Hugo Salou – ENS L3

NP-complétude de 3-sat

Dans ce document, on démontrer la **NP**-complétude du problème de satisfiabilité d'une formule sous 3-CNF, noté 3-SAT :

3-SAT : | **Entrée.** Une formule φ sous forme 3-CNF
Sortie. La formule φ est-elle satisfiable ?

I. Quelques rappels.

On se fixe un ensemble fini de variables $\mathcal{Q}$. On notera $\mathbb{B} = \{\boldsymbol{V}, \boldsymbol{F}\}$ l'ensemble des booléens.

I.1. Valuations & satisfiabilité.

Une *valuation* (ou un *environnent propositionnel*) est une fonction de la forme $\rho : \mathcal{Q} \to \mathbb{B}$. À une variable, on assigne vrai ou faux.

Pour une formule logique φ, on associe $[\![\varphi]\!] : \mathbb{B}^{\mathcal{Q}} \to \mathbb{B}$ une *fonction booléenne*. À une valuation, on associe donc vrai ou faux.

Une formule φ est satisfiable si on peut l'interpréter à vrai, *i.e.* s'il existe une valuation ρ telle que $[\![\varphi]\!](\rho) = \boldsymbol{V}$.

Attention, il ne faut pas confondre *satisfiable* ($\exists \rho$) et *tautologique/ valide* ($\forall \rho$).

I.2. Formes normales conjonctives.

Une formule sous n-CNF (*conjunctive normal form*), c'est une formule de la forme

$$\cdots \land \underbrace{(\ell_1 \lor \ell_2 \lor \cdots \lor \ell_k)}_{k \leqslant n} \land \cdots,$$

où chaque ℓ_i est un *littéral* donc soit $p \in \mathcal{Q}$ soit $\neg p$ avec $p \in \mathcal{Q}$.

Le terme dans l'accolade est appelé *clause*. La taille (*i.e.* le nombre de littéraux) de chacune des clauses est inférieure à n.

On n'a pas de contraintes sur le nombre de clauses (*i.e.* on peut avoir autant de $\wedge$ que l'on veut).

Le problème n-SAT, c'est, dans chaque clause, choisir (au moins) un littéral pour le valuer à vrai.

I.3. Le problème SAT.

Le problème

SAT : | **Entrée.** Une formule φ
Sortie. La formule φ est-elle satisfiable ?

est **NP**-complet. C'est le théorème de *Cook-Levin* et il est admis en MP2I/MPI.

Dans ce théorème, il n'y a pas de contrainte sur la forme de φ.

Pour résoudre ce problème, on applique l'algorithme de Quine. C'est un algorithme force brute légèrement optimisé.

Algorithme de Quine

On choisit $x \in \mathrm{vars}(\varphi)$ (par contrainte ou au hasard).

Première tentative : $x \leftarrow$ ***V***
On tente de résoudre φ avec $x \leftarrow$ ***V***.
Si on réussit, on s'arrête : la formule est satisfiable.

Seconde tentative : $x \leftarrow$ ***F***
On tente de résoudre φ avec $x \leftarrow$ ***F***.
Si on réussit, on s'arrête : la formule est satisfiable.

Dernier cas
Si les deux tentatives ont raté, la formule n'est pas satisfiable.

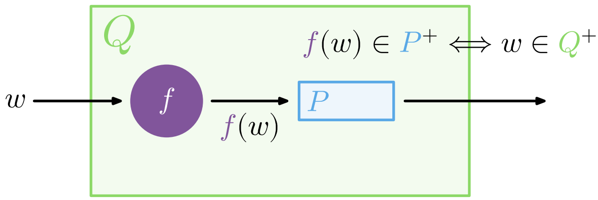
I.4. Réductions polynomiales.

On notera $\mathscr{E}_P$ l'ensemble des entrées d'un problème P. On notera P^+ l'ensemble des *instantes positives* de P, c'est-à-dire les solutions du problème.

On dit qu'un problème Q *se réduit à* P dès lors qu'il existe une fonction $f : \mathscr{E}_Q \to \mathscr{E}_P$ *calculable en temps polynomial* telle que :

$$w \in Q^+ \iff f(w) \in P^+.$$

On notera alors $Q \leqslant_{\mathrm{p}} P$.



Lorsqu'on a $Q \leqslant_{\mathrm{p}} P$, il faut le comprendre par Q est *plus simple à résoudre* que P.

I.5. Problèmes NP-difficiles.

Les problèmes **NP**-difficiles sont les problèmes plus compliqués à résoudre que tous les problèmes dans **NP** (*i.e.* plus compliqués que tous les problèmes vérifiables en temps polynomial).

Un problème P est **NP**-difficile si, quel que soit $Q \in \mathbf{NP}$, on a la réduction polynomiale $Q \leqslant_{\mathrm{p}} P$.

Pour démontrer qu'un problème P est **NP**-difficile, il suffit de trouver un problème Q **NP**-difficile tel que $Q \leqslant_{\mathrm{p}} P$.

S'il existe un problème **NP**-difficile plus simple que le problème P alors P est **NP**-difficile.

