Oracles et fonctions de conseil.

Une machine de Turing avec un oracle pour L peut décider en un pas de calcul si un mot donné appartient à L.

Définition 1. Une *machine de Turing à oracle* dispose d'un ruban particulier et de trois états particuliers : $q_?$, q_{oui} et q_{non}.

Soit $L \subseteq \Gamma^\star$. La machine M^L se comporte comme une machine de Turing normale, sauf quand elle entre dans l'état $q_?$. Dans ce cas, en notant u le mot sur le ruban de l'oracle,

- ▷ si $u \in L$ alors l'état suivant est q_{oui} ;
- ▷ si $u \notin L$ alors l'état suivant est q_{non}.

Remarque 1. On peut définir aussi des machines *non déterministes* à oracle.

Exemple 1.

1. Étant donnée une formule booléenne

$$\phi(x_1, \ldots, x_n),$$

on voudrait former une assignation qui satisfait ϕ s'il y en a. On peut le faire avec un oracle pour Sat avec l'algorithme de ***recherche préfixe*** :

$a \leftarrow \varepsilon$

tant que $|a| < n$ **faire**
On demande à l'oracle si $\phi(a, \mathtt{0}, x_{|a|+2}, \ldots, x_n) \in \textsc{Sat}$.
si l'oracle dit « oui » **alors** $a \leftarrow a\mathtt{0}$
sinon $a \leftarrow a\mathtt{1}$
retourner a

Si $\phi \in \textsc{Sat}$ alors l'algorithme renvoie une assignation qui satisfait ϕ. Sinon, on renvoie $\mathtt{1}^n$.

1. Soit $A \subseteq \Sigma^\star$. On voudrait reconnaitre

$$L_A := \{\mathtt{1}^n \mid A^{=n} \neq \emptyset\},$$

où $A^{=n} := \{w \in A \mid |w| = n\} = A \cap \Sigma^n$.

On peut le faire en temps polynomial avec une machine non-déterministe équipée d'un oracle pour A.

si $x \neq \mathtt{1}^{|x|}$ **alors Rejeter**
Choisir $y \in \Sigma^{|x|}$ de manière non-déterministe.
Demander à l'oracle si $y \in L$.
si l'oracle dit « oui » **alors**
Accepter
sinon
Rejeter

1 Relativisation.

On a prouvé les théorèmes de hiérarchie en utilisant la méthode de diagonalisation. On peut étudier les limites de cette méthode grâce à la ***relativisation***.

Théorème 1 (Baker, Gill, Solovay)**.** Il existe des oracles A et B tels que $\mathsf{P}^A = \mathsf{NP}^A$ et $\mathsf{P}^B \neq \mathsf{NP}^B$.

On prouvera ce théorème dans la suite de cette section.

Définition 2. Soit A un oracle.

On note P^A la classe des langages reconnaissable en temps polynomial par une machine déterministe avec un oracle pour A.

On note NP^A la classe des langages reconnaissable en temps polynomial par une machine non-déterministe avec un oracle pour A.

On étend cette notation avec $\mathscr{C}$ une classe de langages, on définit

$$\mathsf{P}^{\mathscr{C}} := \bigcup_{L \in \mathscr{C}} \mathsf{P}^L \qquad \mathsf{NP}^{\mathscr{C}} := \bigcup_{L \in \mathscr{C}} \mathsf{NP}^L.$$

On peut ainsi défini P^{NP}, $\mathsf{NP}^{\mathsf{NP}}$, *etc.*

Remarque 2. On a $\mathsf{P} \neq \mathsf{EXPTIME}$ par le théorème de hiérarchie en temps. La preuve relativise $\mathsf{P}^A \neq \mathsf{EXPTIME}^A$ pour tout oracle A. C'est une propriété des preuves par diagonalisation.

Remarque 3. L'interprétation du théorème de Baker-Gill-Solovay est que l'on ne peut pas résoudre « P *vs.* NP » par une méthode de diagonalisation.

