Calculabilité et Complexité: CM3

Florian Bridoux

Polytech Nice Sophia

2023-2024

Table des matières

Table of Contents

Réduction

Theorem

Le langage $L_d = \{\langle M \rangle \mid M \text{ n'accepte pas le mot } \langle M \rangle\}$ n'est pas décidable.

- Par l'absurde, supposons qu'une machine M_d reconnaisse L_d. Considérons alors l'entrée $\langle M_d \rangle$ donnée à la machine M_d. Deux cas sont possibles.
  - Si M_d n'accepte pas $\langle M_d \rangle$ alors, par définition du langage L_d, le mot $\langle M_d \rangle$ est dans L_d. Or M_d reconnait L_d, donc M_d accepte $\langle M_d \rangle$, une contradiction.
  - Si M_d accepte $\langle M_d \rangle$ alors, par définition du langage L_d, le mot $\langle M_d \rangle$ n'est pas dans L_d. Or M_d reconnait L_d, donc M_d n'accepte pas $\langle M_d \rangle$, une contradiction.
- Dans les deux cas nous arrivons à une contradiction.
- Donc le langage L_d n'est pas décidable.

Réduction

Définition: Réduction Turing

Une **réduction (many-one) Turing** du langage A au langage B est une fonction calculable $f : \Sigma_A^* \to \Sigma_B^*$ telle que $w \in A \iff f(w) \in B$. On note $A \leq_m^T B$.

- Intuitivement, un problème A se réduit à un problème B si connaissant un algorithme pour décider/calculer B, on peut obtenir un algorithme pour décider/calculer A.
- A est alors plus facile (ou aussi facile) que B!

Remarque

many-one qualifie la fonction f (de plusieurs vers un, il faut comprendre par là que ni l'injectivité ni la surjectivité ne sont imposées). On écrit *one-to-one* quand on impose la bijectivité.

Réduction

Définition

On appelle *instance* de L un mot $w \in \Sigma_L^*$ dont on se demande s'il appartient au langage L.

Théorème

Si le langage A se réduit au langage B ($A \leq_m^T B$), alors:

- si A n'est pas décidable alors B non plus;
- si A n'est pas semi-décidable alors B non plus.
- si A n'est pas co-semi-décidable alors B non plus.

Preuve au TD3.

Remarque

Pour montrer qu'un langage B est indécidable, on choisit un langage A bien connu pour être indécidable, et l'on réduit A à B. En effet, 1) B décidable $\implies$ A décidable et 2) A indécidable implique que B est indécidable.

Réduction

Théorème

Le langage $L_u = \{\langle M \rangle \# w \mid M \text{ accepte le mot } w\}$ n'est pas décidable: son complément $L_{\bar{u}}$ n'est pas semi-décidable.

Preuve:

- Nous allons réduire L_d à $L_{\bar{u}}$ en décrivant une procédure algorithmique pour transformer les instances de L_d en des instances de $L_{\bar{u}}$. Soit w' une instance de L_d.
  1. la machine vérifie $w' \in L_{enc}$, si w' n'est pas un encodage valide alors on retourne l'instance $\langle M_{palindrome} \rangle \# abba$ (on a bien $w' \notin L_d$ et $\langle M_{palindrome} \rangle \# abba \notin L_{\bar{u}}$) ;
  2. si $w' = \langle M \rangle$ est un encodage valide, alors on retourne l'instance $w' \# w' = \langle M \rangle \# \langle M \rangle$ (on a bien $w' \in L_d$ si et seulement si $\langle M \rangle \# \langle M \rangle \in L_{\bar{u}}$).

  Cette réduction montre que si $L_{\bar{u}}$ est décidable alors L_d l'est également, or nous savons que L_d n'est pas décidable, donc $L_{\bar{u}}$ non plus, et par conséquent L_u n'est pas décidable.

Réduction

Théorème

Le langage $L_{halt\epsilon} = \{\langle M \rangle \mid M$ s'arrête quand on la lance sur l'entrée vide $\}$ n'est pas décidable.

