Notions d'Indécidabilité

Problèmes de Décision : Définition

Problèmes de Décision (définition informelle)

Un problème de décision est un problème dont la réponse est binaire (0 ou 1).

Exemple (problème **PREM**) : est-ce qu'un nombre entier est premier ?

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
└Notions d'Indécidabilité

Problèmes de Décision : Définition

Problèmes de Décision (définition informelle)

Un problème de décision est un problème dont la réponse est binaire (0 ou 1).

Exemple (problème **PREM**) : est-ce qu'un nombre entier est premier ?

Problèmes de Décision (définition moins informelle)

Un problème de décision est un ensemble de mots sur un alphabet (fini).

Exemple

- un entier naturel peut être vu comme un mot (une chaîne de caractères) sur l'alphabet $\{0, 1\}$ (sa représentation binaire). Le mot 1001 représente l'entier 9.
- le problème **PREM** correspond à l'ensemble (infini) des représentations binaires d'entiers premiers, i.e.

$$\textbf{PREM} = \{1, 11, 101, 111, \ldots\}$$

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
└Notions d'Indécidabilité

Problèmes Indécidables : Définition

Problème Indécidable

Un problème de décision P sur un alphabet Σ est indécidable s'il n'existe pas d'algorithme[2] A prenant un mot sur Σ et retournant la valeur 0 ou 1, tel que pour tout mot x sur Σ, $A(x)$ retourne 1 si et seulement si $x \in P$.

Autrement dit, un problème est indécidable s'il n'existe pas d'algorithme pour le résoudre.
Nous allons voir deux exemples.

2. Entendez par algorithme un programme Java, C, Python ... La formalisation des notions d'algorithme et de problème n'est pas l'objet de ce cours, mais sera vue en Master dans le cours Calculabilité et Complexité.

Problème de l'Arrêt

Exemple de programmes qui ne s'arrête pas :

```
while(true) print "bonjour" ;
```

Définition

ENTREE : le code source d'un programme P lisant des chaînes de caractères, et une chaîne de caractères x
SORTIE : 1 si et seulement si le programme P s'arrête sur l'entrée x, et 0 sinon.

3. Pour une preuve formelle de ce résultat, il faudrait formaliser la notion d'algorithme. C'est le but des machines de Turing qui ne sont pas l'objet de ce cours.

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
Notions d'Indécidabilité
Problème de l'Arrêt

Problème de l'Arrêt

Exemple de programmes qui ne s'arrête pas :

```
while(true) print "bonjour" ;
```

Définition

ENTREE : le code source d'un programme P lisant des chaînes de caractères, et une chaîne de caractères x
SORTIE : 1 si et seulement si le programme P s'arrête sur l'entrée x, et 0 sinon.

Théorème (Turing, 1936)

Le problème de l'arrêt est indécidable[3].

3. Pour une preuve formelle de ce résultat, il faudrait formaliser la notion d'algorithme. C'est le but des machines de Turing qui ne sont pas l'objet de ce cours.

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
Notions d'Indécidabilité
Problème de l'Arrêt

Preuve d'Indécidabilité du Problème de l'Arrêt

On procède par l'absurde, en supposant qu'il existe un programme $HALT(P, x)$ qui décide le problème de l'arrêt, pour tout P, x donnés en entrée.
A partir de $HALT$, on définit le programme $PARADOX$ suivant :

$PARADOX(c : string)$
  if HALT(c,c) then loop forever
    else stop

On appelle $PARADOX(PARADOX)$, ou plus précisément, on exécute le programme $PARADOX$ sur son code source (qui est une chaîne de caractères)... que se passe-t-il ?

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
Notions d'Indécidabilité
Problème de l'Arrêt

Preuve d'Indécidabilité du Problème de l'Arrêt

Deux cas :

- si $PARADOX(PARADOX)$ s'arrête, alors c'est que $HALT(PARADOX, PARADOX) = 0$, i.e. $PARADOX(PARADOX)$ ne s'arrête pas.
- si $PARADOX(PARADOX)$ ne s'arrête pas, alors c'est que $HALT(PARADOX, PARADOX) = 1$, i.e. $PARADOX(PARADOX)$ s'arrête.

Dans les deux cas on obtient une contradiction, c'est donc que le programme $HALT$ n'existe pas.

Indécidabilité de la logique du premier ordre

- Nous allons montrer qu'il n'existe pas d'algorithme A qui prend en entrée une formule de la logique du premier ordre et retourne 1 si elle est valide, 0 sinon.

