Décidabilité et Indécidabilité. Exemples.

Motivation 1 Introduction aux limites des algorithmes et du calculable

I. Premières notions de calculabilité (Lycée)

Définition 2 **Spécification d'un problème algorithmique** [NSIT 16.2] La spécification d'un problème comporte deux parties :

- la description des entrées admissibles, et
- la description du résultat ou des effets attendus.

A. Les programmes sont des données [NSIT 16 Calculabilité]

Remarque 3 **Programme en tant que donnée** Un programme n'est aussi qu'une donnée manipulée en mémoire.

Exemple 4

- Du code sous chaîne de caractère peut être exécuté avec les programmes `eval` et `exec`.
- Un programme peut être téléchargé `curl`, copié `cp`, déplacé `mv` comme n'importe quel fichier.

Définition 5 **Algorithme / Programme** Dans la suite de la leçon, un programme/algorithme $\texttt{p} \in \mathcal{P}$ sera un code Python/OCaml pouvant être sérialisé en chaîne de caractère et ainsi être passé en entrée d'un autre.

B. Problèmes de décisions

Définition 6 **Problème de décision** Un problème de décision est un problème algorithmique dont le résultat attendu est un booléen : soit vrai, soit faux.

Exemple 7 **Problème du tableau triée** Savoir si un tableau est triée ou non est un problème de décision.

Définition 8 **Notions de terminaisons** [TOR p830 trois voies] Un programme peut avoir plusieurs comportements:

- production d'un résultat
- non-terminaison
- interruption prématurée

Définition 9 **Problème de l'arrêt** [NSIT 16.3]

- Entrée : un programme `p` et une entrée `e`
- Sortie : `True` si l'exécution de `p` sur `e` termine et `False` sinon

Exemple 10 **Indécidabilité du problème de l'arrêt** On démontre par l'absurde qu'il n'existe pas de fonction calculant l'`arret`.

```
def paradoxe(n: int):
  if arret(paradoxe, n):
    while True:
      pass
  else:
    return n
```

```
- Si arret(paradoxe, 0) = True
    alors paradoxe termine sur 0
    mais paradoxe boucle infiniement
- Sinon arret(paradoxe, 0) = False
    alors paradoxe ne termine pas sur 0
    mais paradoxe termine
```

Remarque 11 **On peut prouver la terminaison de fonctions** L'indécidabilité du problème de l'arrêt n'empêche pas de prouver la terminaison de certain problème par l'utilisation de variants par exemple.

Théorème 12 **Complétude des langages** Aucun paradigme de programmation n'est plus expressif qu'un autre.

C. Méthodes

Méthode 13 **Preuve de décidabilité** Pour prouver la décidabilité d'un problème, on peut exhiber une fonction qui le décide.

Méthode 14 **Preuve d'indécidabilité** Pour prouver l'indécidabilité d'un problème, on peut procéder par l'absurde.

II. Réduction et Décidabilité (CPGE) [TOR]

Définition 15 **Problème de décision** Un problème de décision sur un domaine d'entrées E est défini par une fonction totale f de E vers l'ensemble $\mathbb{B}$ des booléens.

A. Problème Décidables / Semi-décidables

Définition 16 **Fonction calculable** Une fonction totale $f : E \to S$ est calculable s'il existe un programme $\texttt{p} \in \mathcal{P}$ tels que pour toute instance $e \in E$. A appliqué à e termine en un temps fini et produit le résultat $f(e)$. On dit alors que `p` résout f.

Théorème 17 Il existe une infinité de fonctions non-calculables.

Définition 18 Problème décidable Un problème est décidable si la fonction f est calculable. C'est-à-dire s'il existe un programme $\texttt{p} \in \mathcal{P}$ qui résout f.

Définition 19 Problème semi-décidable Un problème de décision défini par f est semi-décidable s'il existe un programme p tels que, pour toute instance $e \in E$, le programme p appliqué à E :

- termine en un temps toujours et renvoie V si $f(e) = V$,
- renvoie F, ou ne termine pas

Exemple 20 Le problème de l'arrêt est semi-décidable

Remarque 21 Existence de problèmes non semi-décidables Le problème

- Entrée : un programme $\texttt{p} \in \mathcal{P}$
- Sortie : V si le programme p appliqué à son code source ne renvoie pas V, F sinon

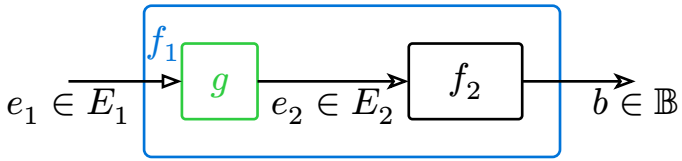
n'est pas semi-décidable par argument diagonal.

Théorème 22 Existence d'un programme universel Il existe un programme universel `eval` qui résout le problème

- Entrée : un programme $\texttt{p} \in \mathcal{P}$ et un argument e pour p
- Sortie : termine et renvoie la valeur produite par `p e` si `p e` termine, ne termine pas sinon.