I.6. NP-complétude ?

Un problème est **NP**-complet s'il est **NP**-difficile et qu'il est dans la classe **NP**.

Les problèmes **NP**-complets sont les problèmes les plus difficiles de la classe **NP**.

II. Le problème 3-SAT.

Pour démontrer la **NP**-complétude de 3-SAT, on a deux propriétés à démontrer :
(1) le problème 3-SAT est dans NP (la partie simple) ;
(2) le problème 3-SAT est **NP**-difficile (la partie complexe).

II.1. 3-SAT est dans NP.

Là, c'est la partie simple : vérifier qu'une formule est satisfiable. Pour cela, on a le droit à des données, un *certificat*.

Dans notre cas, on peut choisir comme certificat ρ une valuation. Pour vérifier que $[\![\varphi]\!](\rho) = \boldsymbol{V}$, il suffit de calculer $[\![\varphi]\!](\rho)$.

Cette vérification se réalise en temps polynomial (en temps linéaire, même). D'où, 3-SAT est dans **NP**.

II.2. 3-SAT est NP-difficile.

On procède par réduction au problème SAT. Pour cela, on commence par se donner une formule φ et on va construire une formule ψ sous 3-CNF telle que φ est satisfiable si, et seulement si, ψ l'est.

La difficulté vient du fait que φ n'a pas de contrainte sur sa forme. Par exemple, si on suppose que φ est sous CNF, alors il faut réussir à transformer une clause de taille inconnue en une 3-clause.

L'idée de cette preuve, est qu'on va s'intéresser aux sous-formules de la formule φ. Pour cela, pour chaque sous-formule $\vartheta \subseteq \varphi$,[1] on construit deux objets

[1]On utilise cette notation ensembliste, même si une formule n'est pas un ensemble.

- une variable propositionnelle x_ϑ ;
- une formule K_ϑ.

L'idée est qu'on **ne peut pas** inclure une sous-formule (qui est plus qu'un littéral) directement dans K_φ. On veut se limiter à des formules simples, pour pouvoir construire simplement la 3-CNF.

Ainsi, à la place de sous-formules directement, on donne des relations sur les x_ϑ pour $\vartheta \subseteq \varphi$.

En ajoutant les x_ϑ, on définit un nouvel ensemble de variable

$$\mathcal{Q}' = \mathcal{Q} \cup \{x_\vartheta \mid \vartheta \subseteq \varphi\}.$$

II.2.a. Définition des K_ϑ puis de Ω.

On définit :

- pour $\vartheta = p$ avec $p \in \mathcal{Q}$, on pose $K_\vartheta = p$;
- pour $\vartheta = \top$, on pose $K_\vartheta = p \vee \neg p$;
- pour $\vartheta = \perp$, on pose $K_\vartheta = p \wedge \neg p$;
- pour $\vartheta = \neg\gamma$, on pose $K_\vartheta = \neg x_\gamma$;
- pour $\vartheta = \gamma \wedge \delta$, on pose $K_\vartheta = x_\delta \wedge x_\gamma$;
- pour $\vartheta = \gamma \vee \delta$, on pose $K_\vartheta = x_\delta \vee x_\gamma$;
- pour $\vartheta = \gamma \to \delta$, on pose $K_\vartheta = x_\delta \vee \neg x_\gamma$.

Tous les K_ϑ sont des 2-CNF.

Posons la formule Ω, une formule sous 3-CNF équivalente à

$$\Omega \equiv \bigwedge_{\vartheta \subseteq \varphi} (K_\vartheta \leftrightarrow x_\vartheta),$$

où les K_ϑ ne sont pas des variables, mais bien des morceaux de formules.

Justifions de la bonne définition de Ω. On commence par développer le $\leftrightarrow$ en deux implications, puis en une 2-CNF. Ensuite, on remplace K_ϑ dans cette définition. On n'est pas garanti d'obtenir une 3-CNF à ce point, car il peut y avoir des $\wedge$ dans une des futures clauses. Pour cela, on utilise les loi de De Morgan.

Cette formule permet de conserver la « structure » de la formule originelle φ. En effet, si elle est vraie, c'est que toutes les variables x_ϑ coordonnent avec les valuations de K_ϑ.

On définit la formule $\psi = \Omega \wedge x_\varphi$. Cette formule est sous forme normale conjonctive et même 3-CNF. De plus, notre construction est polynomiale en la taille de φ.[2]

Il ne reste qu'une propriété à démontrer :

$$\varphi \text{ satisifiable} \iff \psi \text{ satisifiable},$$

ce que l'on fait par double-implication.

II.2.b. Premier sens de l'implication.

On veut montrer le sens « $\implies$ ». On suppose φ satisfiable, et on montre que ψ satisfiable.

Soit $\rho \in \mathbb{B}^{\mathscr{Q}}$ tel que $[\![\varphi]\!](\rho) = \boldsymbol{V}$.

Là, c'est pas trop compliqué, il suffit de construire une valuation μ sur $\mathscr{Q}'$ telle que $[\![\psi]\!]^\mu = \boldsymbol{V}$.

