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 $L_u = \{\langle M\rangle \# w \mid M \text{ accepte le mot } w\}$ 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 $L_{halt\epsilon} = \{\langle M\rangle \mid M \text{ s'arrête quand on la lance sur l'entrée vide}\}$ 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 ?

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 $a \in \Gamma$.

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 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 ?*

3.9 Réductions (oracle)

Les réductions Turing que nous avons définies dans la section 3.7 ont une définition simple, mais sont en fait assez restraintes. Nous proposons maintenant une définition plus générale de la notion de réduction, qui correspond exactement à l'idée d'imaginer que nous puissions résoudre un problème B, et de s'intéresser alors à la résolution d'un problème A.

Définition 8. *Un langage A est* **Turing-réductible** *à un langage B si il existe une machine de Turing avec* **oracle** *B, qui décide le langage A. On note $A \leq^T B$.*

Une machine de Turing avec oracle B est une machine qui peut, au cours de son calcul, obtenir autant de réponses qu'elle souhaite sur des questions d'appartenance au langage B : est-ce que tel $w \in B$? est-ce que tel autre $w' \in B$? Et l'oracle pour le langage B lui donne des réponses oui/non, instantanément. La définition suivante explique comment implémenter formellement cette idée dans le modèle des machines de Turing.

Définition 9. *Une machine de Turing M avec* **oracle** *B a un ruban supplémentaire appelé ruban d'oracle, et trois états spéciaux $q_{question}$, q_{oui} et q_{non}. A chaque fois que M entre dans l'état $q_{question}$, la machine va dans l'état q_{oui} (si $w \in B$) ou q_{non} (si $w \notin B$) avec w le contenu du ruban d'oracle. Les réponses aux questions d'appartenance à B sont données instantanément, et comptent comme une seule étape de calcul.*

Grâce aux machines avec oracle, on peut définir la *calculabilité relative*. C'est l'idée d'une machine branchée à une source d'information (comme par exemple un ordinateur branché à une base donnée, ou au world wide web...) qui est décrite par le langage de l'oracle. Le modèle des machines de Turing sans oracle peut être vu comme un modèle *offline*, alors que le modèle des machines de Turing avec oracle peut être vu comme un modèle *online*.

Définition 10. *L'ensemble des langages décidables avec oracle B est $\{A \mid A \leq^T B\}$.*

Remarque 11. *Une machine de Turing avec oracle, dont l'oracle est un langage récursif, décide un langage qui est également récursif.*

Remarque 12. *Les réductions Turing many-one et Turing (avec oracle) ne sont pas équivalentes :*

- *si A est Turing many-one réductible à B, alors A est Turing réductible à B ;*
- *tout langage récursif est Turing réductible à n'importe quel langage (puisque l'oracle est inutile), mais un langage récursif non vide ne peut pas être Turing many-one réduit au langage $B = \emptyset$ (puisque quels que soient f at w on aura toujours $f(w) \notin B$). Donc par exemple, le langage Σ^* est Turing réductible au langage $\emptyset$, mais pas Turing many-one réductible ;*
- *tout langage est Turing réductible à son complément, mais si A est Turing many-one réductible à B et B est r.e. alors A est également r.e. Par conséquent, le complément du problème de l'arrêt (qui n'est pas r.e.) n'est pas Turing many-one réductible à son complément (le problème de l'arrêt, qui est r.e.).*

3.10 Le problème de correspondance de Post (PCP)

Le problème de correspondance de Post (PCP) est un exemple « simple » de problème indécidable qui ne semble pas (à première vue) parler de machines de Turing...

PCP est un genre de puzzle. Soit Σ un alphabet et $D \subset_{fini} \Sigma^* \times \Sigma^*$ un ensemble fini de dominos, chaque domino étant un couple de mots. Le but est de constituer un *assortiment* des dominos de D, c'est-à-dire une séquence de dominos (les répétitions sont autorisées) telle que la concaténation des premiers mots de chaque paire soit égale à la concaténation des seconds mots de chaque paire. Formellement, un *assortiment* est une suite de $k \in \mathbb{N}$ dominos $\big((w_i, w'_i)\big)_{1 \leq i \leq k}$ telle que $(w_i)_{1 \leq i \leq k} = (w'_i)_{1 \leq i \leq k}$. Bien entendu, certains ensembles de dominos admettent des assortiments, alors que d'autres non.