B. Réductions calculatoires

Définition 23 Réduction calculatoire Soient deux problèmes de décision définis par des fonctions $f_1 : E_1 \to \mathbb{B}$ et $f_2 : E_2 \to \mathbb{B}$. On dit que f_1 se réduit calculatoirement à f_2 et on note $f_1 \leq f_2$, s'il existe une fonction calculable $g : E_1 \to E_2$ telle que pour tout $e \in E_1$, on a $f_1(e) = f_2(g(e))$.



Théorème 24 Réduction à un problème décidable Considérons un problème décidable défini par une fonction $f_1 : E_1 \to \mathbb{B}$, et un problème de décision défini par une fonction $f_2 : E_2 \to \mathbb{B}$ et tel que $f_2 \leq f_1$ alors f_2 est décidable.

Théorème 25 Réduction d'un problème indécidable Considérons un problème indécidable défini par une fonction $f_1 : E_1 \to \mathbb{B}$, et un problème de décision défini par une fonction $f_2 : E_2 \to \mathbb{B}$ et tel que $f_1 \leq f_2$ alors f_2 est indécidable.

Exemple 26 Indécidabilité de la trivialité d'un programme Le problème

- Entrée : un programme $\texttt{p} \in \mathcal{P}$
- Sortie : V si p renvoie V sur toute entrée, F sinon

est indécidable.

Exemple 27 Indécidabilité de l'équivalence sémantique Le problème

- Entrée : deux programmes $\texttt{p\_1} \in \mathcal{P}$ et $\texttt{p\_2} \in \mathcal{P}$ avec même domaine d'entrée E et de sortie S
- Sortie : vrai si pour toute entrée $e \in E$ les comportements de `p_1` et `p_2` appliqué à l'entrée e sont identiques

est indécidable.

DEV

Application 28 Théorème de Rice Soit une fonction totale $f : \mathcal{P} \to \mathbb{B}$ prenant en entrée un programme. On suppose que f est :

- non triviale, c'est-à-dire surjective
- sémantique, c'est-à-dire que, pour tout $\texttt{p\_1}, \texttt{p\_2} \in \mathcal{P}^2$ sémantiquement équivalents, on a $f(\texttt{p\_1}) = f(\texttt{p\_2})$.

Alors, le problème de décision défini par f est indécidable.

Exemple 29 PCP (Problème de Correspondance de Post)

- Une instance est une suite $(u_1, v_1), ..., (u_m, v_m)$ de paires de mots sur Σ
- Une solution est une suite $(i_1, ..., i_n)$ de $\{1, m\}$ tels que $u_{i_1}...u_{i_n} = v_{i_1}..v_{i_n}$
- Une instance est positive si elle a au moins une solution.

Remarque 30 Le Problème de Correspondance de Post est indécidable (admis).

III. Décidabilité en Licence [CAR]

A. Notions de codages injectifs

Définition 31 Langage d'un problème Pour associer un langage à un problème, on utilise un codage qui est une fonction naturelle injective de l'ensemble des instances E dans Σ^*. Le langage associé à un problème est l'ensemble des codages de ses instances positives.

Exemple 32 Codage d'un entier Représentation en base 1, 2, 10, 16.

Exemple 33 Codage d'un graphe Pour $\Sigma = \{0, 1, (,), ;\}$ On associe à chaque sommet u un codage binaire $< u >$ puis chaque arête $(u, v) \in A$ est codée $(< u >; < v >)$ donc $< G >= ((0; 1; 10); (0; 1); (1; 0))$.

B. Des problèmes aux machines

Définition 34 Machine de Turing (MT) (déterministe, à un ruban) Une machine de Turing est un septuplet $(\Sigma, \Gamma, Q, q_0, q_f, \delta, \#)$

- Σ l'alphabet d'entrée, $\# \notin \Sigma$
- Γ l'alphabet du ruban, $\# \in \Gamma$
- Q un ensemble fini d'états dont q_0 l'initial et q_f l'acceptant
- $\delta : \Gamma \times Q \to \Gamma \times Q \times \{\leftarrow, \rightarrow\}$ la fonction partielle de transition
- $\#$ est le symbole blanc qui remplit initialement le ruban

Définition 35 Comportement d'une machine de Turing Une MT M travaille sur un état courant $q \in Q$, un ruban infini contenant un mot de $\Gamma*$ et sur lequel se déplace un curseur. Lancée sur un mot $w \in \Sigma*$ on commence à l'état q_0 avec w sur le ruban et le curseur au début de w

- on note uqv une configuration de M ($\varepsilon q_0 w$ initiale)
- δ permet de déterminer l'éventuelle configuration suivante
- si la suite des configurations est infinie, on dit que M diverge.
- sinon on dit que M s'arrête sur un état acceptant si $q = q_f$ ou refusant sinon.

Définition 36 Langage accepté d'une MT Le language accepté par une MT M est $\mathcal{L}_\in(M) = \{w \mid M \text{ accepte } w\}$

Définition 37 Langage décidable Un langage est décidable si il existe une machine de Turing qui le décide.