La preuve que $\mathsf{IP} = \mathsf{PSPACE}$ ne se relativise pas : il existe A tel que $\mathsf{IP}^A \neq \mathsf{PSPACE}^A$.

Proposition 1. Il existe un oracle A tel que $\mathsf{P}^A = \mathsf{NP}^A$.

Preuve. On choisit $A = \mathrm{QBF}$, et on a bien $\mathsf{P}^{\mathrm{QBF}} = \mathsf{PSPACE}$.

- ▷ On a $\mathsf{PSPACE} \subseteq \mathsf{P}^{\mathrm{QBF}}$ par PSPACE-complétude de QBF.
- ▷ Supposons $L \in \mathsf{P}^{\mathrm{QBF}}$. On a $L \in \mathsf{PSPACE}$ car on peut répondre aux questions posées à l'oracle en espace polynomial puisque $\mathrm{QBF} \in \mathsf{PSPACE}$. D'où $\mathsf{P}^{\mathrm{QBF}} \subseteq \mathsf{PSPACE}$.

Et, on a $\mathsf{NP}^{\mathrm{QBF}} = \mathsf{PSPACE}$. On doit juste montrer que $\mathsf{NP}^{\mathrm{QBF}} \subseteq$

PSPACE. On énumère tous les chemins de calcul d'une machine NP. □

Proposition 2. Il existe un oracle A tel que $\mathsf{P}^A \neq \mathsf{NP}^A$.

Preuve. Pour tout A, on note $L_A := \{\mathtt{1}^n \mid A^{=n} \neq \emptyset\}$. On a que $L_A \in \mathsf{NP}^A$. On va construire A tel que $L_A \notin \mathsf{P}^A$.

Soit $(M_i)_{i\geq 1}$ une énumération des machines à oracle telle que M_i fonctionne en temps au plus $p_i(n)$ sur les entrées de taille n, pour un certain polynôme p_i.[1] On va construire A tel que L_A n'est pas reconnu par ${M_i}^A$. Construisons ce A par étapes, où l'étape i assurera que L_A n'est pas reconnu par ${M_i}^A$.

On part de $A = \emptyset$. À chaque étape, on peut ajouter des éléments dans A ou $\bar{A}$: pour un mot $w \in \Sigma^\star$, à l'étape i, il est soit dans A, soit dans $\bar{A}$, soit on ne sait pas encore. « À la fin » (étape ω), les mots non-décidés sont mis dans $\bar{A}$.

À l'étape i, on s'assure que ${M_i}^A$ donne la mauvaise réponse sur l'entrée $\mathtt{1}^{n_i}$. On choisit n_i de sorte que :

1. on ait $2^{n_i} > p_i(n_i)$ (ce qui est toujours vrai pour un n_i assez grand) ;
2. l'entier n_i est strictement supérieur à la longueur de tous les mots pour lesquels on a déjà fait une décision $(*)$.

On simule M_i sur l'entrée $\mathtt{1}^{n_i}$. Pour chaque requête à l'oracle, on donne une réponse consistante avec les décisions précédentes. Plus précisément, on répond « oui » uniquement pour les chaînes w qu'on a déjà décidé dans A (sinon, on ne répond « non » et w est placé dans $\bar{A}$). À la fin du calcul sur $\mathtt{1}^{n_i}$,

1. si M_i rejette, on décide de mettre dans A un mot de longueur n_i sur lequel M_i n'a fait aucune réponse (possible car $2^{n_i} > p_i(n_i)$) ;

2. si M_i accepte $\mathtt{1}^{n_i}$, on décide que $A^{=n_i} = \emptyset$, ce qui est consistant avec les décisions précédentes par $(*)$, la seconde propriété pour le choix de n_i.