On pose

$$\begin{aligned} \mu : \mathscr{Q}' = \mathscr{Q} \sqcup (\mathscr{Q}' \setminus \mathscr{Q}) &\longrightarrow \mathbb{B} \\ p \in \mathscr{Q} &\longmapsto \rho(p) \\ x_\vartheta \in \mathscr{Q}' \setminus \mathscr{Q} &\longmapsto [\![\vartheta]\!](\rho). \end{aligned}$$

[2]Justifions ... Il ne faut pas voir les sous-formules ϑ de φ comme des sous-ensembles. On n'en a pas $2^{|\varphi|}$, mais bien un nombre polynomial (en réalité, c'est même linéaire en nombre d'opérateurs[3]) en la taille de φ. Ceci justifie bien que notre construction est polynomiale.

[3]En réalité, pour une formule φ avec n connecteurs d'arité 2, il y en a exactement $2n + 1$. Ceci se montre très bien par induction sur une formule à n connecteurs.

On peut démontrer aisément[4] que l'on a, quelle que soit $\vartheta \subseteq \varphi$, on ait $[\![K_\vartheta]\!](\mu) = [\![x_\vartheta]\!](\mu)$. Ceci est vrai par la définition de μ.

Par conjonction $\land$ et équivalence $\leftrightarrow$, on en déduit que $[\![\psi]\!](\mu) = \boldsymbol{V}$. En effet, l'égalité des valuations de K_ϑ et de x_ϑ donne que l'équivalence $K_\vartheta \leftrightarrow x_\vartheta$ est valué à vrai. Ceci étant vrai pour toute formule, on peut conclure.

Aussi, on a $[\![x_\varphi]\!](\mu) = [\![\varphi]\!](\mu) = [\![\varphi]\!](\rho) = \boldsymbol{V}$.

On en déduit que ma formule ψ est satisfiable si φ l'est. À présent, montrons l'autre implication.

II.2.c. Deuxième sens de l'implication.

On veut montrer le sens « $\Longleftarrow$ ».

On va commencer par montrer : quelles que soient ρ et μ, si on a $[\![\Omega]\!](\mu) = \boldsymbol{V}$ alors $\mu\big(x_\varphi\big) = [\![\varphi]\!](\rho)$. Ce qu'on démontre ici, c'est que Ω assure la « structure » de φ.

Pour démontrer cela, on commence par montrer que, toujours en supposant $[\![\Omega]\!](\mu) = \boldsymbol{V}$, pour toute sous-formule $\vartheta \subseteq \varphi$, on a $\mu(x_\vartheta) = [\![\vartheta]\!](\rho)$. Ceci se démontre simplement par induction sur la sous-formule ϑ.[5] Le résultat sur φ se conclut de cette généralisation.

---

[4]On procède par induction sur la formule ϑ, et on procède cas par cas. Par exemple, pour le cas $\vartheta = \gamma \land \delta$:

$$[\![K_{\gamma\land\delta}]\!](\mu) = [\![x_\gamma \land x_\delta]\!](\mu) = [\![x_\gamma]\!](\mu) \cdot [\![x_\delta]\!](\mu) = [\![\gamma]\!](\mu) \cdot [\![\delta]\!](\mu) = [\![\gamma \land \delta]\!](\mu).$$

[5]Un exemple de cas : si $\vartheta = \gamma \land \delta$ alors

$$\mu(x_\vartheta) \underset{(\star)}{=} [\![K_\vartheta]\!](\mu) = [\![x_\gamma \land x_\delta]\!](\mu) = \mu\big(x_\gamma\big) \cdot \mu(x_\delta) \underset{(\star\star)}{=} [\![\gamma]\!](\mu) \cdot [\![\delta]\!](\mu) = [\![\gamma \land \delta]\!](\mu).$$

L'égalité $(\star)$ est vraie car $[\![K_\vartheta \leftrightarrow x_\vartheta]\!] = \boldsymbol{V}$ par hypothèse. L'égalité $(\star\star)$ est vraie par hypothèse d'induction.

On suppose ψ satisfiable, et on montre que φ satisfiable. Soit une valuation $\mu \in \mathbb{B}^{\mathcal{Q}'}$ telle que $[\![\psi]\!](\mu) = \boldsymbol{V}$.

Ceci implique deux résultats :
- $[\![\Omega]\!](\mu) = \boldsymbol{V}$, on peut donc appliquer les remarques précédentes ;
- et $[\![\varphi]\!](\mu) = \boldsymbol{V}$.

Il suffit de poser $\rho = \mu|_{\mathcal{Q}'}$ et on a bien que $[\![\varphi]\!](\rho) = \boldsymbol{V}$.

Ainsi, la formule φ est satisfiable. Ceci conclut la preuve de l'équivalence, et donc la réduction.

III. Conclusion.

L'idée de la preuve, c'est qu'on peut ajouter autant de $\wedge$ et de variables que l'on veut. La seule contrainte que l'on a dans une 3-CNF, c'est la taille d'une clause.

Cette preuve est au programme de MP2I/MPI et c'est parfois un thème qui tombe à l'oral.