Théorème de Cook

Julien Devevey

2018-2019

Ref : Carton - Langages formels, Calculabilité et Complexité p.203

Théorème 1. SAT est NP-Complet.

Démonstration.

SAT est bien dans NP : étant donné une assignation de chaque variable, on vérifie linéairement si une formule est évaluée à True.

On montre ensuite que SAT est NP-Difficile. Soit alors A un problème de NP, on va réduire A à SAT. On note $\mathcal{M}$ une machine de Turing non-déterministe qui décide A en temps polynomial. Soit w une entrée et $n = |w|$ sa taille. Quitte à modifier légèrement $\mathcal{M}$, on peut supposer que le calcul s'effectue en un temps exactement n^k pour un certain $k \in \mathbb{N}$ pour toute entrée de taille n. La machine utilise alors au plus n^k cellules du ruban, et quitte à rajouter des $\sharp$ au langage des configurations, on suppose que les configurations sont de taille exactement n^k. L'objectif est alors de trouver une formule φ_w qui code l'existence de toutes ces configurations formant un calcul acceptant sur w.

$\forall (i,j) \in [\![0, n^k]\!]^2, \forall a \in A = \Gamma \cup Q, x_{i,j,a}$ est la variable qui va coder le fait que la j-ème cellule contienne a dans la i-ème configuration, avec Γ le langage du ruban et Q les états de la machine $\mathcal{M}$. Au total, on vient de définir $|A|n^{2k+2}$ variables, ce qui est polynomial en n. On a donc les variables pour écrire φ_w. On pose :

$$\varphi_0 = \bigwedge_{0 \leq i,j \leq n^k} \left[\left(\bigvee_{a \in A} x_{i,j,a} \right) \wedge \left(\bigwedge_{a,a' \in A, a \neq a'} (\bar{x}_{i,j,a} \vee \bar{x}_{i,j,a'}) \right) \right]$$

Cette formule représente le fait que chaque cellule ne doit contenir, pour chaque configuration, qu'un seul élément de A (la partie de gauche vérifie qu'il y en a au moins un, et à droite qu'il y en a au plus un). On pose une deuxième formule :

$$\varphi_1 = x_{0,0,q_0} \wedge \left(\bigwedge_{i=1}^{n} x_{0,i,w_i} \right) \wedge \left(\bigwedge_{i=n+1}^{n^k} x_{0,i,\sharp} \right)$$

Cette formule correspond alors à la vérification de la première configuration : on vérifie qu'on est dans l'état q_0, qu'on est placé au début du mot w et que celui-ci

est bien écrit sur le ruban, puis complété par des ♯. On définit maintenant une troisième formule, qui elle sert à vérifier qu'à la fin de l'exécution on se situe bien dans un état final. Elle s'écrit :

$$\varphi_2 = \bigvee_{q \in F} \left(\bigvee_{0 \leq j \leq n^k} x_{n^k, j, q} \right)$$

Il nous reste à écrire une dernière formule, qui rendra compte que l'on passe bien d'une configuration à la suivante à chaque étape. Remarquons déjà que le contenu d'une cellule j dans la configuration i ne dépend que du contenu des trois cases aux positions $j-1, j, j+1$ dans la configuration $i-1$. De plus, ce contenu ne change que si l'une de ses trois cases contient l'état actuel de la machine, sinon il reste identique. Alors on peut vérifier si la configuration i est obtenue à partir de la configuration $i-1$ seulement en regardant le contenu des fenêtres de taille 2×3. Or le nombre de ces contenus possibles ne dépend que de l'alphabet et des transitions de la machine : c'est un nombre fixe ne dépendant pas de n. On écrit donc une conjonction utilisant au plus six variables pour chaque configuration, ce qui donne un nombre polynomial de conjonctions, dont on nomme la conjonction φ_3. Alors $\varphi_w = \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ est une formule sous forme CNF qui est vraie si et seulement si w est accepté par $\mathcal{M}$. □

Corollaire 2. 3-SAT est NP-Complet.

Démonstration.

On va écrire une réduction polynomiale de SAT vers 3-SAT, ce qui permettra de dire que 3-SAT est NP-Complet.

Soit une formule φ. On considère son arbre syntaxique, c'est à dire l'arbre étiqueté par les opérateurs $\neg, \wedge, \vee$ dont les feuilles sont les variables de φ. On va créer une formule φ' qui est sous forme normale conjonctive et possède au plus trois littéraux par clause. Pour ça, on prend comme variable toutes les variables qui sont étiquettes de feuilles de l'arbre précédent, et pour chaque noeud interne on crée une nouvelle variable. Ensuite, si le noeud est étiqueté par $\neg$, on associe l'égalité $x_i = \bar{x}_j$ où x_j, si c'est $\wedge$ on lui associe $x_i = x_j \wedge x_k$, et pour $\vee$, $x_i = x_j \vee x_k$, où x_i est la variable associée au noeud et x_j, x_k celles associées à ses fils. On considère φ' qui est la conjonction de toutes ces égalités. Enfin, on remplace les égalités par les formules suivantes :

$$\begin{aligned} x = \bar{y} &\equiv (x \vee y) \wedge (\bar{x} \vee \bar{y}) \\ x = (y \wedge z) &\equiv (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee z) \\ x = (y \vee z) &\equiv (x \vee \bar{y}) \wedge (x \vee \bar{z}) \wedge (\bar{x} \vee y \vee z) \end{aligned}$$

La taille de φ' est proportionnelle à celle de φ car la taille de l'arbre syntaxique est proportionnelle à la fois à la taille de φ et celle de φ'. De plus, on peut satisfaire φ si et seulement si φ' est satisfiable. □