Exemple 13. *L'ensemble* $\left\{\frac{bc}{ca}, \frac{a}{ab}, \frac{abb}{b}\right\}$ *admet l'assortiment* $\frac{a}{ab}, \frac{bc}{ca}, \frac{a}{ab}, \frac{abb}{b}$ *avec le mot du haut égal au mot du bas. L'ensemble de dominos* $\left\{\frac{a}{ab}, \frac{b}{a}\right\}$ *n'admet pas d'assortiment.*

On peut encoder les dominos par une chaîne de caractères, comme par exemple

$$\langle \left\{\tfrac{bc}{ca}, \tfrac{a}{ab}, \tfrac{abb}{b}\right\} \rangle = \texttt{bc\#ca\#\#a\#ab\#\#abb\#b}$$

et la donner en entrée à une machine de Turing. Soit

$$PCP = \{\langle D \rangle \mid D \text{ admet un assortiment}\}.$$

Il se trouve que l'on peut réduire L_{halte} à PCP, en encodant une machine de Turing M dans un ensemble de dominos D_M, de façon à ce que la construction d'un assortiment des dominos de D_M corresponde forcément à l'exécution de la machine D_M sur l'entrée vide. L'idée est qu'à chaque ajout d'un domino de D_M on effectue une nouvelle étape du calcul de M, et la construction d'un assortiment termine (donc un assortiment existe) si et seulement l'exécution de la machine sur l'entrée vide s'arrête.

Théorème 14. *PCP n'est pas récursif.*

On trouvera une démonstration du théorème 14 dans [1] chapitre 5 section 2.

3.11 Indécidabilité de la logique du premier ordre

Hilbert a posé la question suivante en 1928, à un moment ou les fondements des mathématiques allaient être boulversés : (**Entscheidungsproblem**) est-ce que tout énoncé en logique du premier ordre (un fragment des mathématiques) peut être algorithmiquement [3] démontré ou réfuté ? Hilbert pensait que oui, les mathématiques sont belles et nous allons tout connaître d'elles par ce biais. Cette section montre que non.

(source : `http://kilby.stanford.edu/~rvg/154/handouts/fol.html`)

Rappels sur la logique du premier ordre

Une logique du premier ordre est donnée par un langage (S_f, S_r). À Chaque symbole de fonction et prédicat est associé une arité (les symboles de fonction d'arité 0 sont des constantes). Les termes sont définis récursivement comme suit :

— les variables sont des termes ;
— si $t_1, \ldots, t_n$ sont des termes et si f est une fonction n-aire, alors $f(t_1, \ldots, t_n)$ est un terme.

Les formules sont définies récursivement comme suit :

3. par une procédure automatique, en imaginant remplacer les mathématiciens par des machines...

— si $t_1, \ldots, t_n$ sont des termes et si p est un prédicat n-aire, alors $p(t_1, \ldots, t_n)$ est une formule (atomique) ;
— si ϕ et ψ sont des formules, alors $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$ et $\phi \implies \psi$ sont des formules ;
— si ϕ est une formule et x une variable, alors $\forall x : \phi$ et $\exists x : \phi$ sont des formules.

Le théorème de complétude du calcul de la résolution pour la logique du premier ordre dit que si $\Gamma \models \phi$ alors on peut dériver la close vide $\bot$ de $\Gamma \cup \{\neg\phi\}$ (sous forme clausale, obtenue par mise sous forme prénexe puis skolémisation puis mise sous forme normale conjonctive des formules propositionnelles) avec les règles de factorisation et résolution.

Preuve d'indécidabilité

Théorème 15. *Le problème de savoir si $\Gamma \models \phi$ dans la logique du premier ordre est indécidable.*

Démonstration. Nous allons réduire à ce problème de décision celui de décider le langage

$$L_u = \{\langle M \rangle \# w \mid M \text{ accepte l'entrée } w\}.$$

Etant donnés M et w, nous allons construire Γ et ϕ tels que $\Gamma \models \phi$ si et seulement si $\langle M \rangle \# w \in L_u$. Puisque L_u est n'est pas récursif, on pourra en déduire qu'il ne peut pas exister d'algorithme pour décider si $\Gamma \models \phi$ (sinon on pourrait construire un algorithme pour décider L_u, or le théorème 3 nous dit que c'est impossible).

