauche.

Théorème (Curry)

Si E est normalisable vers une forme normale N alors la réduction normale de E mène aussi à N.

Autrement dit, la réduction normale d'un terme normalisable termine toujours.

Les entiers de Church

Codage des entiers en λ−calcul :

$$
\begin{aligned}
\overline{0} &= \lambda f.\,\lambda x.\,x \\
\overline{1} &= \lambda f.\,\lambda x.\,(f\ x) \\
\overline{2} &= \lambda f.\,\lambda x.\,(f\ (f\ x)) \\
&\dots \\
\overline{n} &= \lambda f.\,\lambda x.\,\underbrace{(f\ (f\ \dots\ x))}_{n \text{ fois}}
\end{aligned}
$$

cf TD.

Fonctions λ−représentables

Convention : on écrit parfois $f\,x\,y$ au lieu de $((f\,x)\,y)$.

Définition (Fonctions λ−représentables)

Soit f une fonction de $\mathbb{N}^k \to \mathbb{N}$. On dit que f est λ−*représentable* par un terme F si

$$\forall n_1, \ldots, n_k \in \mathbb{N}, \quad F\,\overline{n_1}\,\ldots\,\overline{n_k} \to_\beta^* \overline{f(n_1, \ldots, n_k)}$$

Thèse de Church : Les fonctions calculables sont les fonctions λ−représentables.

Exercice

Expliquer pourquoi les fonctions λ−représentables sont calculables par une procédure effective.

Remarques divers

La notion de fonction λ−représentable est parfois définie à l'aide de $\equiv_\beta$ (la plus petite relation d'équivalence contenant $\rightarrow_\beta$).

Nous avons utilisé, sans le préciser, des *définitions inductives*. Nous y reviendrons plus rigoureusement dans les prochaines séances.

Plan

1. Introduction
2. Les fonctions récursives
   - Fonction primitives récursives
   - Prédicats primitifs récursifs
   - Fonctions récursives générales
3. λ−calcul
   - Substitution
   - Calcul
   - Représentation des fonctions entières