Calculabilité & Complexité

Calculabilité

Définitions et Théorèmes

Cohérence : propriété d'un système formel dans lequel unénoncé et sa négation ne peuvent être démontrés vrais tous les deux. (notion universelle de la vérité)

Complétude : propriété d'un système où tout énoncé vrai est démontrable.

Décidabilité : existence, dans un système formel, d'un procédé systématique (algorithme) qui permet de déterminer la véracité / fausseté d'un énoncé démontrable. (automatisation des preuves)

Semi-décidabilité : Il existe un algorithme qui termine en temps fini et répond "oui" si l'entrée est vraie.

Th. De complétude : La logique des prédicats est complète.

1er Th. d'incomplétude : Tout système formel "un peu riche" (contenant la thérorie des nombres) est soit incohérent soit incomplet.

2nd Th. d'incomplétude : La cohérence d'un système formel (un peu riche) n'est pas démontrable au sein de ce système.

Turing : Tout système formel "un peu riche" est indécidable.

Th. de Rice : Toute propriété sémantique non triviale d'un programme est indécidable.

Thèse de Church-Turing : Il y a équivalence entre :

- les fonctions intuitivement calculables
- les machines de Turing (ordinateur)
- les fonctions récursives (langage de programmation)
- le lambda-calcul (langage fonctionnel)
- les langages récursivement énumérables (ensemble de termes)

Machines de Turing

Il existe une **machine de Turing universelle** qui, ayant en entrée le codage $< M >$ d'une machine M et un mot m, calcule l'application de M à m. En 2007, il suffit de 2 états, 3 symboles et 6 transitions.

Turing-complet : Un système formel est Turing-complet s'il est aussi puissant que les machines de Turing, c-à-d qu'on peut y décrire toute fonction calculable par une machine de Turing, ou de manière équivalente, avec lequel on peut simuler une machine de Turing universelle.

Turing-équivalence : Un système formel est Turing-équivalent s'il réalise exactement les mêmes fonctions que les machines de Turing.

Construction d'un prédicat indécidable

Soit $T(i, a)$ le prédicat sur $\mathbb{N} \times X^*$ qui retourne vrai si l'exécution de la machine de codage i appliquée à a retourne un résultat (ie $\mathcal{M}(a)$ s'arrête, avec $\langle\mathcal{M}\rangle = i$), et faux sinon.

Supposons T décidable. Il existe $\mathcal{M}_T$ qui décide T.

Soit la machine de Turing $\mathcal{M}$ prenant un argument a et définie par : si $T(a, a)$ est vrai alors $\mathcal{M}$ boucle, sinon $\mathcal{M}$ s'arrête.
(construction : $\mathcal{M}$ duplique son argument a – lire le premier symbole, aller à la fin, l'écrire, revenir au début, etc –, puis exécute $\mathcal{M}_T$ qui laisse 0 ou 1 sur le ruban, et boucle ou termine selon cette valeur)

$\mathcal{M}$ possède un code de Gödel j. Pour ce j, si $T(j, j)$ est vrai alors $\mathcal{M}(j)$ doit boucler, donc $T(j, j)$ doit être faux. Si $T(j, j)$ est faux alors $\mathcal{M}(j)$ doit s'arrêter donc $T(j, j)$ doit être vrai. Contradiction.

La machine $\mathcal{M}$ est impossible, donc $\mathcal{M}_T$ n'existe pas,
donc T est indécidable. $\square$

Indécidabilité du rejet : savoir si une machine n'accepte pas un mot est un problème ni décidable, ni semi-décidable.

Indécidabilité du test à zéro : savoir si une machine n'accepte aucun mot est indécidable.

Indécidabilité de l'équivalence de 2 machines de Turing.

Tout problème ayant un nombre **d'instances finies** est décidable.

Ackermann est une fonction calculable non récursive primitive.

Complexité

- P = classe des problèmes solubles en temps polynomial de la taille de l'entrée.
- NP = classe des problèmes vérifiables en temps polynomial.
- NP-complet = problèmes vérifiables en temps polynomial, pas soluble en temps polynomial (sans doute, voir P=NP?).
- EXPTIME = problèmes solubles en temps exponentiel.
- LSPACE = problèmes nécessitant moins que $log(n)$ espace.
- PSPACE = problème nécessitant un espace polynomial.
- EXPSPACE = problèmes nécessitant un espace exponentiel.