# Le théorème de Ladner

Éloan Rapion

Tous nos langages se baseront sur l'alphabet $\Sigma = \{0, 1\}$ (on peut de toute manière coder tout alphabet fini avec cet alphabet, avec une taille du mot obtenu linéaire en la taille du mot initial). On assimile un problème de décision au langage des mots acceptés. On note $\mathbf{R}$ l'ensemble des langages récursifs. On définit une machine de Turing avec oracle :

**Définition:**
Une machine de Turing avec oracle est une machine de Turing à deux rubans, le ruban de travail et le ruban d'oracle, avec trois états particuliers ?, $Y$ et $N$.

Si $A$ est un langage et $M$ une machine de Turing avec oracle, on pose alors $M(A)$ une machine dont le fonctionnement est similaire à celui de la machine de Turing $M$, sauf que lorque l'état ? est invoqué, la machine se place dans l'état $Y$ ou $N$ selon si le mot écrit sur le ruban d'oracle est dans $A$ ou non. Cette opération n'est pas prise en compte lorsque l'on considère le temps d'exécution de $M$. Il est dit polynomial si un polynôme $p(n)$ majore le temps d'exécution de $M(A)$ sur tout mot de longueur $n \in \mathbb{N}$ et pour tout langage $A$.

Les machines de Turing avec oracle formalisent la notion de réduction dans le contexte des machines de Turing. Un langage $A$ se réduit à un langage $B$ s'il existe une machine de Turing avec oracle $M$ telle que $M(B)$ reconnait $A$. Si $M$ s'exécute en temps polynômial, on dit sue la réduction est polynomiale et on note $A \leq B$. Remarquons que c'est un préordre sur l'ensemble des langages : on note $\equiv$ la relation d'équivalence associée. On note $A < B$ si on a $A \leq B$ mais pas $B \leq A$. Enfin, si on restreint à la classe $NP$, $\mathbf{P}$ et $\mathbf{NP}$-complet sont des classes d'équivalence pour $\equiv$, et elles sont respectivement le minimum et le maximum, pour les classes d'équivalence avec l'ordre induit par $\leq$.

On pose $\mathbf{NPI}$ la classe des langages de $\mathbf{NP}$ qui ne sont ni dans $\mathbf{P}$, ni $\mathbf{NP}$-complet. Notre objectif est de montrer le théorème de Ladner : si $\mathbf{P} \neq \mathbf{NP}$, alors $\mathbf{NPI}$ est non vide. La preuve proposée est celle de l'article publié par Ladner en 1975 [1].

**Lemme 1:** Soit $A$ et $B$ deux langages ne différant que sur un ensemble fini $C$ (c'est-à-dire $A \cup C = B \cup C$). Alors $A \equiv B$. En particulier $A \in \mathbf{P}$ si et seulement si $B \in \mathbf{P}$.

***Preuve:*** On va décrire le fonctionement d'une machine de Turing à oracle $M(B)$ pour réduire $A$ à $B$ en temps polynomial.

D'abord on lit le mot écrit en entrée pour savoir s'il est dans $C$. Pour cela, il faut un état pour chaque préfixe de chaque mot de $C$.

Si on arrive au symbole blanc dans un état correspondant à un mot de $C$, on s'arrête dans un état acceptant ou non selon si ce mot est dans $A$ ou non.

Si on arrive à une lettre qui ne correspond pas à un préfixe d'un mot de $C$, ou qu'on arrive au symbole blanc dans un état qui ne correspond pas à un mot de $C$, on appel l'oracle et on renvoie sa réponse.

Alors comme $C$ est fini et que chaque mot a un nombre fini de préfixes, on a bien une machine avec un nombre fini d'états. De plus elle s'exécute en temps linéaire, donc la réduction est polynômiale. L'équivalence s'obtient car les rôles de $A$ et $B$ sont symétriques. Enfin l'affirmation sur $\mathbf{P}$ résulte du fait que $\mathbf{P}$ est une classe d'équivalence de $\equiv$. □

**Lemme 2:** Soit $A \in \mathbf{R} \setminus \mathbf{P}$. Il existe $B \in \mathbf{R} \setminus \mathbf{P}$ tel que $B < A$.

***Preuve:*** Soit $A \in \Sigma^* \setminus \mathbf{P}$. On pose $N$ une machine de Turing reconnaissant $A$. On va définir une fonction calculable en temps polynomial $T : \mathbb{N} \to \mathbb{N}$ et un langage $B = \{w \in A, T(|w|) \text{ est pair}\}$.

Comme les machines de Turing peuvent être encodées par des mots, on peut faire une suite calculable $P_i$ parcourant toutes les machines de Turing. Comme il existe une bijection calculable de $\mathbb{N}$ vers $\mathbb{N}^2$, on peut de plus exiger que chaque machine de Turing soit représentée une infinité de fois dans la suite. De même, on peut définir une suite $M_i$ parcourant toutes les machines de Turing avec oracle, chacune une infinité de fois.

On pose un bon ordre sur $\Sigma^*$ : un mot est supérieur à un autre si son nombre de lettres est strictement supérieur. Si deux mot on le même nombre de lettres, on considère alors l'ordre lexicographique (avec $0 < 1$).

L'objectif est le suivant : pour tout $n \in \mathbb{N}$, $P_{n/2}$ ne doit pas reconnaitre $B$ si $n$ est pair, et $M_{(n-1)/2}(B)$ ne doit pas reconnaitre $A$ si $n$ est impair. Ainsi, d'une part $B$ ne sera pas dans $\mathbf{P}$, d'autre part $A$ ne se réduira pas à $B$ en temps polynomial.

