Calculabilité CM5

Antonio E. Porreca
aeporreca.org/calculabilite

Universalité et complétude

Théorème 1 : Le langage $L_u = \{\langle M \rangle \# w \mid M \text{ accepte } w\}$ est $\mathbf{RE}$

- Rappel : on sait déjà que ce langage n'est pas $\mathbf{R}$ (cours 3, corollaire 3)

- On ne décrit pas formellement une machine de Turing, mais plutôt on décrit un semi-algorithme informel pour L_u

- Un semi-algorithme est une sorte d'algorithme, qui répond oui pour les mots qui appartiennent au langage…

- …mais il a le droit de ne pas s'arrêter sur le mots qui n'appartiennent pas au langage (il peut aussi les rejeter)

Théorème 1 : Le langage $L_u = \{\langle M \rangle \# w \mid M \text{ accepte } w\}$ est RE

- Voici le semi-algorithme :

    1. D'abord, vérifier si l'entrée est bien de la forme $\langle M \rangle \# w$, sinon rejeter
    2. Simuler la machine M sur l'entrée w jusqu'à son arrêt (si elle s'arrête), et renvoyer le même résultat

- On peut faire (1), voir les UE Langages formels (L2) et Compilation (L3)

- On discutera comment faire (2)

Théorème 2

- Il existe des langages récursivement énumerables qui ne sont pas récursif
- Notamment le langage L_u
- Donc la famille des langages $\mathbf{RE}$ n'est pas close par complémentation
- Notamment, L_u est $\mathbf{RE}$ mais son complémentaire $L_{\bar{u}}$ ne l'est pas

Machines de Turing universelles

- Dans la démonstration du Théorème 1 on utilise une machine qui simule une machine M dont elle reçoit en entrée la description $\langle M \rangle$ sur un mot d'entrée w

- C'est une machine M_u qui satisfait l'équation

$$M_u(\langle M \rangle \# w) = M(w)$$

- On appelle M_u une machine de Turing universelle

- Les ordinateurs de tous les jours suivent le même principe !

Comment construire une machine de Turing universelle

- On a déjà vu comment obtenir le code $\langle M \rangle$ d'une machine M (cours 2, section 3.4)

- On défini, par exemple, un mot binaire qui contient le nombre d'états de M, la taille de son alphabet et surtout sa table de transition

- Pour obtenir l'entrée complete $\langle M \rangle \# w$ on ajoute # et l'entrée w de M

Simulation de $\langle M \rangle \# w$ par M_u

- M_u retient l'état courant de M sur son ruban et la position courante de la tête de lecture de M, en marquant la case correspondante

- A chaque étape, M_u vérifie s'il existe une transition à partir de l'état et symbole courants de M, en lisant sa description $\langle M \rangle$

- Si une telle transition existe, alors on met à jour l'état, le symbole sur le ruban et la position de la tête de M

- Sinon la simulation de M termine, et M_u termine aussi avec le même résultat (acceptation ou rejet)

Turing-complétude

- Un modèle de calcul est une définition formelle (ou formalisable) d'une façon de calculer, avec une syntaxe et une sémantique

- Par exemple : les machines de Turing, les programmes en Python, les ordinateurs électroniques

- Un modèle de calcul qui calcule toutes les fonctions calculables par les machines de Turing est dit Turing-complet

- Un modèle de calcul qui simule une machine de Turing universelle est Turing-complet

- Tous les langages de programmation usuels (Python, C, Java, JavaScript, Haskell, …) sont Turing-complets !

Thèse de Church-Turing

Modèles de calcul équivalents

- Deux modèles de calcul sont équivalents s'ils calculent les mêmes fonctions

- Les machines de Turing avec deux rubans sont équivalentes a celles avec un ruban (TD1)

- Les machines de Turing sont équivalents à tous les langages de programmation (Python, C, Java, …)

Thèse de Church-Turing

- Toute fonction calculable par un algorithme (une description finie et bien définie d'un processus de calcul) est calculable par une machine de Turing

- On appelle ça une thèse et pas un théorème parce que la notion d'algorithme n'est pas définie mathématiquement

- Une version physique plus forte de la thèse de Church-Turing dit que toute fonction physiquement calculable est calculable par une machine de Turing

The End