Notion de Réduction

- Pour montrer l'indécidabilité d'un problème de décision[4] P_1, on peut partir d'un problème de décision P_2 indécidable et on montre l'existence d'un algorithme, appelé *réduction*, qui pour toute instance I_2 de P_2 construit une instance I_1 de P_1 telle que I_1 a une solution si et seulement si I_2 a une solution.
- De cette façon, on montre que P_1 est "aussi difficile" que P_2. S'il existait un algorithme pour résoudre P_1, alors cela nous donnerait un algorithme pour résoudre P_2 : appliquer la réduction puis l'algorithme pour P_1. Cela contredirait le fait que P_2 soit indécidable, donc il n'existe pas d'algorithme pour résoudre P_1.

4. i.e. dont la réponse est 0 ou 1

Notion de Réduction

- Pour montrer l'indécidabilité d'un problème de décision[4] P_1, on peut partir d'un problème de décision P_2 indécidable et on montre l'existence d'un algorithme, appelé *réduction*, qui pour toute instance I_2 de P_2 construit une instance I_1 de P_1 telle que I_1 a une solution si et seulement si I_2 a une solution.
- De cette façon, on montre que P_1 est "aussi difficile" que P_2. S'il existait un algorithme pour résoudre P_1, alors cela nous donnerait un algorithme pour résoudre P_2 : appliquer la réduction puis l'algorithme pour P_1. Cela contredirait le fait que P_2 soit indécidable, donc il n'existe pas d'algorithme pour résoudre P_1.
- Certains problèmes ont directement été montré indécidables : le problème de terminaison d'un programme par exemple, mais aussi le problème de validité dans la logique du premier ordre (travaux d'Alonzo Church et Alan Turing).
- ici, nous allons utiliser une réduction à partir du problème de *correspondance de Post*.

4. i.e. dont la réponse est 0 ou 1

Problème de la Correspondance de Post (PCP)

Soit $\Sigma = \{0, 1\}$. Un mot u sur Σ est une séquence finie d'éléments de Σ. Deux mots u et v peuvent être concaténés pour former un nouveau mot noté uv. L'élément neutre pour la concaténation est noté ϵ (mot vide). Par exemple : $u = 01$ est un mot, $v = 110$ est un mot, $uv = 01110$, $\epsilon u = u\epsilon = u = 01$.
Le problème de la Correspondance de Post (PCP) se formule de la manière suivante :

Entrée

$(u_1, v_1), \ldots, (u_n, v_n)$, $n \geq 1$, n paires de mots (possiblement vides) sur Σ.

Sortie

1 si et seulement si il existe une séquence finie d'indices $i_1, \ldots, i_k \in \{1, \ldots, n\}$ telle que $k \geq 1$ et

$$u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$$

PCP : exemples

- Instance : $(u_1, v_1) = (100, 00),\ (u_2, v_2) = (0, 01)$,

PCP : exemples

- Instance : $(u_1, v_1) = (100, 00),\ (u_2, v_2) = (0, 01)$, Solution (parmi d'autres) : 2,1

$$u_2 u_1 = 0100 = v_2 v_1$$

- Instance : $(u_1, v_1) = (1, 100),\ (x_2, v_2) = (0, \epsilon)$,

PCP : exemples

- Instance : $(u_1, v_1) = (100, 00),\ (u_2, v_2) = (0, 01)$, Solution (parmi d'autres) : 2,1

$$u_2 u_1 = 0100 = v_2 v_1$$

- Instance : $(u_1, v_1) = (1, 100),\ (x_2, v_2) = (0, \epsilon)$, Solution : 1,2,2

$$u_1 u_2 u_2 = 100 = v_1 v_2 v_2$$

- Instance : $(u_1, v_1) = (1, 0),\ (u_2, v_2) = (0, 1)$,

PCP : exemples

- Instance : $(u_1, v_1) = (100, 00),\ (u_2, v_2) = (0, 01)$, Solution (parmi d'autres) : 2,1

$$u_2 u_1 = 0100 = v_2 v_1$$

- Instance : $(u_1, v_1) = (1, 100),\ (x_2, v_2) = (0, \epsilon)$, Solution : 1,2,2

$$u_1 u_2 u_2 = 100 = v_1 v_2 v_2$$

- Instance : $(u_1, v_1) = (1, 0),\ (u_2, v_2) = (0, 1)$, Pas de solution
- Instance :
  $(u_1, v_1) = (0, 100),\ (u_2, v_2) = (01, 00),\ (u_3, v_3) = (110, 11)$?

PCP : exemples

- Instance : $(u_1, v_1) = (100, 00),\ (u_2, v_2) = (0, 01)$, Solution (parmi d'autres) : 2,1

$$u_2 u_1 = 0100 = v_2 v_1$$

- Instance : $(u_1, v_1) = (1, 100),\ (x_2, v_2) = (0, \epsilon)$, Solution : 1,2,2

$$u_1 u_2 u_2 = 100 = v_1 v_2 v_2$$

- Instance : $(u_1, v_1) = (1, 0),\ (u_2, v_2) = (0, 1)$, Pas de solution
- Instance : $(u_1, v_1) = (0, 100),\ (u_2, v_2) = (01, 00),\ (u_3, v_3) = (110, 11)$?

