I. Généralités et cadre

1. Classes R et RE

Déf 1: Une machine de Turing est un heptuplet $(Q, \Gamma, \Sigma, \delta, q_0, \#, F)$ où
- Q est un ensemble fini d'états;
- Γ est l'alphabet de ruban;
- $\Sigma \subseteq \Gamma$ est l'alphabet d'entrée;
- $q_0 \in Q$ est l'état initial;
- $F \subseteq Q$ est l'ensemble des états acceptants;
- $\# \in \Gamma \setminus \Sigma$ est le symbole blanc;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ est la fonction de transition.

Déf 2: Un problème de décision est une question dont la réponse est soit "oui", soit "non".

Ex 3: Étant donné un graphe G, "est-ce que G est connexe ?" est un problème de décision.

Déf 4: Un mot $w \in \Sigma^*$ est accepté par une machine de Turing M si l'exécution de M avec la configuration initiale $q_0 w$ mène à une configuration uqv où $q \in F$. Le langage accepté par une machine M est l'ensemble des mots de Σ^* acceptés par M. On le note $\mathcal{L}(M)$.

Déf 5: L'ensemble des langages récursivement énumérables, RE, est l'ensemble des langages acceptés par une machine de Turing, i.e. les langages L pour lesquels il existe une machine de Turing M telle que $L = \mathcal{L}(M)$.

Ex 6: Le problème ARRET:
- entrée : machine de Turing M, mot w
- sortie : oui si $M(w)$ s'arrête, non sinon

est dans RE.

Déf 7: L'ensemble des langages récursifs, R, est l'ensemble des langages décidés par une machine de Turing, i.e. les langages L pour lesquels il existe une machine de Turing M sans calcul infini telle que $L = \mathcal{L}(M)$.

Ex 8: Le problème CONNEXE:
- entrée : graphe G
- sortie : oui si G est connexe, non sinon

est dans R car on peut exhiber un algorithme qui décide si G est connexe.

Déf 9: Un énumérateur est une machine de Turing qui énumère des mots de Σ^*, i.e. qui écrit sur une bande de sortie des mots de Σ^* séparés par un symbole qui n'est pas dans Σ.

Prop 10: $L \in RE$ ssi il existe un énumérateur qui énumère exactement les mots de L.

Déf 11: On définit $co\text{-}RE = \{A \mid \bar{A} \in RE\}$ où

$\bar{A}$:
- entrée : une instance de A, w
- sortie : oui si $w \notin A$, non sinon

Prop 12: $R = RE \cap co\text{-}RE$. En particulier, $R \subseteq RE$.

Prop 13: R et RE sont stables par union et intersection. De plus, R est stable par complémentaire.

2. Fonctions calculables et réduction

Jusqu'à présent, nous avons considérés les machines de Turing comme des automates acceptant un langage. Mais les machines de Turing peuvent aussi calculer des fonctions :

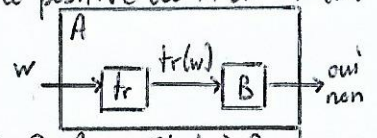
Déf 14: Une machine de Turing calcule une fonction $f : \Sigma^* \to \Sigma^*$ si pour tout mot d'entrée w, elle s'arrête toujours dans une configuration où $f(w)$ se trouve sur le ruban. Une fonction f est calculable s'il existe une machine de Turing qui la calcule.

Thm 15: Les fonctions μ-récursives sont exactement celles calculables par une machine de Turing.

Thm 16: Soit $A \subseteq \mathbb{N}$, les assertions suivantes sont équivalentes:
(i) $A \in RE$
(ii) $A = \pi_2^2(B)$ où $B \subseteq \mathbb{N}^2$ est primitif récursif
(iii) $A = \emptyset$ ou A est l'image d'une fonction primitive récursive
(iv) A est l'image d'une fonction μ-récursive.

DEV 1

Déf 17: Une réduction d'un problème A à un problème B est une fonction calculable tr telle que pour toute instance w de A, w est une instance positive de A ssi $tr(w)$ est une instance positive de B.



Prop 18: Si A se réduit à B, alors : $B \in R \Rightarrow A \in R$
$A \notin R \Rightarrow B \notin R$

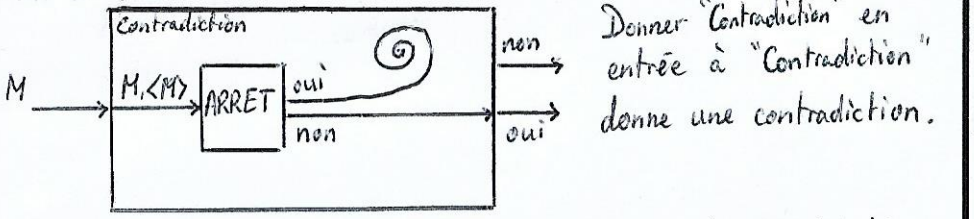
II. Techniques pour prouver l'indécidabilité

Rmq 19: Pour prouver qu'un problème est décidable, il suffit de trouver un algorithme effectif qui décide le problème, ou bien on peut le réduire à un problème que l'on sait décidable d'après la proposition 18.
On s'intéresse ici aux problèmes indécidables.