Pour définir $T$, on va faire en sorte que $T$ soit une fonction croissante. Pour tout $i \in \mathbb{N}$, $T$ vaudra $2i$ assez longtemps pour garantir que $P_i$ ne reconnaitra pas le langage $B$, ou en un temps trop grand. Ensuite, $T$ vaudra $2i + 1$ assez longtemps pour que $M_i(B)$ ne reconnaisse pas le langage $A$, ou en un temps trop grand.

Soit $m \in \mathbb{N}$. Voici comment calculer $T(m)$. On pose $T(0) = 0$. Sinon $m \in \mathbb{N}^*$. on calcule alors $T(m-1)$. Deux cas se présentent :

- Si $T(m-1)$ est pair, c'est que l'on cherche à contredire "$P_i$ reconnait $B$ et a une complexité inférieure à $i + n^i$", avec $2i = T(m-1)$. Pour tous les mots, dans l'ordre défini, on exécute $P_i$ d'une part, et on détermine leur appartenance à $B$ au moyen de $N$ et $T$ d'autre part. On s'arrête cependant après $m$ étapes de calcul. Si après cela, une exécution de $P_i$ sur un mot $w$ a dépassé $i + |w|^i$ étapes, ou si on a trouvé un mot $w$ tel que $P_i$ se trompe sur l'appartenance de $w$ à $B$, on a garanti que $P_i$ ne reconnait pas $B$ (ou après trop de temps), et on pose $T(m) = T(m-1) + 1$. Sinon on doit mener les calculs plus loin, on pose $T(m) = T(m-1)$.
- Si $T(m-1)$ est impair, c'est que l'on cherche à contredire "$M_i(B)$ reconnait $A$ et a une complexité inférieure à $i + n^i$", avec $2i + 1 = T(m-1)$. On calcule alors $M_i(B)$ et $N$ pour tous les mots dans l'ordre défini. On s'arrête après $m$ étapes de calcul. Si après cela, une exécution de $M_i(B)$ sur un mot $w$ a dépassé $i + |w|^i$ étapes, ou si on a trouvé un mot $w$ tel que $M_i(B)$ et $N$ ne donnent pas la même réponse, on a garanti que $M_i(B)$ ne reconnait pas $A$ (ou a une trop grande complexité), et on pose $T(m) = T(m-1) + 1$. Sinon on doit mener les calculs plus loin, on pose $T(m) = T(m-1)$.

Comme on limite pour chaque entrée le nombre d'étapes de calcul, l'exécution de $T$ se fait en temps polynomial.

Montrons $B \leq A$. À partir de $w$, on peut déterminer si $w$ est dans $B$ en calculant $T(w)$, en déterminant sa parité, et en déterminant si $w$ est dans $A$.

Au vu de la construction faite, pour montrer $B \notin \mathbf{P}$ et $B < A$, il suffit de montrer que $T$ n'est pas stationnaire. En effet, s'il existe une machine de Turing polynomiale $L$ reconnaissant $B$, sa complexité est majorée par $i+n^i$ pour un entier naturel $i$. Comme $L$ est une infinité de fois dans la suite des $P_i$, il existe $j \geq i$ tel que $P_j = L$. Alors au vu de sa définition précédante, $T$ ne peut pas dépasser la valeur $2i$. Le raisonnement est identique pour $B < A$.

- Supposons que $T$ stationne sur une valeur $2i$, à partir d'un entier naturel $k$. Alors par définition de $B$, pour tout mot $w$ de taille supérieure à $k$, $w$ est dans $A$ si et seulement si $w$ est dans $B$. De plus, on sait que $P_i$ reconnait $B$, et a une complexité majorée par $i+n^i$, donc est polynomiale. Par conséquent $B \in \mathbf{P}$, et avec le lemme 1, $A \in \mathbf{P}$, ce qui est absurde.
- Supposons que $T$ stationne sur une valeur $2i+1$, à partir d'un entier naturel $k$. Alors pour tout mot $w$ de taille supérieure à $k$, $w \notin B$. Ainsi $B$ est fini, et comme $\emptyset \in \mathbf{P}$, par le lemme 1, on a $B \in \mathbf{P}$. De plus $M_i(B)$ a une complexité majorée par $i+n^i$, donc on peut modifier $M_i$ en lui ajoutant un compteur limitant son nombre d'étapes de calcul pour garantir que $M_i(D)$ est polynomiale pour tout langage $D$, sans changer son exécution pour $D = B$. Enfin $M_i(B)$ reconnait $A$, donc $A \leq B$. Ainsi $A \in \mathbf{P}$, ce qui est absurde.

Finalement, $B$ est récursif car se réduit à $A$, qui est récursif. □

**Théorème (Ladner):**
On a $\mathbf{P} = \mathbf{NP}$ si et seulement si $\mathbf{NPI} = \emptyset$.

***Preuve:*** Si $\mathbf{P} = \mathbf{NP}$, on a $\mathbf{NPI} \subset \mathbf{NP} \setminus \mathbf{P} = \mathbf{P} \setminus \mathbf{P} = \emptyset$.

Réciproquement, on procède par contraposée. On pose $A \in \mathbf{NP}$-complet (par exemple $A = \mathbf{SAT}$). Alors si $\mathbf{P} \neq \mathbf{NP}$, on a $A \in \mathbf{NP} \setminus \mathbf{P}$. De plus $\mathbf{NP} \subset \mathbf{R}$. Donc d'après le lemme 2, il existe un langage $B \notin \mathbf{P}$ tel que $B < A$. Comme $B \leq A$ et $A \in \mathbf{NP}$, on a $B \in \mathbf{NP}$. Comme $A$, qui est dans $\mathbf{NP}$, ne se réduit pas à $B$, on a $B \notin \mathbf{NP}$-complet. Ainsi $B \in \mathbf{NPI}$. □

# Références

[1] Richard E Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1) :155–171, 1975.