- Prouvons que $L_u \leq_m^T L_{halt\epsilon}$. Etant donnée $\langle M \rangle \# w$ une instance de L_u, construisons l'instance $\langle M' \rangle$ suivante de $L_{halt\epsilon}$:
  - Si $\langle M \rangle$ n'est pas un encodage valide alors on retourne $\langle M \rangle$;
  - sinon on construit $\langle M' \rangle$ avec M' la machine qui commence par écrire w sur le ruban (cela est possible en utilisant $|w|$ états), puis entre dans l'état initial de M (M' va alors se comporter comme M), et nous rajoutons également à M' des transitions, depuis tous les états non finaux où M s'arrête (transition indéfinie), vers un état qui boucle à l'infini.
- Dans tous les cas, nous avons bien $\langle M \rangle \# w \in L_u \iff \langle M' \rangle \in L_{halt\epsilon}$, donc si $L_{halt\epsilon}$ est décidable alors L_u est décidable. Or, L_u n'est pas décidable, donc $L_{halt\epsilon}$ n'est pas décidable.

Table of Contents

Théorème de Rice

Definition

Soit P une famille de langages. On appelle P une **propriété non triviale** si il existe deux machines de Turing M_1 et M_2 telles que $L(M_1) \in P$ et $L(M_2) \notin P$.

P et $\bar{P}$ peut être composés d'un nombre fini ou infini de langages. Mais clairement, si P est fini alors $\bar{P}$ est infini et inversement.

Théorème de Rice

Pour toute propriété non triviale P, il n'existe aucun algorithme pour décider si une MT M de code $\langle M \rangle$ vérifie $L(M) \in P$.
Autrement dit, $L_P = \{\langle M \rangle \mid L(M) \in P\}$ n'est pas décidable.

Remarque

Mais L_P est potentiellement semi-décidable ou co-semi-dédicable.

Preuve du Théorème de Rice

On prouve que $L_u \leq_m^T L_P$. Supposons $\emptyset \notin P$ (sinon on considère $\bar{P}$ au lieu de P). Puisque P est non triviale, il existe une MT M_P telle que $L(M_P) \in P$.
Etant donnée $\langle M \rangle \# w$ une instance de L_u, nous allons constuire l'instance $\langle M' \rangle$ de L_p avec M' la machine qui :

1. copie son entrée v sur un ruban séparé pour l'utiliser plus tard;
2. écrit w sur le ruban et place la tête sur la première lettre de w;
3. entre dans l'état initial de M. À partir de là M' simule M, en ignorant v, jusqu'à ce que (éventuellement) M entre dans son état final ;
4. si M entre dans son état final, alors le mot v est recopié sur un ruban blanc (que des symboles B) et la machine M_P est simulée sur v. On entre dans un état final si M_P accepte v.

Preuve du Théorème de Rice

- si M accepte w, alors M' acceptera exactement les mots v que M_P accepte, donc dans ce cas $L(M') = L(M_P) \in P$ et $\langle M' \rangle \in L_P$;
- si M n'accepte pas w, alors M' n'accepte aucun mot v, donc dans ce cas $L(M') = \emptyset \notin P$ et $\langle M' \rangle \notin L_P$.

On en conclut que $L_u \leq_t^M L_p$ est décidable, Or, L_u n'est pas décidable, donc la propriété P n'est pas décidable.

Conséquences du Théorème de Rice

Par application du théorème de Rice, on peut déduire immédiatement que les problèmes suivants sont indécidables : étant donnée une MT M,

- *est-ce M accepte tous les mots ?*
- *est-ce que $L(M)$ est un langage régulier ?*
- *est-ce M accepte tous les palindromes ?*

On peut par contre décider de propriétés triviales (soit systématiquement fausse, soit systématiquement vrais):

- *est-ce $L(M) = \{a\}$ ou $L(M) \neq \{a\}$?*
- *est-ce que $|L(M)|$ est un nombre impair et divisible par 2?*
- *est-ce que $L(M) = L_{halt}$?*

Et bien sûr, savoir si une proporité est triviale est un problème indécidable...

