

Bilan du cours

Bruno Grenet

Université Grenoble Alpes – IM²AG

L3 Informatique

UE Modèles de calcul – Machines de Turing



<https://membres-ljk.imag.fr/Bruno.Grenet/MCAL-MT.html>

Formalisation des algorithmes

Il existe une définition *formelle* de la notion d'*algorithme*

Machine de Turing

- ▶ Un exemple de formalisation possible
- ▶ Nombreuses variantes *équivalentes*
- ▶ Modèle théorique utile pour des preuves, pas pour l'informatique *en pratique*

Nombreux modèles équivalents

- ▶ λ -calcul : modèle proche de la programmation fonctionnelle
- ▶ Machine RAM : modèle proche de la programmation impérative
- ▶ Langage de programmation $\simeq$ formalisation
- ▶ Modèles *exotiques* (automates cellulaires, calcul par ADN, ...): notion de calcul très fondamentale

Thèse de Church-Turing

On a trouvé la *bonne* formalisation des algorithmes

Justification

- ▶ Aucun contre-exemple n'a été trouvé
- ▶ Modèles extrêmement divers tous équivalents
- ▶ Semble correspondre à l'expérience *pratique* (ordinateurs)

Nature de la *thèse*

- ▶ Par définition non démontrable
- ▶ Hypothèse sur monde physique $\simeq$ loi de la physique

Caractéristiques du calcul

Aspects nécessaires ou non pour calculer

Modèles variés et points communs

- ▶ Mémoire non bornée (et exploitable)
- ▶ Localité : à chaque instant, modifications bornées
- ▶ Temps fini

Exemple des machines RAM

- ▶ Suffisant et équivalent : JUMP, GOTO ou WHILE
- ▶ Non suffisant : FOR, REPEAT ou LOOP
- ▶ Remarque : IF ... THEN ... ELSE ... → simulable

ou appels récursifs

Incalculabilité

Les algorithmes ne peuvent pas tout calculer car il y a plus de fonctions que d'algorithmes

Algorithmes : dénombrable

- ▶ Représentation d'un algorithme $\rightsquigarrow$ mot (fini) sur $\{0, 1\}$
 - ▶ quelque soit le modèle !
- ▶ Nombre de mots finis : dénombrable = bijection avec $\mathbb{N}$

Fonctions : indénombrable

- ▶ Spécification d'une fonction $f : \{0, 1\}^* \rightarrow \{0, 1\} \rightsquigarrow$ mot **infini** sur $\{0, 1\}$
- ▶ Preuve par diagonalisation : pas de bijection avec $\mathbb{N}$

Des problèmes indécidables

Les propriétés de la fonction calculée par un algorithme sont indécidables

Théorème de l'arrêt

- ▶ Il n'existe pas d'algorithme H qui, étant donné (le code de) $\mathcal{A}$ et x , détermine si $\mathcal{A}(x) \downarrow$
- ▶ Preuve par contradiction : si H , existe, on construit $D(\langle \mathcal{A} \rangle)$:
 - ▶ si $H(\langle \mathcal{A}, \mathcal{A} \rangle) = 1$: boucle infinie ; sinon : renvoyer 1Alors $D(\langle D \rangle) \downarrow \iff H(\langle D, D \rangle) = 0 \iff D(\langle D \rangle) \uparrow$

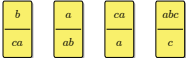
Théorème de Rice

- ▶ Propriété satisfaite par une fonction f : sous-ensemble (non trivial) P des fonctions
- ▶ Théorème de Rice : étant donné (le code de) $\mathcal{A}$, impossible de déterminer si $\mathcal{A}(\cdot) \in P$
 - ▶ $P_x = \{f : f(x) \downarrow\} \rightarrow$ théorème de l'arrêt
 - ▶ $P_0 = \{f : f(z) = 0 \text{ pour au moins un } z\}$
 - ▶ $P_{\text{CARRÉ}} = \{f : f(z) = z^2 \text{ pour tout } z\}$

Autres résultats d'incalculabilité

L'incalculabilité peut se trouver dans tous les domaines

Pavages et domino

- ▶ Peut-on paver le plan avec ces tuiles :  ?
- ▶ Ces dominos peuvent-ils s'aligner de manière finie :  ?

Autres problèmes

- ▶ Étant donné $P \in \mathbb{Z}[x_1, \dots, x_k]$, existe-t-il $n_1, \dots, n_k \in \mathbb{N}$ tels que $P(n_1, \dots, n_k) = 0$?
- ▶ Étant donné une fonction f définie par une *expression mathématique*, est-ce que $f(x) = 0$ pour tout x ?
- ▶ Qui va gagner la partie de *Magic: The Gathering* ?
- ▶ Étant donné une grammaire hors-contexte, est-elle ambiguë ?