$u_3 u_2 u_3 u_1$=(110)(01)(110)(0)=110011100=(11)(00)(11)(100)=$v_3 v_2 v_3 v_1$

Théorèmes

Théorème

Le problème de Correspondance de Post est indécidable.

Preuve admise.

Théorème

Le problème de validité en logique du premier ordre est indécidable.

Nous allons montrer ce résultat en réduisant PCP.

Preuve

Nous considérons le langage du premier ordre contenant un prédicat binaire p, deux symboles de fonction unaires f_0 et f_1, et un symbole de constante a. Etant donné un mot u sur Σ et un terme g, on définit le terme $t_u(g)$ sur L par :

- g si $u = \epsilon$
- $f_{b_s}(f_{b_{s-1}} \dots (f_{b_2}(f_{b_1}(g))))$ si $u = b_1 b_2 \dots b_s$.

Soit $I = \{(u_1, v_1), \dots, (u_n, v_n)\}$ une instance de PCP. Nous allons construire une formule ϕ_I de la logique du premier ordre, sur le langage L, et montrer que ϕ_I est valide si et seulement si I a une solution. La formule ϕ_I se décompose comme suit :

$$\phi_I = \rho \wedge \sigma \rightarrow \tau$$

Premièrement,

$$\rho \equiv p(t_{u_1}(a), t_{v_1}(a)) \wedge p(t_{u_2}(a), t_{v_2}(a)) \wedge \dots \wedge p(t_{u_n}(a), t_{v_n}(a))$$

Preuve

Cela signifie qu'on met dans la relation p les termes qui correspondent aux paires de mots. On peut étendre cette relation aux concaténations (paires à paires) de couples de mots de I. En particulier, la formule suivante signifie que si x et y sont en relation, alors on peut concaténer u_i à x et v_j à y, pourvu que $i = j$.

$$\sigma \equiv \forall x \forall y.(p(x, y) \rightarrow \bigwedge_{i=1}^{n} p(t_{u_i}(x), t_{v_i}(y))$$

Enfin, τ signifie l'existence de deux éléments égaux en relation :

$$\tau \equiv \exists z.p(z, z)$$

Clairement, cette réduction est effective (on peut écrire un algorithme qui l'implémente). Montrons qu'elle est correcte.

Preuve

Supposons que I a une solution $i_1, \ldots, i_k$ et montrons que ϕ_I est valide. Pour cela, soit $\mathcal{M}$ une structure sur L. Supposons que $\mathcal{M} \models \rho \wedge \sigma$. On sait par hypothèse que

$$u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$$

Donc $t_{u_{i_k}}(t_{u_{i_{k-1}}}(\ldots(t_{u_{i_1}}(a))) = t_{v_{i_k}}(t_{v_{i_{k-1}}}(\ldots(t_{v_{i_1}}(a))) = g$ pour un certain g. Par hypothèse, on sait que $\mathcal{M} \models p(t_{u_{i_1}}, t_{v_{i_1}})$ et comme $\mathcal{M} \models \sigma$, on obtient aussi que $\mathcal{M} \models p(t_{u_{i_2}}(t_{u_{i_1}}), t_{v_{i_2}}(t_{v_{i_1}}))$, et plus généralement, que $\mathcal{M} \models p(g, g)$. Donc $\mathcal{M} \models \exists z.p(z, z)$.

Preuve

Réciproquement, supposons que ϕ_I est valide et montrons que I a une solution. On défini une structure H comme ceci :

- son domaine est l'ensemble des termes clos sur L (sans variables)
- les fonctions et constantes sont interprétées par elles-mêmes ($f^H(t) = f(t)$)
- l'interprétation p^H de p est définie inductivement : $p^H(a, a) = 1$ et pour tous termes clos h et g, $p^H(t_{u_i}(g), t_{v_j}(h)) = 1$ si et seulement si $i = j$ et $p^H(g, h) = 1$.

Clairement, $H \models \rho \wedge \sigma$, donc $H \models \tau$ puisque ϕ_I est valide. Donc il existe un terme clos t tel que $H \models p(t, t)$. Par définition de H, il est facile de voir que t se décompose nécessairement en $t = t_{u_{i_k}}(t_{u_{i_{k-1}}}(\ldots(t_{u_{i_1}}(a))) = t_{v_{i_k}}(t_{v_{i_{k-1}}}(\ldots(t_{v_{i_1}}(a)))$. Donc

$$u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$$

et I a une solution. □

Autres Problèmes Indécidables

- **10é problème de Hilbert** : résolution d'équations diophantiennes (à solution entière). Indécidabilité prouvée en 1971 (Matiyasevic).

