914

NOM : PINAULT Prénom : Laureline Jury :

Algèbre ← Entourez l'épreuve → Analyse

Sujet choisi : 914. Décidabilité et indécidabilité. Exemples.

Autre sujet :

INTRODUCTION

Déf 1 : Un problème est la donnée de la représentation des éléments d'un ensemble au plus dénombrable et d'une question sur ces éléments.

Exple 2 : INSTANCE : un entier en base 10
QUESTION : cet entier est-il premier ?

Exple 3 : INSTANCE : un entier comme produit de ses diviseurs premiers
QUESTION : cet entier est premier ?

But : Est-ce que étant donné un problème et une instance on peut toujours déterminer mécaniquement si l'instance répond à la question.

(I) Calcul théorique

1. Un premier modèle de calcul : les machines de Turing

Déf 4 : Une machine de Turing est un 7-uplet $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ où :
- Q est un ensemble d'état
- Σ, Γ des alphabets, $\square \in \Sigma$, $\square \in \Gamma$ et $\Sigma \subsetneq \Gamma$
- δ fonction de transition : $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- $q_0, q_{acc}, q_{rej} \in Q$, $q_{acc} \neq q_{rej}$

Déf 5 : Une configuration est l'état du ruban et l'état où on est, q est l'état et la tête de lecture pointe sur la première lettre de v. (plan)

Rqe 6 (Robustesse) On pourrait définir de manière équivalente une machine de Turing :
- dont la tête de lecture peut rester sur place
- avec plusieurs rubans
- avec Σ, $|\Sigma| = 2$
- avec une fonction de transition non déterministe

2. Problèmes décidables et indécidables

Déf 7 : Un langage est décidable (ou Turing-décidable) si il existe $\mathcal{M}$ une machine de Turing qui s'arrête sur toute entrée et accepte si et seulement si l'entrée est dans le langage.

Rqe 8 : Un problème $\mathcal{P} = (\text{INSTANCE}, \text{QUESTION})$ peut avoir comme un langage $\mathcal{L}(\mathcal{P}) = \{\langle x \rangle, x \in \text{INSTANCE}$ QUESTION$(x)\}$ ou $\langle x \rangle$ est un codage de x. On dit alors que $\mathcal{P}$ est décidable si $\mathcal{L}(\mathcal{P})$ l'est.

Rqe 9 : Il existe des problèmes indécidables (argument diagonal).

3. Autres modèles de calculs

Déf 10 (fonctions récursives) Les fonctions récursives sont l'ensemble des fonctions : [redacted]
- contenant les constantes, les projections, et le successeur
- clos par composition, récurrence et minimalisation

Déf 11 (RAM) Description d'un langage assembleur, c'est

une suite d'instructions de la forme : CHARGER, LIRE, ECRIRE, $\leftrightarrow$, $+$, $\times$, $-$, $\div$ ...

Prop 12. Les fonctions récursives, [redacted] le modèle RAM et les machines de Turing sont équivalents, c'est à dire qu'ils définissent la même classe de problème décidable :

$L \subseteq \Sigma^*$ est décidable $\Leftrightarrow \chi(L)$ est récursive
$\Leftrightarrow \chi(L)$ est calculable par une suite d'instructions RAM.

où $\chi(\cdot)$ désigne la fonction caractéristique.

Rq 13. La notion de décidabilité permet de définir l'équivalence de modèles de calcul mais réciproquement on peut montrer l'indécidabilité d'un problème avec le modèle de calcul qu'on veut.

Thèse de Church-Turing. Tout modèle de calcul "raisonnable" (ie décrivant ce qu'on peut mécaniquement faire) est équivalent au modèle des machines de Turing.

II Exemples de problèmes décidables et indécidables

0. Comment montrer la décidabilité ou l'indécidabilité d'un problème ?

a) La décidabilité

Idée 14. Donner un algorithme résolvant le problème (eg une machine de Turing décidant).

Idée 15. Le transformer en un problème qu'on sait décidable.

b) L'indécidabilité

Idée 16. ("à la main") Supposer que le problème est décidable et aboutir à une contradiction.

Exemple 17. Le problème de l'acceptation ($\{\langle \Pi, w\rangle, \Pi$ accepte $w\}$) et le problème de l'arrêt ($\{\langle \Pi, w\rangle, \Pi$ s'arrête sur $w\}$) pour les machines de Turing sont indécidables.

Idée 18. On effectue une réduction d'un problème qu'on sait indécidable vers ce problème (voir annexe 2).

Exemple 19. Le problème de l'acceptation uniforme ($\{\langle \Pi\rangle, \Pi$ accepte toute entrée$\}$ et le problème de l'arrêt uniforme ($\{\langle \Pi\rangle, \Pi$ s'arrête sur toute entrée$\}$) sont indécidables pour les machines de Turing.