1. Le problème ARRET

Thm 20: ARRET $\notin \mathcal{R}$.

Idée de la démonstration : construire la machine "Contradiction".



Donner "Contradiction" en entrée à "Contradiction" donne une contradiction.

2. Utilisation de la réduction pour prouver l'indécidabilité

Déf 21: On définit le problème de l'acceptation d'un mot par une machine de Turing

ACCEPTATION :
- entrée : machine de Turing M, mot w
- sortie : oui si M accepte w, non sinon

Thm 22: ACCEPTATION est indécidable.
(Par réduction d'ARRET à ACCEPTATION.)

Déf 23: Problèmes de correspondances de Post :

POST :
- entrée : alphabet fini Σ, famille de tuiles $((u_1,v_1),\dots,(u_N,v_N))$
- sortie : oui s'il existe $i_1,\dots,i_p \in \{1,\dots,N\}$ tels que $p \geq 1$ et $u_{i_1}\dots u_{i_p} = v_{i_1}\dots v_{i_p}$, non sinon

$\text{POST}_{\text{marqué}}$:
- entrée : alphabet fini Σ, famille de tuiles $((u_1,v_1),\dots,(u_N,v_N))$
- sortie : oui s'il existe $i_1,\dots,i_p \in \{1,\dots,N\}$ tels que $p \geq 1$ et $u_1 u_{i_2}\dots u_{i_p} = v_1 v_{i_2}\dots v_{i_p}$, non sinon

Thm 24: $\text{POST}_{\text{marqué}}$ est indécidable (par réduction de ACCEPTATION à $\text{POST}_{\text{marqué}}$). Et POST est indécidable (par réduction de $\text{POST}_{\text{marqué}}$ à POST).

III. Décidabilité et indécidabilité de différents problèmes.

1. Machines de Turing

Rmq 25: On a déjà vu que les problèmes ARRET et ACCEPTATION sont indécidables.

Thm 26 (de Rice): Soit $\mathcal{P}$ telle que $\emptyset \subsetneq \mathcal{P} \subsetneq \mathcal{RE}$.
Alors le problème

$P_{\mathcal{P}}$:
- entrée : machine de Turing M
- sortie : oui si $\mathcal{L}(M) \in \mathcal{P}$, non sinon

est indécidable.

Appli 27: Le problème du langage vide

VIDE :
- entrée : machine de Turing M
- sortie : oui si $\mathcal{L}(M) = \emptyset$, non sinon

est indécidable (appliquer le théorème de Rice à $\mathcal{P} = \{\emptyset\}$).

2. Automates finis

Thm 28: Le problème ACCEPTATION pour les automates finis est décidable.

Prop 29: Le problème VIDE pour les automates finis est décidable.

Prop 30: Le problème du langage universel pour les automates finis

UNIV :
- entrée : langage L sur un alphabet Σ
- sortie : oui si $L = \Sigma^*$, non sinon

est décidable.

3. Grammaires algébriques

Thm 31: Le problème ACCEPTATION pour les langages algébriques est décidable via l'algorithme de Cocke - Younger - Kasami.

Prop 32: Le problème VIDE pour les langages algébriques est également décidable.

Prop 33: Le problème

INTERSECTION:
| entrée: G, G' deux grammaires algébriques
| sortie: oui si $\mathcal{L}(G) \cap \mathcal{L}(G') = \emptyset$, non sinon

est indécidable.

Prop 34: Le problème

EGAL:
| entrée: G, G' deux grammaires algébriques
| sortie: oui si $\mathcal{L}(G) = \mathcal{L}(G')$, non sinon

est indécidable.

Cor 35: Le problème UNIV pour les langages algébriques est aussi indécidable.

4. Logique

Déf 36: On définit le problème

VALIDE:
| entrée: formule close φ
| sortie: oui si φ est valide, non sinon

Prop 37: Le problème VALIDE pour le calcul propositionnel est décidable.

Thm 38: En logique du premier ordre, le problème VALIDE est indécidable. DEV 2

Déf 39: L'arithmétique de Presburger est la théorie du premier ordre des entiers munis de l'addition mais pas de la multiplication.
C'est le langage L engendré par $\{0, 1, +, =\}$ où
- 0 et 1 sont des symboles de constantes
- $+$ est un symbole de fonction binaire
- $=$ est le prédicat binaire de l'égalité.

Prop 40: Le problème

PRES
| entrée: formule close φ de L
| sortie: oui si $\mathbb{N} \models \varphi$, non sinon

est décidable.

Déf 41: Si on ajoute à L le symbole de fonction $\times$ et aux axiomes de l'arithmétique de Presburger ceux relatifs à la multiplication sur les entiers, on obtient la théorie de l'arithmétique de Peano.

Thm 42: La théorie de Peano n'est pas décidable.
(Admis)

Références:

- Autebert, *Calculabilité et décidabilité, une introduction*
- Carton, *Langages formels, calculabilité et complexité*
- Cori-Lascar, *Logique mathématique*, tome II.
- Sipser, *Introduction to the theory of computation*
- Wolper, *Introduction à la calculabilité*