Calculabilité
Cours 3 : réductions

Kévin Perrot – Aix Marseille Université – printemps 2019

Table des matières

3.7 Réductions (many-one)

Une des formulations les plus populaires du résultat de Turing en 1936 est donnée par le théorème de l'arrêt. Voyons maintenant une façon plus épurée de formuler ces idées, qui nous permettront d'aller plus loin de façon élégante.

Théorème 1. *Le langage* $L_d = \{\langle M\rangle \mid M$ *n'accepte pas le mot* $\langle M\rangle\}$ *n'est pas re.*

Démonstration. 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. □

La preuve est cette fois encore plus simple ! Ce résultat nous dit qu'il n'existe pas de MT M_d (un seul et même algorithme qui répond oui/non correctement pour chaque instance) pour décider si une machine M reconnait le mot $\langle M\rangle$ (même si la machine M_d a le droit de ne pas s'arrêter si M ne reconnait pas $\langle M\rangle$). Ce problème peut sembler artificiel, mais il sert de *graine* pour dériver la non récursivité d'autres problèmes (plus naturels), via une méthode qui s'appelle une **réduction**.

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.

Définition 2. *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* $A \leq_m^T B$.

On appelle *instance* de L un mot $w \in \Sigma_L^*$ dont on se demande s'il appartient au langage L. Une réduction montre que si le langage B est décidable alors il en est de même du langage A. On utilise ensuite l'idée suivante : 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. On aura alors

$$B \text{ décidable} \implies A \text{ décidable} \qquad \text{et} \qquad A \text{ indécidable}$$

et l'on peut en déduire (règle de résolution !) que par conséquent B est indécidable. On notera que le raisonnement est tout aussi valide en remplaçant *décidable* par *récursivement énumérable.*

1

Corollaire 3. *Le langage $L_u = \{\langle M\rangle\#w \mid M$ accepte le mot $w\}$ n'est pas récursif. Plus précisément, son complément $L_{\bar{u}}$ n'est pas re.*

Démonstration. 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 récursivement énumérable alors L_d l'est également, or le théorème 1 nous dit que L_d n'est pas récursivement énumérable, donc $L_{\bar{u}}$ non plus, et par conséquent L_u n'est pas récursif. □

Corollaire 4. *Le langage $L_{halt\epsilon} = \{\langle M\rangle \mid M$ s'arrête quand on la lance sur l'entrée vide$\}$ n'est pas récursif.*

Démonstration. Nous allons réduire L_u à $L_{halt\epsilon}$. Etant donnée $\langle M\rangle\#w$ une instance de L_u, nous construisons l'instance $\langle M'\rangle$ suivante pour $L_{halt\epsilon}$:

1. on vérifie $\langle M\rangle \in L_{enc}$, si $\langle M\rangle$ n'est pas un encodage valide alors on retourne l'instance $\langle M'\rangle = \langle M\rangle$;
2. 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 [2].

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 récursif alors L_u est récursif, or le théorème 3 nous dit que L_u n'est pas récursif, donc $L_{halt\epsilon}$ n'est pas récursif. □

Voyez-vous la réduction utilisant le théorème 4 pour démontrer le théorème de l'arrêt ?

3.8 Théorème de Rice

Nous avons vu que de nombreuses questions sur les MT sont indécidables. Certaines questions sont clairement décidables, comme par exemple : est-ce qu'une MT donnée a 5

1. Navré pour l'anglicisme, *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).

2. Pour les fous de formalisme : soit $\langle M\rangle\#w$ une instance de L_u avec $M = (Q, \Sigma, \Gamma, \delta, q_0, B, q_F)$ et $w = w_0 \dots w_k$, dans le second cas nous construisons $\langle M'\rangle$ avec $M' = (Q \cup \{q'_0, \dots, q'_{k+1}\} \cup \{q'_{loop}\}, \Sigma, \Gamma, \delta', q'_0, q_F)$ avec $\delta'(q,a) = \delta(q,a)$ pour tout q et a où $\delta(q,a)$ est définie, et $\delta'(q'_i, B) = (q'_{i+1}, w_{k-i}, L)$ pour tout $0 \le i \le k$ afin d'écrire w sur le ruban initiallement vide, et $\delta'(q'_{k+1}, B) = (q_0, B, R)$ pour aller dans l'état initial de M sur le début du mot w, et $\delta'(q,a) = (q'_{loop}, B, R)$ pour tout $q \in Q$ et $a \in \Gamma$ pour lesquels $\delta(q,a)$ est indéfinie, et enfin $\delta'(q'_{loop}, a) = (q'_{loop}, B, R)$ pour tout $a \in \Gamma$.

états ? Il s'avère cependant que **toute question non triviale qui concerne uniquement le langage reconnu par une MT (plutôt que la machine elle-même) est indécidable**. Une question non triviale étant une question qui n'est pas toujours vraie ou toujours fausse.

Définition 5. *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 $L(M_2) \notin P$.*

Théorème 6. *Pour toute propriété non triviale P, il n'existe aucun algorithme pour décider si une MT M vérifie $L(M) \in P$. Autrement dit, $L_P = \{\langle M \rangle \mid L(M) \in P\}$ n'est pas récursif.*

Démonstration. Nous utilisons une réduction depuis L_u. Sans perte de généralité, nous pouvons supposer $\emptyset \notin P$ (sinon on considère le complément de 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 $\langle M' \rangle$ (pour le problème d'appartenance de $L(M')$ à P) avec M' la machine qui :

1. copie son entrée u 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. A partir de là M' simule M, en ignorant u, jusqu'à ce que M entre dans son état final ;
4. si M entre dans son état final, alors le mot u est recopié sur un ruban blanc (que des symboles B) et la machine M_P est simulée sur u. On entre dans un état final si M_P accepte u.

Etant donné n'importe quels $\langle M \rangle$ et w, la machine M' (et donc son code $\langle M' \rangle$) peut effectivement être construite algorithmiquement.

La correction de la réduction ($\langle M \rangle \# w \in L_u \iff \langle M' \rangle \in L_P$) est assurée par les observations :

— si M accepte w, alors M' acceptera exactement les mots u 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 u, donc dans ce cas $L(M') = \emptyset \notin P$ et $\langle M' \rangle \notin L_P$.

On en conclut que si la propriété P est décidable alors L_u est décidable, or le théorème 3 nous dit que L_u n'est pas décidable, donc la propriété P n'est pas décidable. □

Remarque 7. M_1 **simule** M_2 = *M_1 se comporte comme M_2* = *M_1 suit la table de transition de M_2.*

Appliquons le théorème de Rice. Les problèmes suivants sont indécidables :
— *est-ce qu'une MT donnée en entrée accepte tous les mots ?*
— *est-ce que $L(M)$ est un langage régulier pour une MT M ?*
— *est-ce qu'une MT donnée accepte tous les palindromes ?*

Références

[1] M. Sipser. *Introduction to the theory of computation.* Course Technology, 2006.