En conclusion, ${M_i}^A(\mathtt{1}^{n_i})$ accepte ssi $A^{=n_i} = \emptyset$. Et donc ${M_i}^A$ se trompe sur $\mathtt{1}^{n_i}$. $\square$

On en conclut le théorème de Baker-Gill-Solovay.

2 Fonctions de conseil.

Définition 3. Une *fonction de conseil* est une fonction $f : \mathbb{N} \to \{\mathtt{0}, \mathtt{1}\}^\star$ où, sur l'entrée $x \in \{\mathtt{0}, \mathtt{1}\}^n$ le « conseil » $f(n)$ est donné à la machine de Turing (en plus de l'entrée).

Définition 4. Soit $\mathscr{C}$ une classe de langages, et $\mathscr{F}$ une classe de fonctions de conseil. On définit $\mathscr{C}/\mathscr{F}$ l'ensemble des langages A tels qu'il existe $B \in \mathscr{C}$ et une fonction de conseil $f \in \mathscr{F}$ telle que :

$$\forall x \in \{\mathtt{0}, \mathtt{1}\}^\star, \qquad x \in A \qquad \iff \qquad \langle x, f(|x|)\rangle \in B.$$

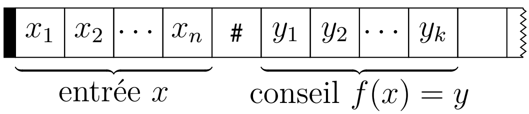

Figure 1 | *Ruban d'entrée d'une machine avec fonction de conseil*[2]

Exemple 2. On note

$$\mathsf{poly} := \left\{ f \;\middle|\; \begin{array}{l} \text{il existe } p \text{ un polynôme tel} \\ \text{que } |f(n)| \leq p(n) \text{ pour tout } n \end{array} \right\},$$

1. On énumère les machines polynomiales à horloge, c'est-à-dire que l'on énumère toutes les machines qui s'arrêtent en temps n^k pour tout $k \in \mathbb{N}$. On force l'arrêt au bout de n^k étapes de calculs. On parle de « *clocked Turing machines* » en anglais.

et

$$\mathsf{log} := \left\{ f \;\middle|\; \begin{array}{l} \text{il existe } c \text{ une constante telle} \\ \text{que } |f(n)| \le c \cdot \log n \text{ pour tout } n \ge 1 \end{array} \right\}.$$

On obtient donc des classes P/poly, $\mathsf{NP}/\mathsf{poly}$, P/log, NP/log, *etc.*

Théorème 2. Pour un problème $A \subseteq \{\mathtt{0}, \mathtt{1}\}^\star$, les deux propriétés suivantes sont équivalentes :

1. $A \in \mathsf{P}/\mathsf{poly}$;
2. A peut être résolu par une famille de circuits booléens de taille polynomiale.

Preuve. Pour « 1. $\implies$ 2. », soit $A \in \mathsf{P}/\mathsf{poly}$, et $x \in \{\mathtt{0}, \mathtt{1}\}^n$ avec

$$x \in A \qquad \iff \qquad \langle x, f(n) \rangle \in B,$$

où $B \in \mathsf{P}$. On a que B peut être résolu par une famille de circuits booléens $(C_n)_{n \ge 1}$. Supposons que $\langle x, y \rangle = \mathtt{1}^{|x|}\mathtt{0}xy$. Avec $y = f(n)$, cette chaîne est de longueur $2n+1+|f(n)|$, et il suffit de considérer $C_{2n+1+|f(n)|}$.

Pour « 2. $\implies$ 1. », soit A résolu par une famille de circuits $(C_n)_{n \ge 1}$ de taille polynomiale. On donne comme conseil $f(n)$ une description du circuit C_n. On peut conclure que $A \in \mathsf{P}/\mathsf{poly}$ puisque le problème d'évaluation ValCirc est dans P. □

2. où on construit $\langle \cdot, \cdot \rangle$ avec un séparateur `#`.