Soit $M = (Q^M, \Sigma^M, \Gamma^M, \delta^M, q_0^M, B^M, q_F^M)$ et w une instance du problème d'appartenance à L_u. Nous définissons le langage $S = (S_f, S_r)$ avec $S_f = \{(\epsilon, 0)\} \cup \{(a, 1) \mid a \in \Gamma^M\}$ et $S_r = \{(f_q, 2) \mid q \in Q^M\}$: la constante ϵ, un symbole de fonction unaire a pour toute lettre $a \in \Gamma^M$, et un symbole de prédicat binaire f_q pour tout état q de la machine M. Voici l'idée que nous allons suivre : les variables x correspondront à des mots sur Γ^M, ϵ sera utilisé pour le mot vide, $a(w)$ sera le mot aw, et $f_q(x, y)$ signifiera que M, sur l'entrée w, peut atteindre la configuration $\bar{x}qy$ (avec $\bar{x}$ le mot miroir de x).

La formule

$$f_{q_0^M}(\epsilon, w)$$

correspond à la configuration initiale de la machine M sur l'entrée w, qui est atteignable, donc cette formule doit être vraie, nous la plaçons dans nos hypothèses Γ. Attention à la distinction entre le mot w (par exemple $abba$) et le terme de logique correspondant (dans cet exemple $a(b(b(a(\epsilon))))$, que nous écrirons aussi $abba$ pour faciliter la lecture).

Nous allons maintenant faire en sorte que la formule

$$\phi = \exists\, x\, \exists\, y : f_{q_F^M}(x, y)$$

soit vraie dans cette théorie (c'est-à-dire soit une conséquence logique de Γ) si et seulement si $\langle M \rangle \# w \in L_u$ (c'est-à-dire ssi M accepte w). Pour cela nous allons ajouter des formules dans Γ, qui nous permettront de déduire que si une configuration est atteignable (le prédicat correspondant est conséquence logique de Γ), alors la configuration suivante donnée par la fonction de transition δ^M est également atteignable (le prédicat correspondant est conséquence logique de Γ).

Pour toute transition de la forme $\delta^M(q, a) = (p, b, R)$, nous rajoutons dans Γ la formule

$$\forall\, x\, \forall\, y : f_q(x, ay) \implies f_p(bx, y),$$

et pour toute transition de la forme $\delta^M(q,a) = (p,b,L)$, nous rajoutons dans Γ les formules

$$\forall\, x\, \forall\, y : f_q(cx, ay) \implies f_p(x, cby) \text{ pour tout } c \in \Gamma.$$

Pour être rigoureux nous devons aussi ajouter dans Γ les formules suivantes pour tout $q \in Q$:

$$\forall\, x : f_q(x, \epsilon) \iff f_q(x, B)$$
$$\forall\, y : f_q(\epsilon, y) \iff f_q(B, y)$$

La construction de Γ et ϕ est terminée. Puisque M possède un nombre fini de transitions sur un alphabet de taille finie, Γ est de taille finie.

Il nous reste à argumenter que $\Gamma \models \phi$ si et seulement si M accepte le mot w. Pour le sens indirect, si l'on part de l'hypothèse que M accepte w alors il existe une suite de transition de la machine de Turing M à partir de l'entrée w qui atteint l'état final q_F^M, et à chacune de ces transitions nous pouvons faire correspondre une application de la règle de résolution à partir de la formule $f_{q_0^M}(\epsilon, w)$, et puisque M sur l'entrée w atteint l'état final q_F^M alors nous allons pouvoir déduire que $\exists\, x\, \exists\, y : f_{q_F^M}(x, y)$, c'est-à-dire $\Gamma \models \phi$. Pour le sens direct (qui est un peu plus compliqué à prouver formellement), si l'on part de l'hypothèse que $\Gamma \models \phi$, alors il existe une suite d'application des règles de résolution et factorisation telle que l'on dérive ϕ de Γ, et de cette suite d'application nous pouvons en déduire une suite de transitions de la machine de Turing M à partir de l'entrée w (en fait, on peut se rendre compte que la règle de factorisation sera inutile, et que les applications de la règle de résolution partent de $f_{q_0^M}(\epsilon, w)$ pour arriver à ϕ en utilisant à chaque étape une formule qui correspond à une transition de la machine de Turing M). □

Références

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