Définition 38 Re et R On définit

- $RE = \{\mathcal{L} \subseteq \Sigma* \mid \exists \text{ M}, \ \mathcal{L}_\in(M) = \mathcal{L}\}$
- $R = \{\mathcal{L} \subseteq \Sigma* \mid \exists \text{ M qui ne diverge pas}, \ \mathcal{L}_\in(M) = \mathcal{L}\}$

Proposition 39 Clôture de R. R est clos par union intersection et complémentaire.

Proposition 40 Clôture de RE. RE est clos par union intersection.

C. Exemples

DEV **Théorème 41 Indécidabilité du PCP** [SIP]

Définition 42 Arithmétique de Presburger L'arithmétique de Presburger est la théorie du premier ordre des entiers munis de l'addition. C'est le langage des formules pour lesquelles on dispose

- des constances 0 et 1
- $+$ comme symbole de fonction binaire
- $=$ comme prédicat binaire d'égalité

DEV **Théorème 43 PRES est décidable** Le problème

- Entrée : une formule de l'arithmétique de Presburger
- Sortie : vrai si la formule est vraie sur les entiers

est décidable.

Définition 44 Arithmétique de Peano En ajoutant à l'arithmétique de Presburger le symbole de fonction $\times$ de la multiplication, on obtient l'arithmétique de Peano.

Théorème 45 L'arithmétique de Peano est indécidable (admis) Le problème

- Entrée : une formule de l'arithmétique de Peano
- Sortie : vrai si la formule est vraie sur les entiers

est indécidable.

Liens P = NP (étude restreinte à la complexité)

Décidabilité et Indécidabilité. Exemples.

1 Motiv

I. Premières notions de calculabilité (Lycée)

2 Def Spécification d'un problème algorithmique [NSIT 16.2]

A. Les programmes sont des données [NSIT 16 Calculabilité]

3 Rem Programme en tant que donnée

4 Ex

5 Def Algorithme / Programme

B. Problèmes de décisions

6 Def Problème de décision

7 Ex Problème du tableau triée

8 Def Notions de terminaisons [TOR p830 trois voies]

9 ~~Def Problème de l'arrêt [NSIT 16.3]~~

10 Ex Indécidabilité du problème de l'arrêt

11 Rem On peut prouver la terminaison de fonctions

12 Thm Complétude des langages

C. Méthodes

13 Métho Preuve de décidabilité

14 Métho Preuve d'indécidabilité

II. Réduction et Décidabilité (CPGE) [TOR]

15 Def Problème de décision

A. Problème Décidables / Semi-décidables

16 Def Fonction calculable

17 Thm

18 Def Problème décidable

19 Def Problème semi-décidable

20 Ex Le problème de l'arrêt est semi-décidable

21 Rem Existence de problèmes non semi-décidables

22 Thm Existence d'un programme universel

B. Réductions calculatoires

23 Def Réduction calculatoire

24 Thm Réduction à un problème décidable

25 Thm Réduction d'un problème indécidable

26 Ex Indécidabilité de la trivialité d'un programme

27 Ex Indécidabilité de l'équivalence sémantique

28 App Théorème de Rice

29 Ex PCP (Problème de Correspondance de Post)

30 Rem

III. Décidabilité en Licence [CAR]

A. Notions de codages injectifs

31 Def Langage d'un problème

32 Ex Codage d'un entier

33 Ex Codage d'un graphe

B. Des problèmes aux machines

34 Def Machine de Turing (MT)

35 Def Comportement d'une machine de Turing

36 Def Langage accepté d'une MT

37 Def Langage décidable

38 Def Re et R

39 Prop Clôture de R.

40 Prop Clôture de RE.

C. Exemples

41 Thm Indécidabilité du PCP

42 Def Arithmétique de Presburger

43 Thm PRES est décidable

44 Def Arithmétique de Peano

45 Thm L'arithmétique de Peano est indécidable (admis)

Commentaires

Les machines de Turing étant explicitement hors-programme MPI, elles n'apparaissent qu'en ouverture, en troisième partie de la leçon.

- Pour la partie I, il suffit de suivre la tortue lycée même si il faut faire des grands sauts parfois.
- Pour la partie II, il est aisé de suivre la Tortue Prépa et de mettre un item pour chaque encadré de celle-ci.

ATTENTION on remplacera la notion d'algorithme et de fonction OCaml par la notion de programme $\texttt{p} \in \mathcal{P}$

- Pour la partie III A) et B), il faut suivre le carton de la même manière. La partie III B) se trouve au chapitre « décidabilité des théories logiques » du carton.

Il faut faire attention entre la confusion des « fonctions » mathématiques calculables ou pas, et les « fonctions » informatique, qui sont des programmes.

Bibliographie

[NSIT] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen, *Numériques et Sciences Informatiques Terminale.*

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés.*

[CAR] O. Carton & G. G. Gagnees, *Langages formels: Calculabilité et complexité.*

[SIP] M. Sipser, *Introduction to the Theory of Computation.*