1. Problèmes relatifs aux langages

Prop 20. Les problèmes de l'acceptation, du vide et de l'équivalence pour les automates finis sont décidables. (Idée 14)

Prop 21. Les problèmes de l'acceptation et du vide pour les grammaires context-free sont décidable (Idée 14).
En revanche le problème de l'équivalence est indécidable pour les grammaires context-free (Idée 18)

Prop 22. Le problème du vide et celui de savoir si le langage reconnu est régulier ou context-free sont indécidables pour les machines de Turing (Idée 18)

Thm 23 (Rice). Plus généralement, toute propriété non triviale des langages accepté par une machine de Turing est indécidable. (Idée 18)

2. Problèmes mathématiques

Exple 24. (PCP, problème de correspondance de Post). Soit une suite de couples de mots $(u_1, v_1), \dots (u_k, v_k)$ sur un alphabet X. Le problème de savoir s'il existe un mot $w \in X^*$ tel que $w = u_{i_1} \dots u_{i_r} = v_{i_1} \dots v_{i_r}$ est indécidable. (Idée 18)

Appli 25. (Problème du zéro dans les matrices 3x3) Etant donnée une suite de matrices 3×3 à coefficients dans $\mathbb{Z}$ $\Pi_1, \dots, \Pi_k$, le problème de savoir si il existe $\Pi = \Pi_{i_1} \dots \Pi_{i_r}$ tel que $\Pi_{3,2} = 0$ est indécidable. (Idée 18)

Exple 26. (Convergence d'une suite d'entier). Soit (u_n) une suite de $\mathbb{Z}$ définie ~~explicitement~~ effectivement alors la convergence de (u_n) est indécidable (Idée 18)

Exple 27. Le problème de l'associativité d'une loi de composition (récursivement totale) est indécidable. (Idée 19)

Exple 28 (admis) (10ème problème de Hilbert). Le problème de savoir si un polynome à plusieurs variables admet une racine entière est indécidable.

3. Décidabilité de théories logiques

Def 29. Une théorie logique est dite décidable si l'ensemble de ses fmles closes forme un langage décidable.

Exple 30. (Presburger) La théorie au 1er ordre des entiers munis de l'addition est décidable (Idée 18).

Exple 31. (Tarski) La théorie au 1er ordre des entiers munis de l'addition et de la multiplication est indécidable. (Idée 18')

Thm 32.

4. Système de réécriture de terme

Def 33. Un système de réécriture de terme est un ensemble de règles $\{l_i \to r_i\}_{i \in I}$ où I est fini et les r_i sont des termes. On écrit $s \to^* t$ si on peut passer de s à t en appliquant un nombre fini de fois ces règles à des sous termes.

Il est dit confluent si $(s \to^* t_1 \wedge s \to^* t_2) \Rightarrow (\exists\, t,\ t_1 \to^* t \wedge t_2 \to^* t)$.

Il est dit terminant si il n'existe pas $t_1 \to t_2 \to \dots \to t_n \to \dots$

Prop 34. La confluence et la terminaison d'un système de réécriture de terme sont indécidables. (Idée 19').

III Par delà le clivage décidabilité-indécidabilité

1. Indécidable mais... tout n'est pas perdu : la semi-décidabilité

Def 35. Un langage est reconnaissable s'il existe une machine de Turing qui accepte son entrée si et seulement si elle est dans le langage (voir annexe 3)

Prop 36. Il existe des langages non reconnaissables (argument diagonal).

Prop 37. Il existe des langages reconnaissables mais non décidables.

Exple 38. Le problème de l'arrêt est reconnaissable.

Exple 39. La démonstration de théorèmes est indécidable. Cela n'empêche ni le développement de systèmes de démonstration ni les mathématiciens de se pencher sur ce sujet.

2. Décidable mais... tout n'est pas gagné : la complexité

Ce n'est pas parce qu'un problème admet un algorithme qui le décide que cet algorithme est réalisable en pratique : il peut nécessiter trop d'espace ou trop de temps de calcul.

Exple 40. Décider si un graphe est connexe ou fortement connexe se fait avec un algorithme polynomial en temps : c'est considéré comme un problème facile.

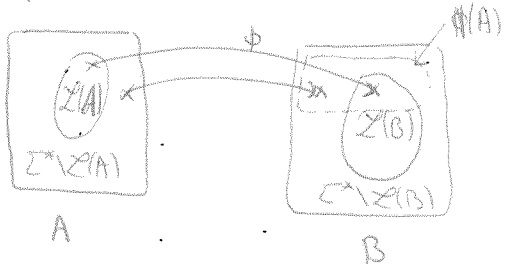
Exple 41. On ne connaît pas d'algorithme polynomial en temps qui décide la satisfiabilité des formules booléennes : c'est considéré comme un problème difficile.

Rq 42. Tout comme pour la décidabilité, on peut définir des classes de problèmes ayant la même type de complexité et invariante par changement de modèle "raisonnable" : ce sont les classes de complexité (annexe 4)

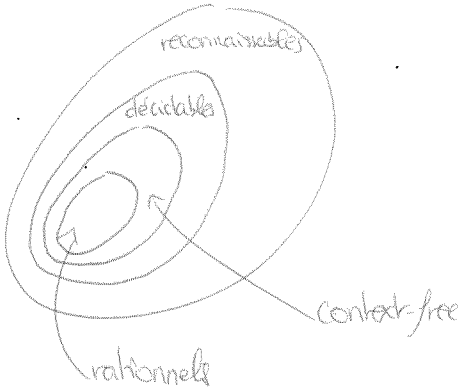
annexe 1 (Configuration)

est représenté par $1 0 q_1 1 1 0 1$.

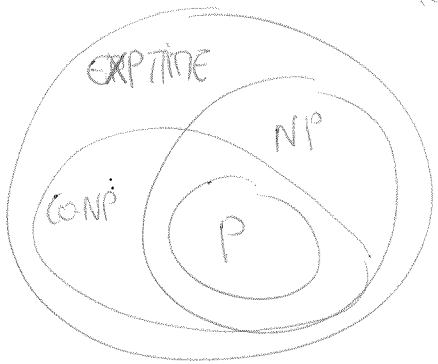
annexe 2 (réduction du problème A au problème B)



annexe 3 (la hiérarchie des langages)



annexe 4 (classes de complexité (en temps))



Références

- Autebert
- Sipser
- Carton
- Wolper