Entrée $p(x_1, \ldots, x_n)$ un polynôme à coefficients entiers
Sortie $\exists i_1 \ldots \exists i_n \in \mathbb{Z} \cdot p(i_1, \ldots, i_n) = 0$?

Remarque : c'est décidable dans les réels (Tarski, 1951).

Autres Problèmes Indécidables

- **10é problème de Hilbert** : résolution d'équations diophantiennes (à solution entière). Indécidabilité prouvée en 1971 (Matiyasevic).

Entrée $p(x_1, \ldots, x_n)$ un polynôme à coefficients entiers
Sortie $\exists i_1 \ldots \exists i_n \in \mathbb{Z} \cdot p(i_1, \ldots, i_n) = 0$?

Remarque : c'est décidable dans les réels (Tarski, 1951).

- **Complexité de Kolmogorov** : étant donné un mot binaire w et un entier $k \in \mathbb{N}$, décider s'il existe un programme Java qui écrit w et dont le code est un fichier d'au plus k bits. **Remarque** : le choix du formalisme qui écrit le mot (Java,Lisp,C,machine de Turing) influe sur la complexité.

Autres Problèmes Indécidables

- **10é problème de Hilbert** : résolution d'équations diophantiennes (à solution entière). Indécidabilité prouvée en 1971 (Matiyasevic).

  Entrée $p(x_1, \ldots, x_n)$ un polynôme à coefficients entiers
  Sortie $\exists i_1 \ldots \exists i_n \in \mathbb{Z} \cdot p(i_1, \ldots, i_n) = 0$?

  Remarque : c'est décidable dans les réels (Tarski, 1951).
- **Complexité de Kolmogorov** : étant donné un mot binaire w et un entier $k \in \mathbb{N}$, décider s'il existe un programme Java qui écrit w et dont le code est un fichier d'au plus k bits. **Remarque** : le choix du formalisme qui écrit le mot (Java,Lisp,C,machine de Turing) influe sur la complexité.
- **Intersection de deux automates à pile** : étant donnés deux automates à pile A et B, décider sur l'intersection de leurs langages $L(A) \cap L(B)$ est vide.

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
Notions d'Indécidabilité
Indécidabilité du problème de validité

Autres Problèmes Indécidables

- **PCP avec** $n = 7$: dans cette variante on se donne 7 paires de mots $(u_1, v_1), \ldots, (u_7, v_7)$. **Remarque :** décidable pour $n \leq 2$, ouvert pour $3 \leq n \leq 6$.
- **Castor affairé :** étant donné un programme (donné par son code source, par exemple Java) qui écrit des 0 et des 1 et qui s'arrête, on veut décider si il est maximal, i.e. qu'il n'existe pas un autre programme dont le code source n'est pas plus grand, mais qui écrit un mot strictement plus grand.

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
Notions d'Indécidabilité
Indécidabilité du problème de validité

Exercices

1. Montrer que PCP sur un alphabet à une lettre est équivalent au problème de décider si une équation linéaire sur les entiers naturels a une solution non-triviale. En déduire que PCP sur un alphabet à une lettre est décidable.
2. Montrer que le problème suivant est indécidable. Etant donnés deux programmes P et Q qui retournent tous les deux une valeur Booléenne, est-ce que les deux programmes sont équivalents, i.e. est-ce que pour toute entrée x, on a $P(x) = Q(x)$?

INFO-F-302, Cours d'Informatique Fondamentale *Logique pour l'Informatique*
Notions d'Indécidabilité
Indécidabilité du problème de validité

Exercices

1. Montrer que PCP sur un alphabet à une lettre est équivalent au problème de décider si une équation linéaire sur les entiers naturels a une solution non-triviale. En déduire que PCP sur un alphabet à une lettre est décidable.
2. Montrer que le problème suivant est indécidable. Etant donnés deux programmes P et Q qui retournent tous les deux une valeur Booléenne, est-ce que les deux programmes sont équivalents, i.e. est-ce que pour toute entrée x, on a $P(x) = Q(x)$?
   - on va faire une réduction à partir du problème de l'arrêt, qui est indécidable.
   - étant donné un programme R, on considère les deux programmes :

     ```
     procedure AUX(x):
        r := R(x) ;
        return 1 ;
     ```

     ```
     procedure CONST(x):
       return 1;
     ```
   - pour toute entrée x, on a $AUX(x) = CONST(x)$ si et seulement si $R(x)$ s'arrête sur l'entrée x. Si le problème d'équivalence était décidable, alors le problème de l'arrêt le serait aussi.