Table of Contents

Thèse de Church Turing

Nous avons vu que les machines de Turing ne peuvent pas calculer toutes les fonctions, et même qu'elles ne sont capables d'en calculer qu'une infime partie. Il est légitime de se poser les questions suivantes :

- Est-ce un bon modèle de calcul ?
- Ne pourrions nous pas définir un modèle de calcul qui puisse calculer un plus grand ensemble de fonctions ?

Définition

Deux modèles de calcul sont **équivalents** s'ils sont capables de se simuler mutuellement (donc ils calculent exactement le même ensemble de fonctions).

Par exemple les MT multi-rubans sont équivalentes aux MT.

Thèse de Church Turing

Il se trouve que tous les modèles de calcul réalistes qui ont été définis jusqu'à aujourd'hui sont équivalents aux machines de Turing.
Historiquement,

- en 1933 Kurt Gödel et Jacques Herbrand définissent le modèle des fonctions μ-récursives.
- En 1936, Alonzo Church définit le λ-calcul.
- En 1936 (sans avoir connaissance des travaux de Church), Alan Turing propose sa définition de machine. Church et Turing démontrent alors que ces trois modèles de calcul sont équivalents !

La thèse de Church-Turing (ou plutôt ses deux versions) sont des énoncés que la communauté pense vrais, mais qu'il n'est pas possible de prouver. Ils énoncent que les machines de Turing capturent correctement la notion de calcul : toute autre façon de calculer, ou de définir le calcul, reviendrait au même.

Thèse de Church-Turing version physique

Thèse:

Toute fonction physiquement calculable est calculable par une MT.

Autrement dit, tout modèle de machine, qui peut effectivement exister selon les lois de la physique, sera au mieux équivalent aux machines de Turing, sinon moins puissant (en terme d'ensemble de fonctions calculables).
Robin Gandy (qui a effectué son doctorat sous la direction d'Alan Turing) a travaillé sur cette question et démontré un résultat que nous reproduisons ci-dessous dans une formulation simplifiée.

Thèse de Church-Turing version physique

Théorème

Toute fonction calculée par une machine respectant les lois physiques :

1. homogénéité de l'espace (partout les mêmes lois),
2. homogénéité du temps (toujours les mêmes lois),
3. densité d'information bornée (pas plus de n bits au m^2),
4. vitesse de propagation de l'information bornée (pas plus de c $m.s^{-1}$),
5. quiescence (configuration initiale finie et état de repos tout autour),

est calculable par une machine de Turing.

Est-ce satisfaisant ? Une version quantique de ce théorème a été démontrée.

Thèse de Church-Turing version algorithmique

Thèse

Toute fonction calculée par un algorithme est calculable par une MT.

Pas facile de définir ce que pourrait être, un **algorithme**. On peut dire qu'un algorithme est exprimable par un programme rédigé dans un certain langage de programmation, qu'il décrit des instructions pouvant être suivies sans faire appel à une quelconque réflexion.

Remarque

Cette version de la thèse de Church-Turing est parfois appelée *version symbolique*, car elle exprime ce qu'il est possible de calculer à l'aide de symboles mathématiques auxquels on donne un sens calculatoire.

Thèse de Church-Turing version algorithmique

Rappelons qu'Alan Turing est parti de l'idée d'un calculateur humain avec son stylo devant une feuille de papier, pour construire le modèle des machines qui portent son nom. Les machines de Turing sont-elles assez générales ? Ou bien peut-on imaginer une autre façon non-équivalente de décrire les algorithmes, mais qui soit en accord avec notre intuition ?
La Thèse postule que les machines de Turing capturent en toute généralité la notion d'algorithme, et que l'on peut s'en contenter.

Malgré de nombreux efforts, la thèse de Church-Turing n'a jusqu'ici jamais été contredite !